Message from the Technical Program Chair

Welcome to IPtel2001, the second international workshop on Internet Telephony. The workshop follows the successful first workshop held April 12 and 13, 2000 in Berlin, Germany. We received 41 papers, of which we accepted 22. One of the accepted papers was withdrawn by the authors. In keeping with the first workshop, this workshop is drawing a very international group of attendees, with close to half the participants from Europe and Asia. About a third of the attendees are from academic institutions, with the remainder split evenly between research labs and other corporate R&D.

In its commercial incarnation, Internet telephony is now about five years old, although initial practical attempts date back to the early days of the Internet, with experimental work starting around 1974. However, the displacement of the legacy circuit-switched telephone network is taking significantly longer than many people had predicted. With hindsight, this is not surprising, given that Internet telephony is not a single application or protocol, but rather an "eco system" with dozens of protocols and strong legal and regulatory entanglements. The web started out as a self-contained system and then later acquired gateways into existing data applications; Internet telephony has to immediately connect to about a billion telephone terminals. Initially and currently, the main driver for Internet telephony is regulatory arbitrage, bypassing local access charges or inflated international tariffs. Again, Internet telephony differs in this aspect from earlier technologies such as email, the web, text chat and peer-to-peer file sharing, which added fundamentally new capabilities, in return for which users were willing to put up with unreliable and cumbersome service. We are only slowly moving beyond replicating the existing telephone features and habits, realizing that Internet telephony will succeed not by virtue of a small set of "killer application", but rather by the ability to quickly add thousands of small, but locally valuable, applications and its ability to be integrated seamlessly with other Internet-based means of communications. These applications include instant messaging, event notification and back-end services.

The workshop presents a snapshot of some of the technical problems that still need to be addressed in a convincing way. As usual in this field, quality of service contributes the largest set of papers, with the additional requirement of measuring speech quality objectively. Interconnection to other systems, principally to landline and mobile services, as well as between signaling systems, requires thought on how to isolate and abstract the behavior across system boundaries.

As Internet telephony moves from research into deployment, operational issues need to be addressed, exposing the limits of the existing set of Internet services and an Internet architecture evolved towards supporting TCP-based client-server applications.

Initial efforts in service creation for IP telephony focused on APIs and then languages. Now, there is strong interest in allowing features to be deployed across distributed systems.

Each paper was reviewed by at least three members of the technical program committee:

Mauricio Arango, Sun Microsystems
Wulf Bauerfeld, Detecon
Gregory Bond, AT&T Labs – Research
Scott Bradner, Harvard University
Georg Carle, GMD Fokus
Jon Crowcroft, University College London
Christian Huitema, Microsoft
G. S. Kuo, NCU
Thomas La Porta, Bell Laboratories, Lucent Technologies
Thomas Magedanz, GMD FOKUS
William Marshall, AT&T Labs – Research
D. Medhi, University of Missouri-Kansas City

Dave Oran, Cisco
Joerg Ott, Universität Bremen
Brian Rosen, Marconi
Jonathan Rosenberg, dynamicsoft
Henry Sinnreich, WorldCom
Ralf Steinmetz, GMD German National Research Center for Information Technology
Heiner Stüttgen, NEC CCRLE
Wilhelm Wimmreuter, Siemens
Lars Wolf, University of Karlsruhe
Adam Wolisz, TU Berlin

Ashutosh Dutta and Kundan Singh, Columbia University, helped with local arrangements, such as printing the proceedings, registration and catering. Jiri Kuthan served as an informal general chair, offering helpful advice, keeping things on a reasonable schedule and managing the workshop web pages. Columbia University kindly let us use their physical facilities.

We would like to thank the following additional reviewers for their helpful expert comments on some of the papers:

Noopur Bakshi
Wenyu Jiang
Jiri Kuthan
Jonathan Lennox
Paulo Mendes
Henning Sanneck
Kundan Singh
Dorgham Sisalem
Michael Smirnov
Salil Talaulikar
Karthik Venkataraman

Michael Smirnov and Jiri Kuthan, in particular, deserve recognition for their last-minute help when some TPC member reviews were late.

We are pleased to be able to have Prof. Dave Farber (Univ. Pennsylvania) deliver the keynote address on the topic "Predicting the Unpredictable - The FCC and the Future of Telecommunications and the Internet". Prof. Farber has just completed his service as chief technologist to the FCC.

Micromuse was kind enough to provide financial support for the workshop, allowing us to offer a dinner for all attendees.

I hope that you find the technical program rewarding; your questions and comments will help to further the technical work in this nascent community of researchers.

Henning Schulzrinne
Columbia University
Technical Program Chair, IPtel 2001

Technical Program

Monday, April 2, 2001

Tuesday, April 3, 2001

Quality of Service

On the Efficiency of Voice over Integrated Services using Guaranteed Service

Maarten Büchli, Danny De Vleeschauwer, Jan Janssen,
Annelies Van Moffaert and Guido H. Petit

Alcatel Bell, Network Strategy Group
Francis Wellesplein 1
B-2018 Antwerp, Belgium
Tel.: +32 3 2407081 Fax: +32 3 2404888
E-mail: maarten.buchli@alcatel.be

***Abstract*—This paper presents an efficiency study of voice over Integrated Services (IntServ). In particular, the Guaranteed Service class is considered. This service class provides a deterministic upper bound on the end-to-end queuing delay. A method to calculate the optimal packetization delay, and hence, the optimal packet size, is presented. Choosing this packet size involves a trade-off between bandwidth efficiency and delay. Two scenarios are considered in this paper: an IP-phone-to-IP-phone and a gateway-to-gateway scenario. For the latter scenario two multiplexing approaches are evaluated and it is shown that they achieve approximately equal bandwidth efficiency. In addition our results demonstrate that with aggregated voice flows on one reserved bit pipe (gateway-to-gateway scenario) high bandwidth efficiency can be achieved.**

***Index terms*– IntServ, Voice over IP, efficiency**

A. Introduction

1st. Background

Introducing telephony services on IP networks brings its own challenges with respect to voice quality, call set-up time and reliability. The performance of a VoIP network should be comparable to the current PSTN. Especially, the voice quality is of great concern. It depends on many parameters: on the application layer the type of codec, packetization and dejittering delay and on the transport layer the one-way delay, jitter and packet loss. The Quality of Service (QoS) of packet switched networks (e.g. IntServ) controls the transport parameters. The requirements for these parameters are requested by the voice application such that together with the application parameters a certain desired speech quality is obtained. By offering different classes of speech quality, an operator is able to offer telephony services at different prices, targeting different market segments.

2nd. Overview of previous work

Telephony has very stringent delay requirements. When perfect echo control is applied, the mouth-to-ear delay should not exceed 150 ms in order to obtain traditional PSTN quality [10]. Obtaining a small delay comes often at the expense of the bandwidth efficiency (i.e. ratio between the codec rate and the bit rate that has to be reserved). The size of the header of an IP packet is relatively large, often resulting in poor efficiency. The overhead of an IP packet consists of an IP/UDP/RTP header with a total size of 40 bytes. Header compression is not considered since it may only be used at a certain link on the path and not end-to-end. In [1], [2] the efficiency of telephony over packet networks with deterministic queuing delay guarantees was studied for the IP-phone-to-IP-phone scenario. It was shown that the bandwidth efficiency was quite poor. Therefore, it was suggested to make an overreservation to decrease the maximum queuing delay, thus allowing for an increase in packetization delay, and hence, resulting in relatively less header overhead. The excess of the reserved bandwidth can be consumed by best-effort traffic.

This paper extends these results with several contributions. First, for the calculations of the optimal packetization delay the (static) dejittering delay is also taken into account. Second, the bandwidth efficiency is studied for different types of codecs and third, the gateway-to-gateway scenario is considered. In other words, the scenario where a single reservation is made for an aggregate of voice flows.

In this paper we first consider the IP-phone-to-IP-phone scenario with regard to the optimal packet size when a static dejittering delay is used and the bandwidth efficiency when using different types of codecs. Then, we present a study of the gateway-to-gateway scenario. In this case one bit pipe is reserved between two gateways to transport multiple calls. Two methods are considered to multiplex voice flows into a bit pipe. One method is to multiplex IP packets from different flows and the second one is to multiplex voice frames from different voice flows into a single IP packet.

3rd. Contents

In Section B the different components of the mouth-to-ear delay are listed. How to specify the traffic parameters that describe a voice flow is shown in Section C. The IntServ architecture is discussed in Section D. The calculation of the optimal packetization delay for an IntServ network with Weighted Fair Queuing (WFQ) schedulers is described in Section E. The efficiency corresponding to this optimal packetization delay is evaluated in Section F. The paper concludes with Section G.

B. MOUTH-TO-EAR DELAY

An important parameter for interactive voice communications is the mouth-to-ear (M2E) delay. This is the time between the moment the sending party has spoken a word and the moment it is heard at the receiving party. The M2E delay consists of a deterministic and a stochastic part. The deterministic part consists of packetization (T_{pack}), serialization (T_{ser}), propagation (T_{prop}), dejittering ($T_{dejitter}$) and other (encoding, decoding etc.) delay (T_{oth}). The stochastic part consists of the queuing delay in each node. Although the queuing delay is a stochastic quantity, we are only interested in the maximum queuing delay, i.e. the delay of the slowest possible packet, because if this one arrives in time for play-out, all others do too. For this maximum queuing delay the absolute maximum or a reasonable quantile (for instance the 99% quantile) can be chosen. Taking all this into account, the mouth-to-ear delay can be written as

$$\hat{T}_{m2e} = \hat{T}_{queue} + T_{dejitter} + T_{pack} + T_{prop} + T_{ser} + T_{oth} \quad (1)$$

The Guaranteed Service [12] controls the maximum queuing delay and provides a deterministic upper bound for the traffic that fits within the traffic envelop that is specified in the traffic contract. Hence, the other delay components are not under the control of Guaranteed Service. The bound is worst-case, and therefore, not tight. The actual queuing delay observed in the network will usually be (much) smaller. An example of the delay distribution is shown in Figure 1. The tail of the delay is caused by the stochastic queuing delay.

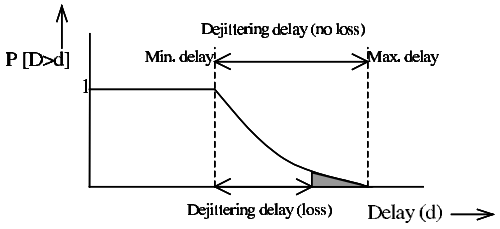


Figure 1: Example of delay distribution

Here the dejittering delay is chosen equal to the maximum queuing delay in order to prevent packets arriving too late for play-out. This is the delay denoted by the arrow with caption 'Dejittering delay (no loss)'. Since the receiver knows this value it is easy to set the dejittering delay. When a certain packet loss is allowed the dejittering delay can be chosen smaller, for instance equal to the delay denoted by the arrow 'Dejittering delay (loss)'. In this case, the packets with a delay within the gray colored area may be lost (depending on the queuing delay of the first packet) because they arrived too late for play-out. However, to estimate the resulting packet loss one has to know different quantiles of the queuing delay, i.e. the shape of the delay distribution. This information is not available in IntServ networks.

The analysis presented in this paper is worst case, i.e. the dejittering delay is chosen equal to the maximum queuing delay. Adaptive dejittering algorithms (see e.g. [11]) also exist, they estimate the queuing delay of the first packet. When perfect dejittering is used the queuing and dejittering delay of each packet is equal to the maximum queuing delay. However, it is questionable whether adaptive dejittering algorithms can converge fast enough during the typical length of a phone call.

C. TRAFFIC SPECIFICATION

In IntServ networks reservations can be made with the signaling protocol RSVP [5]. Such reservations are soft state, i.e. they have to be updated regularly otherwise the reservation expires. Other methods to establish reservations are also possible, for instance by SNMP [12]. When SNMP is used the reservation is static, it will only be torn down when it is explicitly instructed to do so.

When RSVP is used the sender sends a PATH message that includes (amongst others) a traffic specification, which consists of five parameters: peak rate (p), token rate (r), bucket depth (b), maximum packet size (M) and the minimum policed unit (m). For voice flows these parameters can be calculated as a function of the bit rate of the codec (R_{cod}), the packetization delay in seconds (T_{pack}) and the header size of a packet in bytes (S_{OH}). It is assumed that the voice source does not use voice activity detection (VAD), and hence, $r=p$, and that the voice source has a fixed bit rate codec and a fixed packetization delay.

In that case, the peak rate (and token rate) can be calculated as follows,

$$p = \frac{R_{cod}}{8} + \frac{S_{OH}}{T_{pack}} \quad \text{[byte/s]} \quad (2)$$

The maximum packet size M is calculated as

$$M = \frac{T_{pack} \cdot R_{cod}}{8} + S_{OH} \quad \text{[byte]} \quad (3)$$

Because the voice source sends packets of a fixed size with fixed inter-packet times (i.e. it is a non-bursty source), the maximum burst size is equal to the maximum packet size M, hence $b=M$. The minimum policed unit m can be chosen arbitrary as long as $m \leq M$. The header size S_{OH} (IP/UDP/RTP) is 40 bytes. The codec bit rates R_{cod} together with the granularity are shown in Table 1 for different types of codecs. The codec granularity is the minimum time between two consecutive voice frames at the output of the coder. The packetization delay must always be a multiple of this.

CODEC	Bitrate [kb/s]	Granularity [ms]
G.711	64	0.125
G.726	16 / 24 / 32 / 40	0.125
G.728	12.8 / 16	0.625
G.729	6.4 / 8 / 11.8	10
G.723.1	5.3 / 6.3	30
GSM-FR	13	20

Table 1: Bit rate and granularity for different codecs

D. INTEGRATED SERVICES

The Internet Engineering Task Force (IETF) has proposed two architectures to provide QoS for IP networks: Integrated Services (IntServ) and Differentiated Service (DiffServ). In the DiffServ architecture, Per Hop Behaviors (PHB) are defined [8], [9]. Such a PHB defines the treatment to an aggregate of traffic. The Integrated Services [4] architecture is a per flow model. Three service classes exist in IntServ: Guaranteed Service [12], Controlled Load [15] and Best-Effort. In this paper we only consider the Guaranteed Service since it provides quantitative guarantees with respect to maximum queuing delay and bandwidth. We do not consider DiffServ.

The bandwidth R that has to be reserved and the maximum queuing delay $\hat{T}_{queue}$ are related as follows for a certain traffic specification (p, r, b, M, m) [12],

$$\hat{T}_{queue} = \begin{cases} \frac{(b-M)(p-R)}{R(p-r)} + \frac{M}{R} + \frac{C_{tot}}{R} + D_{tot} & r < R < p \\ \frac{M}{R} + \frac{C_{tot}}{R} + D_{tot} & R \geq p \end{cases} \quad (4)$$

In the formula above, the C_{tot} and D_{tot} are referred to as the rate-dependent and rate-independent error-terms. They capture the difference of a real packet-based scheduler from the ideal (fluid flow) General Processor Sharing (GPS) scheduler [7]. Each IntServ node on the path that supports Guaranteed Service must update the C_{tot} and D_{tot} terms.

Rate-based schedulers, as e.g. WFQ, guarantee a certain bandwidth to a flow. In this case, delay and bandwidth are coupled. The virtual serialization effect is captured by the term C_{tot} and it is increased with M at each node. The fact that the scheduler is non-preemptive is captured by the term D_{tot}. At nodes that use WFQ it is increased with MTU / C_{link}, where C_{link} is the capacity of the link at that node and MTU the maximum transmission unit. For other rate-based schedulers the term D_{tot} can be different. In this paper we assume that WFQ scheduling is used at each node.

Schedulers, as for example Earliest Deadline First (EDF) [6], are delay-based schedulers. They have the property that bandwidth and delay is uncoupled. Therefore, the term C_{tot} is not increased in this case. However, the term D_{tot} is increased with the maximum queuing delay (i.e. the deadline) and the maximum service time of another packet.

E. OPTIMAL PACKETIZATION DELAY

In this section it is shown how to calculate the optimal packetization delay, and hence, the optimal packet size. In [1] it was already shown how to calculate this optimum. However, the dejittering delay was not taken into account. Therefore, the results are only valid when a perfect adaptive dejittering mechanism is used. In this paper we assume that static dejittering is used, i.e. the dejittering delay is assumed to be equal to the maximum queuing delay such that no packets arrive too late for play-out.

We divide the mouth-to-ear delay budget $\hat{T}_{m2e}$ into two parts. The first part consists of the packetization, maximum queuing and dejittering delay. All these terms depend on the packet size M. The queuing delay through eq. (4) and the dejittering delay because it is chosen equal to the maximum queuing delay. The packetization delay depends on M through eq. (3). The second part, which we denote as T_{min}, includes the serialization, propagation and all other fixed delays. When calculating the optimal packet size only the first part is considered (which is equal to $\hat{T}_{m2e} - T_{\min}$). This is done, because the delay components in the first part are related to each other when Guaranteed Service is used.

There exists an optimal packet size when Guaranteed Service is used due to a trade-off between bandwidth efficiency and delay. When high efficiency is desired, the packetization delay should be chosen large. However, from eq. (4) it turns out that the maximum queuing delay will increase with the packet size because $C_{tot}=N_{stops} \cdot M$ (N_{stops} is number of hops) in case of WFQ. On the other hand when a small delay is desired, a small packetization delay has to be chosen. This results in small packets with relative large headers (hence low efficiency) but also in a small maximum queuing delay.

To visualize this trade-off and to show the existence of an optimal packetization delay the bit rate R that has to be reserved is shown in Figure 2. As an example, this figure was made for a scenario with 10 hops, a codec of 32 kb/s and a $\hat{T}_{m2e} - T_{\min}$ of 100ms. The curve 'codec+header' shows the bandwidth of the voice flow including the header overhead as a function of the packetization delay (see eq. (2)). The curve denoted as R_{min} is the minimum amount of bandwidth that has to be reserved to obtain a maximum queuing delay regardless of the peak rate of the flow (see eq. 4) such that $T_{pack} + \hat{T}_{queue} + T_{dejitter} = \hat{T}_{m2e} - T_{\min}$. The curve 'R' shows the reservation that has to be made in order to reserve enough capacity while adhering to the delay constraint. The point in the figure where the bandwidth that has to be reserved in the network is minimal is the point of the optimal packetization delay. This optimum can be calculated with eq. (5) when WFQ schedulers are used in each node on the path. The optimal packet size can then be calculated with eq. (3). In this optimum the reserved bandwidth R is equal to the peak rate p. Note that

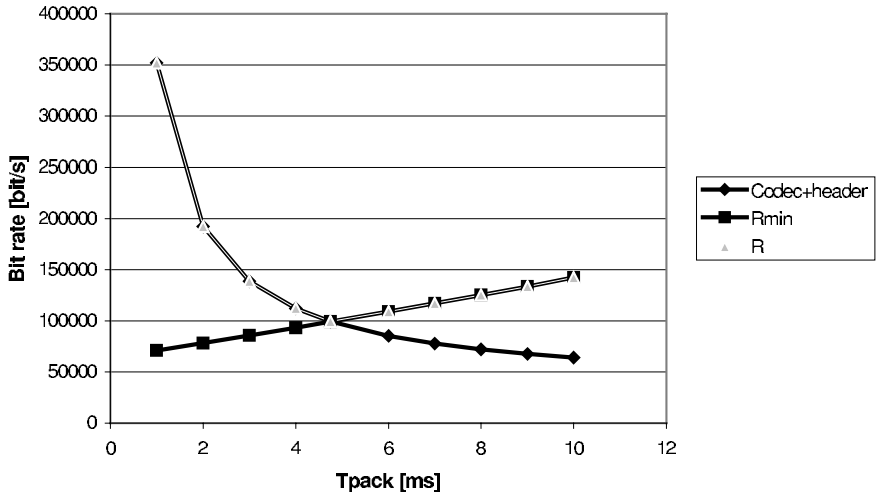


Figure 2: Different bit rates as a function of the packetization delay

the peak rate depends on the packetization delay due to the header overhead.

The optimal packetization delay is given by [2]

$$T_{pack}^{opt} = \frac{\hat{T}_{m2e} - T_{\min} - 2\sum_{i=1}^{N_{stag}} \frac{MTU^i}{C_{link}^i}}{1 + 2N_{stag}} \qquad (5)$$

with:

N_{stag}	Number of hops
MTU^i	Maximum packet size at node i
C_{link}^i	Link capacity at node i
$\hat{T}_{m2e}$	Mouth-to-ear delay budget
T_{min}	Serialization, propagation and other delays

To be able to determine this optimal packetization delay the sender should know the values of MTU^i, N_{stag}, T_{min} and C_{link}^i. IntServ capable routers support a set of general control and characterization parameters [13]. These parameters can be obtained by using for instance RSVP. In Table 2 the relevant parameters are listed and explained. These parameters can be used to calculate the optimal packetization delay with eq. (5).

Parameter name	Symbol
NUMBER_OF_IS_HOPS	Nstag
MINIMUM_PATH_LATENCY	Tmin
PATH_MTU	min. MTU on the path
AVAILABLE_PATH_BANDWIDTH	min. Clink on the path

Table 2: General parameters supported by IntServ

F. EFFICIENCY OF GUARANTEED SERVICE

In this section we define the efficiency as the ratio between the codec bit rate R_{cod} and the bit rate R reserved in the network. The efficiency of the guaranteed service is studied for two scenarios. The first scenario is the IP-phone-to-IP-phone scenario, which is depicted in Figure 3. Such a scenario can occur for instance in a corporate network.

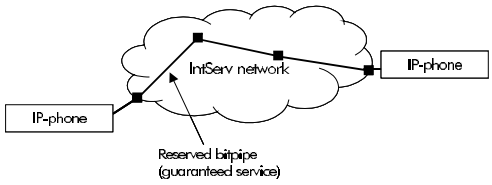


Figure 3: IP-phone-to-IP-phone scenario

Another scenario is an IntServ network where a telecom operator has deployed a number of VoIP gateways. Between all gateways a bit pipe is reserved and all the calls from one gateway to another are aggregated. When the bit pipe is in danger of getting saturated it is possible to either block new calls or to increase the capacity of the bit pipes at the next RSVP reservation update. This scenario is shown in Figure 4.

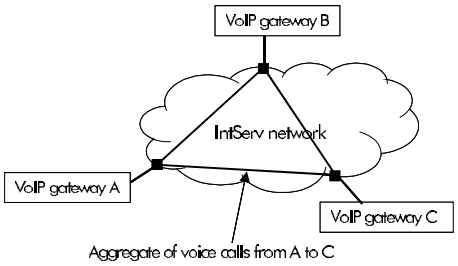


Figure 4: Gateway-to-gateway scenario

1st. IP-phone-to-IP-phone

With eq. (5) the optimal packetization delay can be calculated. In Figure 5 the optimal packetization delay is shown as a function of the number of traversed hops for several delay budgets. The delay budget (shown in the legend) is equal to $\hat{T}_{m2e} - T_{\min}$. Hence, to calculate the mouth-to-ear delay the propagation delay, serialization and other delays (i.e. encoding, etc.) should be added.

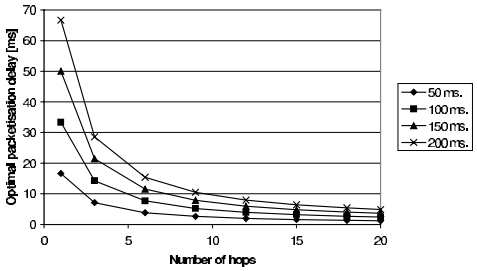


Figure 5: Optimal packetization delays

Figure 5 is only valid for codecs with infinitesimal small granularity. As explained before the packetization delay can only be a multiple of the codec granularity. This phenomenon becomes particularly important when the codec granularity is larger than the optimal packetization delay. In this case, the latter should be chosen equal to the codec granularity. This will result in an overreservation because the packetization delay cannot be chosen optimally. From Figure 5 it can be concluded that the optimal packetization delay is often smaller than the codec granularity (see Table 1). When the optimal packetization is in between two multiples of the codec granularity, the packetization delay should be chosen equal to the closest multiple of the codec granularity.

The peak rate of the voice flow can be calculated from the optimal packetization delay with eq. (2). Because the packetization delay is optimally chosen, the peak rate is equal to the bandwidth that has to be reserved. The efficiency as a function of the number of hops is shown in Figure 6 for a delay budget ($\hat{T}_{m2e} - T_{\min}$) of 100 ms. The bit rates shown in the legend are the codec bit rates R_{cod} (i.e. without IP/UDP/RTP overhead).

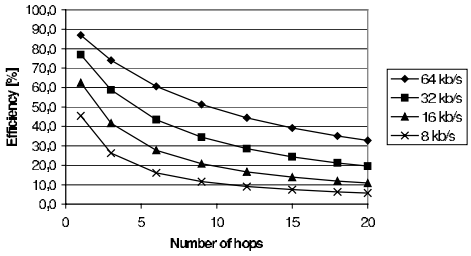


Figure 6: Efficiency for a delay budget of 100 ms

From Figure 6 several remarks can be made. First, a large number of hops on the path results in lower efficiency. This is because the maximum queuing delay accumulates at each hop. Second, the larger the bit rate of the codec, the better the efficiency (but the amount of bandwidth that has to be reserved increases with the bit rate of the codec of course). In other words, the smaller the bit rate of the codec the more dominant the header overhead becomes. Another observation can be made when the efficiency figures are made for different delay budgets (not shown in this paper). The tighter the delay budget the worse the efficiency is. This is because tighter delay budgets result in smaller packetization delays, and hence, smaller packets. Relative small packets result in more overhead due to header bytes.

The efficiency in the case above is quite low. This can be improved by allowing an overreservation in order to reduce the maximum queuing delay. In this case an assumption has to be made about the percentage of voice traffic on the network.

We denote the target efficiency as ε (in eq. (5) ε=1 was used). Eq. (5) now becomes [1]

$$T_{pack}^{opt} = \frac{\hat{T}_{m2e} - T_{\min} - 2\sum_{i=1}^{N_{stag}} \frac{MTU_{\max}^{i}}{C_{link}^{i}}}{1 + 2N_{stag}\varepsilon} \qquad (6)$$

The target efficiency ε has to be chosen equal to the target percentage of voice traffic on the network. The reserved bandwidth that is not used can be consumed by best-effort traffic. In Figure 7 the optimal packetization delays are shown when the voice traffic is targeted at 10% of the network capacity, hence R=10·p. From this figure it can be concluded

that by allowing an overreservation the efficiency will improve. This is due to the fact that the queuing delay is decreased by making an overreservation, leaving more delay budget for packetization. Larger packetization delays result in packets with relatively less overhead.

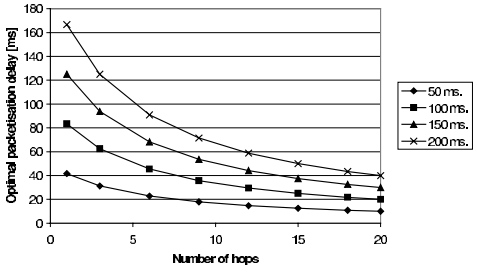


Figure 7: Optimal packetization delay in case of 10% voice load on the network

2nd. Gateway-to-gateway

This section discusses the gateway-to-gateway scenario. In this case several voice flows are aggregated into a single bit pipe. This pipe can be considered as a virtual leased line that is provided by the guaranteed service. When the number of calls is increasing, the gateway can either block new calls or increase the reservation at the next reservation update.

Due to aggregation the processing required for the set-up of reservations at each router will decrease since only one (large) reservation is made instead of a reservation for each individual flow. Also the amount of state information will reduce. Two methods for multiplexing voice flows on one bit pipe are discussed. First, the multiplexing of packets from different flows is discussed, and second, the multiplexing of voice frames into a single packet. Finally, both methods are compared.

1) Multiplexing IP packets

The first approach to multiplex multiple voice calls over one reserved bit pipe is to multiplex the packet flows of the different voice sources. This is shown in Figure 8.

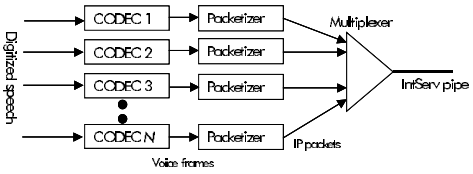


Figure 8: Multiplexing of different packet flows into one bit pipe

In this case it should be taken care of that the multiplexer does not introduce (too much) packet loss and delay. Packet loss and delay due to multiplexing can be avoided completely when the packetizers are clocked in such a way that the packets are produced one after the other. This way no queuing is needed in the multiplexer and no packet loss occurs. This timing is illustrated in Figure 9 when all voice sources use the same type of codec and packetization delay.

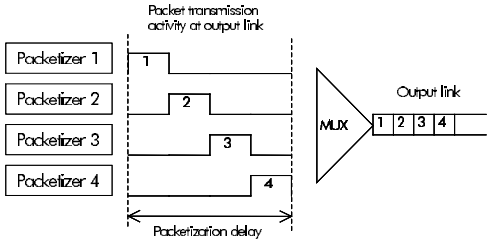


Figure 9: Timing of packetizers

To determine the optimal packetization delay (and hence, the packet size) for an aggregate of identical packet flows eq. (5) has to be modified slightly. Denote N as the maximum number of multiplexed voice flows. Note that taking N as the current number of flows would be more efficient. Notice that this will result in changing the packetization delay of each flow at the set up or release of a flow. Now, the optimal packetization delay is given by

$$T_{pack}^{opt} = \frac{\hat{T}_{m2e} - T_{\min} - T_{MUX} - 2\sum_{i=1}^{N_{stag}} \frac{MTU^i}{C_{link}^i}}{1 + (2N_{stag}\varepsilon / N)} \qquad (7)$$

When the packets are generated as shown in Figure 9, the T_{MUX} (multiplexing delay) term will be zero.

With this multiplexing approach each call uses a different destination port number. Hence, the port number at the receiver is used to associate the packet flow with a certain voice call.

2) Multiplexing voice frames

Another multiplexing method is to multiplex voice frames from different sources into a single IP packet. This is shown in Figure 10.

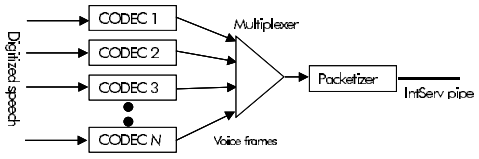


Figure 10: Multiplexing of voice frames into a single packet

For this approach eq. (5) can still be used. The different codecs are modeled as a single source with a rate of $N \cdot R_{cod.}$ The packet bit rate is equal to the codec bit rate plus multiplexing overhead times the number of flows plus the IP/UDP header.

This multiplexing approach needs some additional overhead. This is because the receiving gateway must know which voice frames belong to a certain voice flow/call. Several multiplexing schemes [3], [14] have been proposed. For most schemes an additional overhead of 3 or 4 bytes is introduced per multiplexed flow. However, no standardized solution exists to multiplex RTP flows into one IP/UDP packet yet.

3) Comparison

In this section the efficiency is investigated for the two multiplexing approaches described in the previous two sections. For purposes of comparison it is assumed that ten homogeneous voice flows are multiplexed into one bit pipe. These results are compared with the efficiency of ten separate per-flow reservations.

As an example, we consider the case where no overreservation is allowed, i.e. $\varepsilon = 1$ in eq. (6). The delay budget for packetization, queuing and dejittering is 100 ms. Each voice flow is assumed to use a 32 kb/s codec. Hence, the aggregate flow without overhead has a rate of 320 kb/s. In case of the voice frame multiplexing approach it is assumed that each packet has an IP/UDP header (28 bytes) and 4 bytes of multiplexing overhead per flow. In Figure 11 the efficiency is shown for the two multiplexing approaches and the per flow reservation approach.

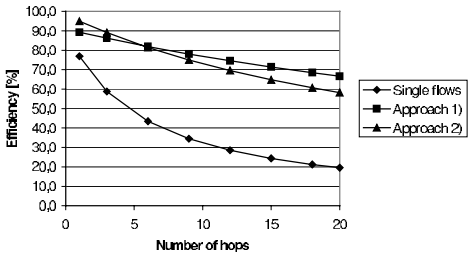


Figure 11: Efficiency for both scenarios

From the results of Figure 11 it is clear that the per-flow reservation is the least efficient and requires the largest bandwidth reservation per voice flow. This is because the maximum queuing delay depends on the packet size and the amount of reserved bandwidth. The bandwidth of a single voice flow is small, and hence, the packet size has to be chosen small in order to meet the maximum queuing delay requirements. When the payload of an IP packet is small, the header overhead is relatively large.

From Figure 11 it turns out that both multiplexing approaches are close with respect to efficiency (and hence, bandwidth reservations). The first approach has IP/UDP/RTP header overhead for each flow. Due to the aggregate reservation the maximum queuing delay of the bit pipe is relatively small such that more delay budget is left for packetization. This results in larger packets, and hence, relatively less overhead. In the second approach all flows are aggregated into a single IP packet. Due to multiplexing schemes some extra per-flow overhead is introduced in each packet for timing and sequencing. However, due to the high aggregate bit rate the packet size becomes large enough to achieve a high efficiency.

In case of approach 1 no multiplexing schemes are needed. The receiver only needs to use the port number to associate the different flows to a voice call. The multiplexing schemes for the second approach are not standardized yet and introduce more complexity.

The optimal packetization delays for the three different approaches are shown in Table 3. Due to the aggregate reservation the maximum queuing delay is small such that a large part of the delay budget is left for packetization in case of approach 1. Because of this large packetization delay the packet size does not become too small (the packet is only filled by a single source). In case of approach 2 the packetization delay is very small but the packet is filled with the aggregate of voice flows.

	Optimal packetization delay [ms]	
Number of hops	Per-flow approach Approach 2	Approach 1
1	33	83
3	14	63
6	8	45
9	5	36
12	4	29
15	3	25
18	3	22
20	2	20

Table 3: Optimal packetization delay

Summarizing, it can be concluded that the per-flow reservations are the least efficient. Therefore, all flows going from one gateway to the other should be aggregated. The efficiency of both multiplexing approaches is close to each other. However, approach 1 has the advantage that individual flows can easily be distinguished by the destination port number as opposed to approach 2 where a multiplexing scheme is needed.

G. CONCLUSIONS

In this paper the Guaranteed Service of IntServ has been studied for two VoIP scenarios: an IP-phone-to-IP-phone and a gateway-to-gateway scenario.

When per-flow reservations are used, the efficiency of small-bandwidth voice flows with stringent delay requirements is low. This results in a network transporting voice traffic that consists for a large percentage of header overhead.

It was also observed that the smaller the bit rate of the codec the more dominant the header overhead becomes. The efficiency can be improved by making an overreservation to achieve a small maximum queuing delay such that the packetization delay can be increased. A larger packetization delay results in larger packets, and hence, relatively less overhead. The overreservation will result in reserved but unused bandwidth. However, this excess bandwidth can be used by best-effort traffic.

Another method to improve the efficiency is to reserve high bit rate IntServ pipes. This is only possible in the gateway-to-gateway scenario. Because of the high bandwidth the maximum queuing delay in the WFQ nodes on the path is small. Therefore, more delay budget is left for packetization. Such a large IntServ pipe can be used to multiplex multiple voice flows. Two multiplexing approaches were evaluated. One approach is to multiplex packets from different voice sources and the second approach is to multiplex voice frames from different sources on a bit pipe. Both multiplexing methods achieve approximately the same efficiency. However, when the first approach is used the voice flows can easily be distinguished using the destination port number. For the second approach a multiplexing scheme is needed. Such a scheme is not yet standardized and it introduces additional complexity. Therefore, multiplexing of packets (approach 1) is preferred.

In our future work we will include DiffServ networks into the calculation. From the viewpoint of an end-to-end IntServ connection a DiffServ cloud is considered as a single IntServ hop that also has to update the C_{tot} and D_{tot} terms appropriately. Hence eq. (5) can be modified to take DiffServ networks into account, if the updating mechanism for C_{tot} and D_{tot} is known.

H. ACKNOWLEDGEMENTS

This work was carried out within the framework of the project LIMSON, sponsored by the Flemish institute for the promotion of scientific and technological research in the industry (IWT).

I. REFERENCES

[1] Mario Baldi, Davide Bergamasco and Fulvio Risso, "On the Efficiency of Packet Telephony", 7th IFIP Int'l Conference Telecommunication Systems, March 1999.

[2] Mario Baldi and Fulvio Risso, "Efficiency of Packet Voice with Deterministic Delay", IEEE Communications Magazine, pp. 170-177, May 2000.

[3] Mooi C. Chuah, Enrique J. Hernandez-Valencia, "A LightWeight IP Encapsulation (LIPE) Scheme", IETF draft <draft-chuah-avt-lipe-00.txt>, June 2000, work in progress.

[4] R. Braden, D. Clark and S. Shenker: IETF Request for Comment 1633, http://www.ietf.org/rfc/rfc1633.txt, "Integrated Services in the Internet Architecture: an Overview", June 1994.

[5] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin: IETF Request for Comment 2205, http://www.ietf.org/rfc/rfc2205.txt, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", September 1997.

[6] Fabio M. Chiussi and Vijay Sivaraman, "Achieving High Utilization in Guaranteed Services Networks Using Early-Deadline-First Scheduling", IWQOS '98, pp. 209-217, Napa, California, May 1998.

[7] R. Guérin and V. Peris, "Quality-of-Service in packet networks: basic mechanisms and directions", Computer Networks 31, pp. 169-189, 1999.

[8] J. Heinanen, F. Baker, W. Weiss and J. Wroclawski: IETF Request for Comment 2597, http://www.ietf.org/rfc/rfc2597.txt, "Assured Forwarding PHB group", June 1999.

[9] V. Jacobson, K. Nichols and K. Poduri: IETF Request for Comment 2598, http://www.ietf.org/rfc/rfc2598.txt, "An Expedited Forwarding PHB", June 1999.

[10] J. Janssen, D. De Vleeschauwer, G.H. Petit, "Delay and Distortion Bounds for Packetized Voice Calls of Traditional PSTN Quality", Proceedings of the 1st IPTEL workshop, GMD report 95, pp. 105-110, Berlin, April 2000.

[11] S.B. Moon, J. Kurose, D. Towsley, "Packet audio play-out adjustment: performance bounds and algorithms", ACM/Springer Multimedia Systems, Vol. 6, pp. 17-28, January 1998.

[12] S. Shenker, C. Partridge and R. Guerin: IETF Request for Comment 2212, http://www.ietf.org/rfc/rfc2212.txt, "Specification of Guaranteed Quality of Service", September 1997.

[13] S. Shenker, J. Wroclawski: IETF Request for Comment 2215, http://www.ietf.org/rfc/rfc2215.txt, "General Characterization Parameters for Integrated Service Network Elements", September 1997.

[14] Keiko Tanigawa, Tohru Hoshi, Koji Tsukada: "Simple RTP Multiplexing Transfer Methods for VoIP", IETF draft <draft-tanigawa-rtp-multiplex-01.txt>, November 1998, (Expired).

[15] J. Wroclawski: IETF Request for Comment 2211, http://www.ietf.org/rfc/rfc2211.txt, "Specification of the Controlled-Load Network Element Service", September 1997.

Dimensioning Links for IP Telephony

Bengt Ahlgren, Anders Andersson, Olof Hagsand and Ian Marsh
SICS
CNA Laboratory
Sweden
{bengta, olof, andersa, ianm}@sics.se

Abstract—
Packet loss is an important parameter for dimensioning network links or traffic classes carrying IP telephony traffic. We present a model based on the Markov modulated Poisson process (MMPP) which calculates packet loss probabilities for a set of superpositioned voice input sources and the specified link properties. We do not introduce another new model to the community, rather try and verify one of the existing models via extensive simulation and a real world implementation. A plethora of excellent research on queuing theory is *still* in the domain of ATM researchers and we attempt to highlight it's validity to the IP Telephony community.

Packet level simulations show very good correspondence with the predictions of the model. Our main contribution is the verification of the MMPP model with measurements in a laboratory environment. The loss rates predicted by the model are in general close to the measured loss rates and the loss rates obtained with simulation. The general conclusion is that the MMPP-based model is a tool well suited for dimensioning links carrying packetized voice in a system with limited buffer space.

Keywords— **Link Dimensioning, Markov Process, IP Telephony, MMPP/D/1/K**

I. Introduction

Voice applications, such as telephony, have been used on the best effort service provided by the Internet for quite some time. Currently many telephone operators have advanced plans to use IP technology as a bearer also for the regular telephone service. This, however, requires that the IP network can provide service guarantees.

Quality of Service (QoS) issues are being addressed by many forums, committees and researchers. Research on IP QoS has concentrated on the issues of classifying, scheduling and admission of packets into a network. Less has been done on how to dimension an IP network carrying real time traffic.

This paper focuses on dimensioning IP network links intended to carry packetized telephony or voice calls. It is feasible that existing carriers would like to allocate a portion of their bandwidth for this service and through mechanisms like differentiated services [12] provide superior service for this kind of data and subsequently levy higher charges.

The objective of this work is 2 fold, firstly to add to the differentiated services model parts which are not addressed in the specifications of the service. The diff-serv model explicitly states well dimensioned links are assumed but it is not addressed in the framework of the group. Therefore it is a work item which needs to investigated. Secondly, we wanted to produce a tool which helps operators dimension links for IP telephony. Telecom operators understand very well how to dimension networks for voice but not over IP networks. Our goal is to assist them, so a typical scenario would be given a set of parameters, how many users should they admit to their new IP telephony service and still deliver them the quality they promised.

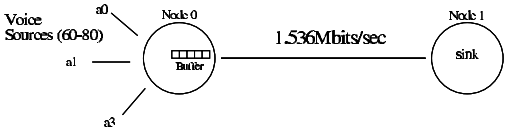


Fig. 1. Problem: dimensioning a link for voice sources over IP.

Our approach is to look at work done in *both* the ATM and traditional telephony communities as well as to use tools and simulators from the IP community to verify these ideas in an environment relevant for the Internet today. We have seen very little work which has taken this approach. The research community is divided into one of the two camps (but is changing as ATM and telephony people are more engaged in Internet research now).

Our assumption in this work is that IP telephony traffic is not mixed with TCP data into the same buffers. This would break the queueing theory irreparably and in a real network, add unmanageable delays and possibly losses to the time sensitive audio data. This is the situation we have today.

Simple mechanisms exist to classify data flows, for example using the source and destination addresses in the IP header and the source and destination ports in the UDP header. This identifies a unique application on 2 hosts in the Internet. Mechanisms also exist to separate data flows into separate queues in a router. A simple scheme is to place a "higher priority" data, such as audio, into a separate buffers and serve this queue before other data. This would be a priority queueing scheme but many others exist to isolate and protect time, or even loss, sensitive data.

Figure 1 illustrates the problem scenario we are addressing. A number of packet voice sources are multiplexed onto a link. The link has a limited amount of buffering which sometimes will result in the loss of packets with the obvious consequences on sound quality. With a link of a given bandwidth and a number of voice sources, what kind of quality could be expected if we ran 60 sources? What if we increased the number to 80 - can we still expect adequate quality? How will we affect the system by changing the amount of buffering in the router?

We present a mathematical model based on a Markov modulated Poisson process (MMPP) which can predict the packet loss probability. We first verify the model using the NS packet level simulator. The main contribution of this paper is the verification of the MMPP model with measurements in a lab network. These experiments show a very good correspondence between the loss rate predicted by the model and the loss rate measured in the lab.

The rest of the paper is organized as follows. After summa-

rizing relevant related work in the next section, we present the MMPP-based mathematical model and the reasoning leading to this model in Section III. Section IV describes the parameters we used in the experiments. Sections V and VI describe the NS simulations and the laboratory experiments, respectively. The experimental results are presented and discussed in Section VII and the paper is concluded with Section VIII.

II. Related work

Link dimensioning for voice has been a research topic for several decades in both academia and the telecommunications industry. Starting a little more than ten years back, the research focus has been on link dimensioning for ATM networks. Most of the results in the domain of ATM networks are also applicable in the domain of IP networks, since both are packet switching systems. The majority of the results from previous research is theoretical or results from simulations. Our research also has results from measurements of a real system.

Several approaches have been suggested in the literature to solve the problem of dimensioning links in packet switched networks. Anick, Mitra and Sondhi [2] study a multiplexer with infinite buffer with a *stochastic fluid flow* model but it is shown by Zheng [16] that this model only works for a multiplexer under heavy load. Tucker [17] studies a multiplexer with finite buffer using the fluid flow model, but it does not fit the model well for small buffers. Heffes and Lucantoni [7] uses a two-state *Markov modulated Poisson process* (MMPP) quite successfully to estimate the delay in a multiplexer with infinite buffer size. They suggest that the same approach for calculating the parameters of the MMPP can be used for a multiplexer with finite buffer size, but Nagarajan, Kurose and Towsley [11] show that this does not work in the case of finite buffer size. Instead, they develop a different method for finding the parameters of the MMPP. Baiocchi *et al.* [4] approximate the arrival process with a two-state MMPP and suggest a method called *asymptotic matching* for the calculation of the parameters of the MMPP. This approach is used by Andersson [1] together with a procedure to calculate the loss probabilities developed by Baiocchi, Melazzi and Roveri [3] to study a multiplexer loaded with a superposition of voice sources.

III. Mathematical preparations and model

In this section we develop a mathematical model for dimensioning a link carrying voice traffic. We begin with some basic material on Poisson and Markov processes so the interested reader can follow the reasoning up to the model we present. In the case of the model itself we start with the arrival process of a single IP telephony source and proceed with the superposition of independent identically distributed sources. The sources are then multiplexed on a bottleneck link through a queue of limited size. A more detailed description of this model can be found in previous work by one of the authors [1]. The model is based on a model developed by Baiocchi, Melazzi and Roveri [3].

A. Markov and Poisson processes

The interested reader should check the references for further details and proofs of the theorems. Resnick's "Adventures in Stochastic Processes" [13] contains a thorough presentation of Markov processes and Allan Gut's "An Intermediate Course in Probability" [6] gives a detailed presentation of Poisson processes.

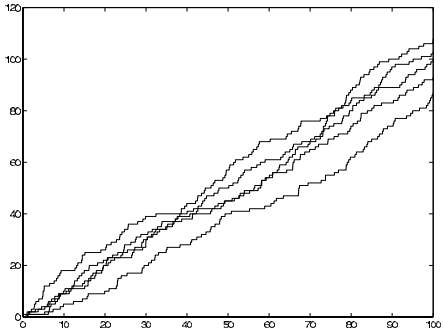


Fig. 2. Five sample paths of a stochastic process

B. Markov Processes

An important class of stochastic processes are Markov processes. This class of processes have some special properties that make them manageable to treat mathematically. A Markov process is governed by the *Markov property* which states that the future behaviour of the process given its path only depends on its present state. Before we proceed into what a Markov process is we need to introduce the concept of *intensity*.

B.1 The Intensity concept

The concept of intensity comes from the question "If we know that an event at time T has not yet occurred, what is the probability that it does in just a moment?". It is possible to summarize this in one formula:

$$P(T \leq t + \Delta t \mid T > t) \tag{1}$$

where Δt is small. By using the definition of conditional probability we obtain:

$$\begin{aligned} P(T \leq t + \Delta t \mid T > t) &= \frac{P(T \leq t + \Delta t, T > t)}{P(T > t)} \\ &= \frac{P(t < T \leq t + \Delta t)}{P(T > t)} \\ &= \frac{F(t + \Delta t) - F(t)}{1 - F(t)}, \end{aligned}$$

where F is T's distribution function. Assuming that F is differentiable, i.e T has density function f, we have

$$F(t + \Delta t) - F(t) = f(t)\Delta t + o(\Delta t).$$

Hence the probability in equation 1 is essentially equal to $\lambda(t) \cdot \Delta t$, where the function

$$\lambda(t) = f(t)/\{1 - F(t)\} \tag{2}$$

is called the *intensity function* for T. Intuitively, if T does not occur at t, is the probability that it is going to occur in the interval $(t, t+\Delta t]$ for small Δt approximately proportional to Δt, and the proportionality constant is the intensity $\lambda(t)$.

B.2 The Markov property

Let $\{X(t), t \geq 0\}$ be a time continuous stochastic process which assumes non-negative integer values. The process is called a discrete Markov process if for every $n \geq 0$, time points $0 \leq t_0 < t_1 < \cdots < t_n < t_{n+1}$ and states $i_0, i_1, ..., i_{n+1}$ it holds that:

$$P(X(t_{n+1}) = i_{n+1} \mid X(t_n) = i_n, X(t_{n-1}) = i_{n-1}, ..., X(t_0) = i_0) = P(X(t_{n+1}) = i_{n+1} \mid X(t_n = i_n)).$$

The definition states that only the present state gives any information of the future behaviour of the process. Knowledge of the history of the process does not add any new information.

In this thesis we only concern ourselves with processes who have time-homogeneous properties. In other words the intensity of leaving a state is constant in time. So it is natural to make the following definition which states that the transition probabilities only depends on which state the process is in and not on the time.

Let $\{X(t), t \geq 0\}$ be a discrete Markov process. If the conditional probabilities $P(X(s+t) = j \mid X(s) = i)$, for $s, t \geq 0$, do not depend on s, the process is said to be time homogeneous. Then we define the transition probability functions $p_{ij}(t) = P(X(t) = j \mid X(0) = i)$ and the transition matrix $\mathbf{P(t)}$, whose element with index (i, j) is $p_{ij}(t)$. Note that $p_{ii}(0) = 1$ and $p_{ij}(0) = 0$ for $i \neq j$, so that $\mathbf{P(0)} = \mathbf{I}$. In many cases it is necessary to study the time between occurrences and therefore we can make the following definition. Let $X(t), t \geq 0$ be a Markov process. We call the times $0 \leq T_1 < T_2 < T_3 < ...$ such that at T_i the process makes a transition from one state to another the occurrence times. We also introduce the duration $Y_n = T_n - T_{n-1}$, where $T_0 = 0$. Figure 3 illustrates the durations of a stochastic process. At times $T_1, T_2, ...$ are there state transitions and the durations $Y_1, Y_2, ...$ are the time spent in a particular state before the next state transition.

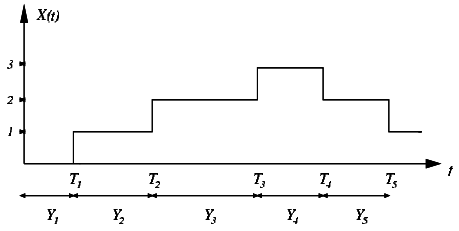


Fig. 3. The duration time

B.3 The Intensity matrix

It is desirable to characterise the behaviour of a Markov process in one matrix. It turns out that the derivative of $\mathbf{P(t)}$ yields such an form.

Let $q_{ij} = p'_{ij}$, for $i \neq j$ we call q_{ij} the transition intensity from state i to state j. Let $X(t), t \geq 0$ be a discrete Markov process. Assume that $q_{ij} = p'_{ij}(0) \geq 0$ for $i \neq j$ and $q_{ii} \leq 0$ such that

$$P(X(t+h) = j \mid X(t) = i) = q_{ij}h + o(h)$$

and

$$P(X(t+h) \neq i \mid X(t) = i) = -q_{ii}h + o(h) = q_i h + o(h),$$

where $q_i = -q_{ii}$, and

$$q_i = \sum_{j \neq i} q_{ij}.$$

The indices i and j shall run through the whole state space of the process. We will name $q_{ij}, i \neq j$, the transition intensity from state i to state j. And we define the intensity matrix $\mathbf{Q}$, whose elements with index (i, j) are q_{ij}. Every Markov process in this thesis will meet the assumption made in the definition above. Another name for the intensity matrix is the *generator matrix*. The summation condition $q_i = \sum_{i \neq j} q_{ij}$ is quite natural since q_{ij} is the transition intensity from state i to state j, and if these intensities are summed over $i \neq j$ we should receive the total intensity out of state i, which is given by q_i. This also means that the rows of $\mathbf{Q}$ sum up to zero.

B.4 Birth-Death processes

A useful class of Markov processes when analysing queueing systems are *birth-death* processes. The only possible state transitions in this kind of processes are from i to $i-1$ or from i to $i+1$. The transition intensity from state i to $i+1$ is designated $\lambda_i \geq 0$ for $i \geq 0$ and the transition intensity from state i to state $i-1$ is designated $\mu_i \geq 0$ for $i \geq 1$.

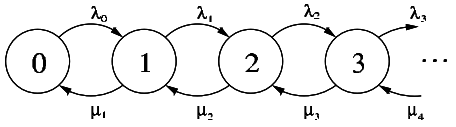


Fig. 4. Model graph for a Birth-death process

The state space of the birth-death process is $\{0, 1, 2, 3, ...\}$. The intensity matrix will be of tri-diagonal type since there are only two ways of leaving a state. Hence, we have the intensity matrix

$$\mathbf{Q} = \begin{pmatrix} -\lambda_0 & \lambda_0 & 0 & 0 & 0 & \dots \\ \mu_1 & -(\lambda_1 + \mu_1) & \lambda_1 & 0 & 0 & \dots \\ 0 & \mu_2 & -(\lambda_2 + \mu_2) & \lambda_2 & 0 & \dots \\ \vdots & \vdots & & & & \ddots \end{pmatrix}$$

As mentioned earlier, certain types of queueing systems are suitably modelled by birth-death processes. The numbers $\{\lambda_i\}$ and $\{\mu_i\}$ are interpreted as the arrival rate of the queue and service rate of the server, respectively.

C. The Poisson process

This section gives a short introduction to Poisson processes. A more complete description of this type of processes can be found in [6]. Poisson processes are one of the most important classes of stochastic processes, and find applications in diverse areas of science such as physics, teletraffic modelling and biology. We introduce a counter which count the number of occurrences from a starting point, and set

$$X(t) = number\ of\ occurrences\ in\ the\ interval\ (0,t]$$

Thus $X(t)$ will increase by one for every occurrence. These increases of one will be called *increments* in the sequel. In many applications is it realistic to assume that occurrences in disjunct intervals are independent of each other, i.e the process $X(t), t \in T$ is said to have independent increments. If the distribution of the increments does not change in time, the process $X(t)$ is said to have stationary increments. If the number of occurrences after time t follows the probability function

$$p_{X(t)}(x) = P(X(t) = x) = e^{-\lambda t}\frac{(\lambda t)^x}{x!}\ for\ x = 0, 1, 2, ...$$

where λ is the intensity of the occurrences, we say that $X(t)$ is a Poisson process. Let us now summarise these requirements in a definition. A Poisson process with intensity $\lambda > 0$ is a stochastic process $X(t), t \geq 0$ such that

(i) $X(t)$ *is integervalued, increasing and* $X(0) = 0$,
(ii) $X(t)$ *has independent and stationary increments*,
(iii) $X(t) \in Po(\lambda t)$.

This is one of many possible definitions of the Poisson process. In [6] there are three different definitions given, all of them equivalent, however proofs of some properties may be considerably simplified with the right choice of definition. In the next section are some important properties of Poisson processes stated.

C.1 Properties

Since $X(t) \in Po(\lambda t)$ we know that

$$E[X(t)] = \lambda t\ and\ Var[X(t)] = \lambda t$$

It can easily be shown that if $X(t)$ is Poisson distributed then the increments from s to a later point $s + t$ is also Poisson distributed. We conclude this in a theorem.

Theorem 1: Let $X(t)$ be a Poisson process and $s, t \geq 0$, then the following is true

$$X(s+t) - X(t) \in Po(\lambda t) \quad (3)$$

The theorem states that if you move the starting time to s and observe what occurs after s you simply get a new Poisson process. It can be shown that the Poisson process can only increase one unit at a time. The next theorem states the distribution of the duration Y_n defined as in definition III-B.2.

Theorem 2: Let $X(t), t \geq 0$ be a Poisson process with intensity λ and let $Y_1, Y_2, Y_3, ...$ be the durations. Then $Y_n \in Exp(\frac{1}{\lambda})$ for $n = 1, 2, 3, ...$

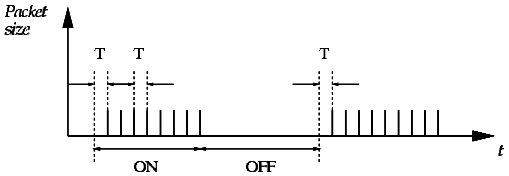


Fig. 5. Characteristics of a single source.

It can also be shown that the Y_n:s are independent of each other. Another important property of Poisson processes is the so called *lack of memory* property.

Theorem 3: If $T \in Exp(\frac{1}{\lambda})$ then we have

$$P(T > t + s \mid T > s) = e^{-\lambda t} = P(T > t) \quad (4)$$

The theorem states that if at time s, we know that there has been no occurrence, then the residual waiting time until an occurrence is $Exp(\frac{1}{\lambda})$-distributed, i.e. the residual time has the same distribution and expectation value as T itself. This is the reason why it is sometimes said that the exponential distribution is memoryless, the lack of memory property.

D. Single source properties

Most standard voice encodings have a fixed bit rate and a fixed packetisation delay. They are thus producing a stream of fixed size packets. This packet stream is however only produced during talk-spurts—the voice coder sends no packets during silence periods.

The behaviour of a single source is easily modelled by a simple on-off model (Figure 5). During talk-spurts (ON-periods), the model produces a stream of fixed size packets with fixed inter-arrival times T. Note that the first packet is produced one packet time after the start of an on-period. This is the result of the packetisation—the voice coder has to collect voice samples before it can produce the first packet.

The number of packets in a talk-spurt, denoted with the stochastic variable N_b, is assumed to be geometrically distributed on the positive integers with mean n. This means that we can never have zero packets in a talk-spurt. This variant of the geometric distribution is sometimes called *first success* distribution (see for instance Gut [6, page 258]), and has the probability function:

$$P(N_b = k) = qp^{k-1}, k = 1, 2, 3, \ldots \quad (5)$$

where q represents the probability that a packet is the last one in a talk-spurt. This means that $p = \frac{n-1}{n}$. This fact implies that the ON-periods have a expected value of $\alpha = nT$, where n is the expected value of the number of packets in a talk-spurt.

We assume that the OFF-periods are exponentially distributed with mean β, which is well documented and discussed by Sriram and Whitt [15]. A voice source may be viewed as a two state birth-death process with birth rate β and death rate α. The OFF state represents the idle periods and the ON state represents the talk-spurts. While in a talk-spurt, packets are generated with a rate of $\frac{1}{T}$ packets per second.

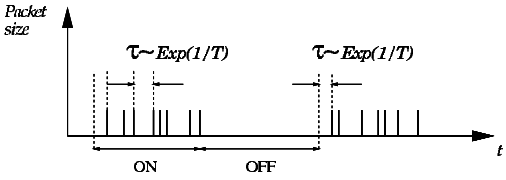


Fig. 6. Single source approximated with exponentially distributed inter-arrivals.

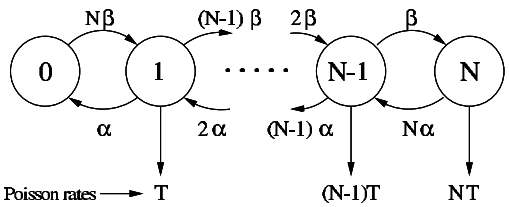


Fig. 7. Superposition of N voice sources with exponentially distributed inter-arrivals.

E. Approximating the single source

We have chosen to approximate the above model using exponentially distributed inter-arrival times with mean T instead of fixed inter-arrival times. The purpose of the approximation is to simplify the modelling of many sources.

We let $\tau \in Exp(\frac{1}{T})$ denote the stochastic variable which describes the inter-arrivals during talk-spurts, and N_b be the geometrically distributed stochastic variable with the probability function stated in Equation 5 with mean n describing the number of packets in a talk-spurt. Moreover τ and N_b are assumed to be independent. It can be easily seen that the ON-periods (denoted U) are exponentially distributed and that the mean length of a talk-spurt is the same as in the deterministic inter-arrival case (nT). Figure 6

illustrates the behaviour of a single source with exponentially distributed inter-arrivals.

As in the previous section the OFF-periods are assumed to be exponentially distributed with mean β. Because of the exponentially distributed inter-arrival times during a talk-spurt, the emission of packets during an ON-period can be regarded as a Poisson process with intensity T. We can use the two state birth-death process to describe the packet generation with one state representing the idle periods and the other state representing the talk-spurts where packets are generated as a Poisson process with intensity T.

F. The superposition of independent voice sources

The superposition of voice sources can be viewed as a birth-death process where the states represent the number of sources that are currently in the ON-state. Here state i represents that i sources are active in a talk-spurt. We refer to the birth-death process as the *phase process* $J(t)$. The birth rate is given by the mean of the exponentially distributed idle periods, and we denote the mean as $\frac{1}{\beta}$. The death rate is determined by the mean of duration of the talk-spurts and is denoted $\frac{1}{\alpha}$. The probability p_{on} that a source is on is given by:

$$p_{\text{on}} = \frac{\alpha}{\alpha+\beta}.$$

G. Markov modulated Poisson process

The *Markov modulated Poisson process* (MMPP) is a widely used tool for analysis of tele-traffic models (see, e.g., Heffes and Lucantoni [7]). It describes the superposition of sources of the type described in Section III-E. When the phase process is in state i, i sources are on. The model graph of the MMPP is shown in Figure 7.

The superposition of Poisson processes is also a Poisson process. We can therefore simply add the intensities of the sources that are currently in a talk-spurt and receive a new Poisson process for the superposition.

To validate the accuracy of approximating with a MMPP process, we calculated the index of dispersion of intervals (IDI) using a formula from Sriram and Whitt [15]. The IDI, also called the squared coefficient of variation, gives us some measure of how similar the traffic is in terms of burstiness. A value of 1 shows the traffic is as bursty as Poisson traffic, whereas a value as 18 is the burstiness of a single voice source. The high value accounts for the fact that the source is indeed bursty. The time period under which one observes this behaviour is very important.

Figure 8 shows c^2_{kN}, the IDI, versus k for k between 1 and 10000 and the number of sources, N, equal to 1, 10, 60 and 130. As a reference we have added the value of c^2_{kN} for a Poisson process. Data was obtained from simulations using a Matlab program. The solid line shows the c^2_{kN} for sources with deterministic inter-arrival times between packets during a talk-spurt, and the dashed lines show the c^2_{kN} for sources with exponentially distributed inter-arrival times, i.e., the MMPP approximation.

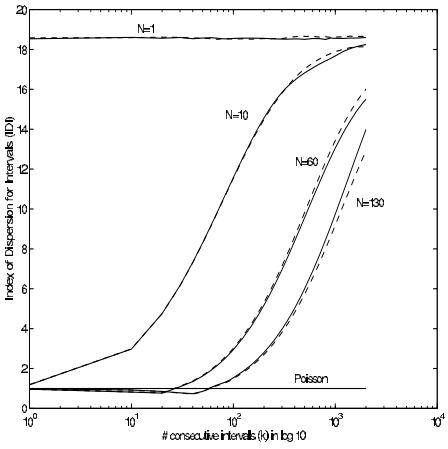


Fig. 8. k-interval squared coefficient of variation curves for superposition of N voice sources.

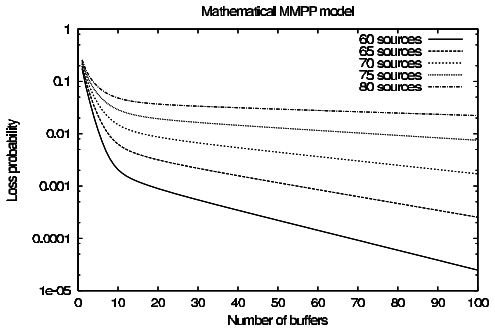


Fig. 9. Loss probabilities computed with the MMPP model.

We see in the figure that the two descriptions of a single source behave in a similar way when they are superpositioned. The figure also shows that the superpositioned arrival process behaves as a Poisson process if we look at it for a short instant of time but it is much burstier if we study it over a longer period of time.

H. The multiplexer: MMPP/D/1/K queue

The arrival process described by the MMPP model is fed into a simple D/1/K queue. It is deterministic, has a single FIFO server and a buffer size (waiting room) which we vary. This kind of model is described in detail by Baiocchi *et al.* [3], [4]. We use their method and formulas for calculating the loss probability.

IV. Parameter values

We used the following parameters to run the MMPP model, simulations and lab experiments:

- 32 kb/s ADPCM voice encoding with 16 ms packet inter-arrival time, which results in 64 bytes of voice payload per packet
- A protocol header overhead consisting of 12 bytes for RTP, 8 bytes UDP and 20 bytes IP. We do not include any link layer headers. The resulting total packet size is 104 bytes, and the resulting bit rate is 52 kb/s.
- The number of successive packets in one talk-spurt is geometrically distributed on the positive integers with a mean of 22, which results in a mean talk-spurt length of 352 ms. The idle time between two successive bursts is exponentially distributed with a mean of 650 ms. The resulting average fraction of time a source is in a talk-spurt is 0.351.
- The bottleneck is a T1 link with a bandwidth of 1.536 Mb/s.

These values coincide with Sriram and Whitt [15] as well as previous work done by Zheng [16] whilst at SICS and Andersson [1], except that we in this paper include protocol header overhead for the RTP/UDP/IP protocol stack.

Figure 9 shows loss curves computed with the MMPP model for a sample set of buffer sizes. The next steps are to compare these loss probabilities from the model with results from NS simulations and measurements from a lab network.

Sources (N)	Load (λ)
29	34.5 %
60	71.4 %
80	95.3 %
84	98 %

TABLE I
NETWORK LOAD FOR A NUMBER OF SOURCES.

```
set cbr($i) [new Agent/CBR/UDP]

set exp($i) [new Traffic/Expoo]
$exp($i) set packet-size 104
$exp($i) set burst-time 0.352s
$exp($i) set idle-time 0.65s
$exp($i) set rate 52K

$cbr($i) attach-traffic $exp($i)
```

Fig. 10. Tcl code fragment defining a source NS-2.

A. Load

We use between 60 and 80 sources to load the link. To define a load that is independent of the link bandwidth the load factor, or λ, is used in the literature:

$$Load\,(\lambda) = \frac{N \times P_{\text{on}} \times Rate_{\text{peak}}}{C}$$

where N is number of sources, C is the link capacity, P_{on} is the probability that the source is on and $Rate_{\text{peak}}$ speaks for itself. Table I shows loads for different numbers of sources.

We decided to run between 60 and 80 sources as 84 sources is where the mean bandwidth of the sources equals to the bandwidth of the link. The peak allocation is as low as 29 sources (100 % utilisation when $P_{\text{on}} = 1$) so taking advantage of the probability that a source is off yields much higher link utilisation.

B. Buffer size

We have chosen to simulate a multiplexer with an output link capacity of 1.536 Mb/s and buffer sizes ranging from 2 to 100 packets. With this choice of parameters we introduce a maximum queueing delay of 54 ms in the buffer. According to ITU recommendation G.114 [8] a delay of 0-150 ms acceptable for telephony, between 150 and 400 ms can also be acceptable, but over 400 ms is not. The total acceptable delay must be divided into a delay budget for each node in the path between the sender and receiver. If the path has 15 hops, and half of the delay budget can be allocated to queueing delay, then we get 13.3 ms per hop. This translates to approximately 24 buffers per hop. For higher bandwidth links, the queueing delay per buffered packet decreases inversely proportional to the bandwidth.

V. NS simulation

We used ns-2 [5], a packet level simulator to verify the MMPP model. Figure 1 shows the topology used in the simulations and Figure 10 the Tcl code that is used to start "agents". They are constant rate sources, denoted by "CBR/UDP". Traffic/Expoo

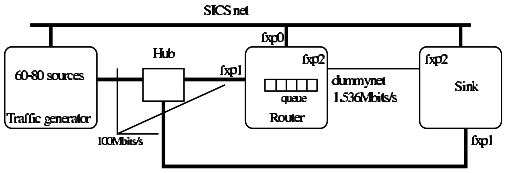


Fig. 11. Topology for Laboratory. The outgoing interface of the router is also connected to the sink.

```
#define INVERSE_M ((double)   4.6566128200e-10) /* lit-
tle number */

int calc_length(double burstlen) {
  double rand, logvalue;

  rand = INVERSE_M * random();
  logvalue = burstlen * -log(rand);

  return ((int)(logvalue + 0.5));
}
```

Fig. 12. C code to "randomize" a burst length

generates traffic based on an exponential on/off distribution with the parameters specified in the next four lines. Each CBR source `$i$` uses a different random number seed, hence the sources will start independently of each other.

The simulation should run long enough for the system to reach steady state, ideally the system should be run for an infinite amount of time, however this is not practical due to time and resource constraints. A reasonable tradeoff is to use a simulated time of 1000 seconds in both the simulation and the lab experiments. 1000 seconds with an interval of 16 ms generates 22000 packets per source and 1.32 million packets for 60 sources or 1.76 million for 80 sources.

VI. Lab network measurements

A. Topology

Figure 11 shows the experimental setup. A single machine acts as a traffic generator and emulates several IP Telephony 'calls' multiplexed together. The traffic is then sent on a shared 100 Mb/s Ethernet and received by two hosts: (1) a machine configured as a router; (2) a sink machine for measurement purposes. An outgoing link of the router is connected to the sink. In this configuration the traffic is emitted by the generator, passes through the router and is received by the sink. Since the sink can observe the packets before it enters the router, it can directly compare latency and loss of each individual packet. The outgoing link of the router is constrained to 1.536 Mb/s using Dummynet [14] which is explained in the next section. All the machines in the experiment were running FreeBSD 3.4.

B. Dummynet

Dummynet is a link emulator which allows arbitrary bandwidths and latencies to be specified. It is often used for emulating a slower link than what is physically available. Buffer sizes can be set for a given link and loss rates set to emulate the effect of lossy links. It is possible to create the illusion for TCP/UDP and IP that the link is like a WAN rather than a LAN. We are primarily interested in the lower bandwidth and configurable queue sizes. We modified the output functionality slightly to enable simpler calculation of the total number of packets received as well as the drop rate. Recording the total number of packets received gives us an additional check if the traffic generator or any system component lost/dropped packets during the experiment.

The total number of sent packets remained the same for a given source count and can be checked with the output of the traffic generator. It is trivial with a script to divide the loss by the total number of packets to obtain the loss rate.

C. Packet capture

To verify the loss rate we gathered the packets on the sink machine via a program that we developed[1] using the Berkeley Packet Filter [10] . Figure 11 shows that the output of the generator is attached directly to the sink machine as well as the outgoing link of the router. This enables us to capture all the packets and the ones not dropped by the router. A simple difference between the two should verify the loss rate reported by Dummynet. Our bpf program captures packets with a specific destination and port, and prints the time of arrival, RTP `src` and `seq` fields.

D. Traffic generator

The idea of the traffic generator is to create a sequence of packets that resemble many individual IP telephony calls multiplexed together. Furthermore, it should perform this job as accurately as possible with each packet emerging with a given deadline.

D.1 Trace file generation and playback

In order to be able to repeat experiments, we first precalculate the sending times of the packets and generate trace files. These files are then fed into the traffic generator which sends packets according to the trace. The trace files also allow us to test our setup to see if packets were being generated at the right times (such as inter-arrival times and sequence). The files are generated on a per source basis. The average length of a burst is calculated as shown in Equation 6.

$$\textit{burst length} = \text{rand}\left(\frac{P_{\text{on}}}{\textit{interval}}\right) \tag{6}$$

The C-code for the rand function is shown in Figure 12.

Using the logarithm of the random variable generates burst lengths which are exponentially distributed.

The same calculation is applied for the idle (with P_{off}) period. The result is (reading vertically for each source) an exponentially distributed series of ON and OFF sequences with a mean ON of 0.351 seconds, OFF of 0.65 seconds which results in a burst length of 22 packets. An example of a trace file[2] with ten sources is shown in Figure 13.

The file shows for each time step (in this case 16 ms) which of the 10 sources are on or off. In the example, sources 1, 3, 5, 7 and 9 sends packets in the first time step. The traces of

[1] Not tcpdump. We wrote out our own kernel filter to extract the packets we wanted as well as a user space program to output headers from 2 interfaces simultaneously.

[2] Actually it is converted into a binary format for more compact representation

```
          source
        0 1 2 3 4 5 6 7 8 9
time
0       0 1 0 1 0 1 0 1 0 1
1       0 1 1 0 1 1 0 1 1 1
2       1 1 1 1 1 1 0 1 0 1
3       1 0 1 0 1 0 1 0 0 1
4       1 0 1 0 1 0 1 0 0 1
5       0 1 1 0 1 1 0 1 1 0
```

Fig. 13. Traffic generator trace file.

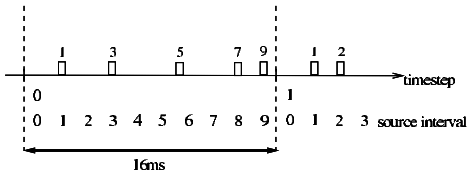


Fig. 14. Traffic generator sending times

one source can be followed by reading a column downwards. Source 2, for example, sends no packet in the first timestep, but then sends a packet in each of the succeeding steps.

If there are n sources, each timestep is further subdivided into n sub steps. Each sub step defines the sending interval for each source. For example, with ten sources and a time step of 16 ms starting at t, source 0 sends its packet within $[t, t+1.6]$; source 1 sends within $[t+1.6, t+3.2]$, etc. If a source does not send its packets within its interval, it is said to miss its deadline. Packets that miss their deadline are recorded by the generator and printed when the run has completed as well as the largest value by which a packet was delayed.

So for the trace file above, the first steps of a packet sequence is shown in Figure 14. The sending of each packet is depicted as a horizontal interval, corresponding to the entering and leaving of the send system call, respectively. In the picture, the packets of source 5 and 7 missed their deadlines. The actual sending time on the link can be measured by an external mechanism, such as the packet capture program described previously.

D.2 Traffic generator verification

As a simple test for a trace file of 220000 packets we obtained values 36.9 % for the on time, 63.1 % for the off time by simply counting the ones and zeros in one column of the file. The mean number of packets in a burst equalled 22.5. Using the trace files turned out to be more useful than we first expected, despite the performance gains of replaying pre-calculated files they also allowed us to test the performance of our traffic generator (setting all the sources on), cross check parameters as just stated as well as generating special sequences for analysing queue behaviour.

D.3 Traffic generator verification

We calculated the index of dispersion of intervals, or IDI (see Section III-G), also for the lab traffic generator. In Figure 15 we can see that the simulation and lab traffic generator produce similar types of traffic. The larger the observation time the more skewed the traffic is. One voice source is equal to about 18.1 also a value is given for a Poisson sources. The graphs show the

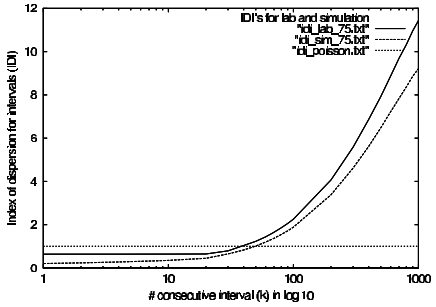


Fig. 15. IDI curves for superposition of 75 sources

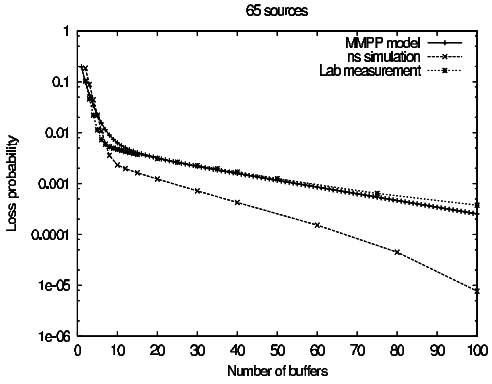


Fig. 16. 65 sources for model, NS and lab (log scale)

result of a trace which was 10000 simulated seconds, resulting in 17.3 million packets for the lab and 16.3 for the simulation.

The traffic generator was also tested to ensure it (and the machine on which we run on) was capable of outputting packets as close to their deadlines as possible

VII. Results

In this section we present and discuss the results from the MMPP model, the NS simulations and the measurement in the lab network. Recall from Section IV that in all three cases we used the 32 kb/s ADPCM voice encoding with 16 ms packetization. This results in 64 bytes of voice payload in each packet and a total packet size of 104 bytes including the RTP, UDP and IP protocol headers.

Figures 16 and 17 show the packet loss probability as a function of the number of buffers on (y) log scales. We can see in these graphs that both the MMPP model results and the NS simulations in general compare well with the measurements in the lab. The exception is for very small buffer sizes and when the loss rate is small.

The MMPP model is most of the time closer to the lab measurements than the NS simulations are, which is an interesting

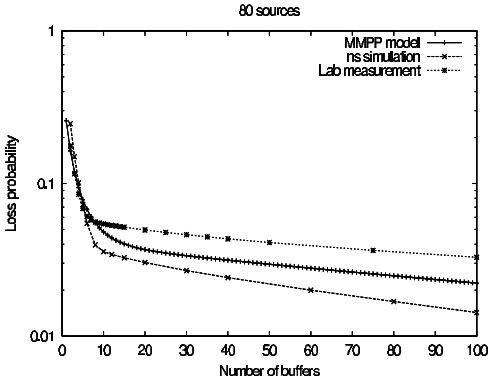


Fig. 17. 80 sources for model, NS and lab (log scale)

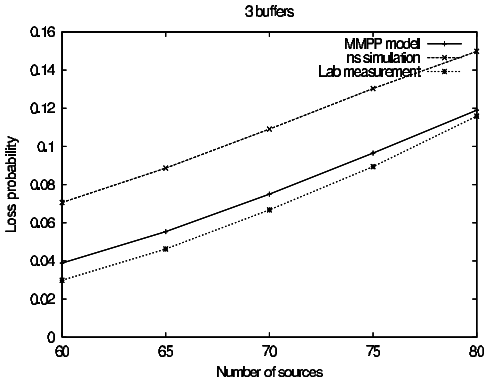


Fig. 18. Loss probability Vs buffers (3)

result. The ns simulations consistently show the lowest loss rates for more than 7–8 buffers. We analysed the output from the traffic generators in NS and in the lab to try to come up with an explanation. We found that there is a small difference in mean total rate between the two that can explain the difference in loss rate.

The second set of graphs presented in Figures 18 to 21 plots the packet loss probability as a function of the number of voice sources for four different buffer lengths measured in packets. These buffer lengths correspond to a maximum queueing delay of 1.6, 2.7, 5.4 and 21.7 ms, respectively. We immediately see that the relationship between the number of sources and loss rate is close to linear for few buffers, but far from linear for many buffers. Visual observation suggests an exponential relationship. In the region above 10 buffers, the lab measurements often has the highest loss rate. Below about 10 buffers, the lab measurements have the lowest loss rate.

One interesting detail is that for very small buffers, the loss curve obtained in the NS simulation is shifted one buffer to the

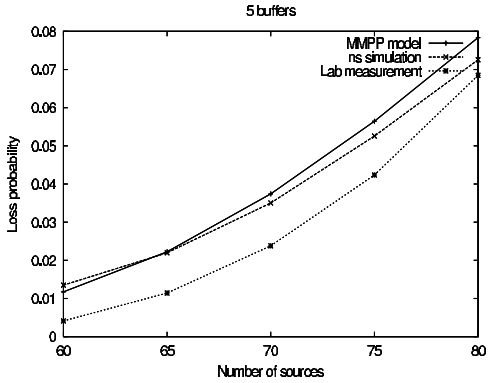


Fig. 19. Loss probability Vs buffers (5)

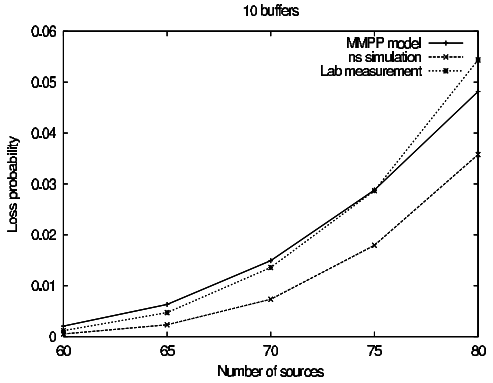


Fig. 20. Loss probability Vs buffers (10)

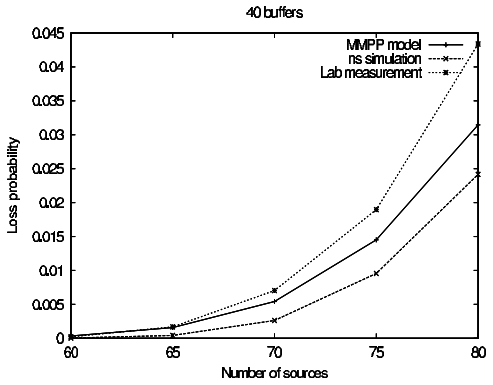


Fig. 21. Loss probability Vs buffers (40)

right in the plots. Even though we have gone to great lengths in ensuring that the three environments have identical properties, there are nevertheless subtle differences that can explain discrepancies like this.

One obvious difference in the models we used is that the bandwidth offered by Dummynet is not exactly the same as in ns. Using netperf we found there to be about a 3% difference between what netperf and dummynet report as their measured and configured bandwidths respectively. Perhaps more subtle and not so obvious is the amount of buffering in the system, in NS we simply state the buffer size in packets (between 2 and 100). In a real system this is much harder to calculate as buffers exist in many places in the system, for example in the queue between the Ethernet driver and `ip_input()` routine on the input side. Ethernet cards can also buffer packets on the output side. This is the default configuration as most Ethernet cards are used on host systems where this is not an issue. Nevertheless, the buffering in a real system is probably larger than the simulation and maybe account for differences in the systems under comparison.

VIII. Conclusions and Future Work

We have studied the packet loss behaviour when a number of homogeneous voice sources are multiplexed onto a bottleneck link. The goal is to find an accurate mathematical model which can be used to dimension the link.

We have implemented a mathematical model based on a Markov modulated Poisson process (MMPP) in Matlab. The model was compared with both simulations using NS and measurements in a lab environment. The comparison shows that the model in general predicts the loss rate well. The exceptions are for small loss rates in some cases. An interesting result is that most of the time the model predicts the loss rate better than the simulations in ns.

This result once more proves that the only way to reliably verify a model is to make measurements of a real system. We found that the relationship between the load and loss rate is close to linear for few buffers (around three), but looks exponential for many (10 and above) buffers.

The general conclusion is that the MMPP-based model is well suited for predicting loss rates for superpositioned voice sources in a system with limited buffer space. The mathematical model is an important tool for conveniently dimensioning network links. The lab environment is constrained to physical limits as well as finite resources where the model is clearly not. Running a lab experiment consumes resources and time a lab experiment takes on average 12 hours to complete. For each number of sources and each buffer size the experiment is re-started which is one reason why we use Dummynet, we can change the buffer sizes *without* re-booting the router. The simulation typically takes 2 hours whereas the model consumes only about 10 minutes as well as considerably less physical resources[3].

The results we have obtained can be applied in several different ways. One scenario could be as stated in the introduction an operator has reserved 1Mbit/sec and would like to promise users a quality no worse than GSM, for example. The operator allows users to download an audio tool with GSM encoding and decides to fix the loss rate of an ingress router at 5%. Figure 17 shows loss probability on the Y axis and buffer size (in packets) on the X axis. From a graph like this we could tell the operator they should have a queue length in the ingress router of at least 6 packets from reading off the graph. In our experiments we used a link of 1.536Mbits/sec and PCM coding but the principle is the same.

Another scenario could be the same conditions but the operator does not know how many customers they should let use the system and still keep under the target of 5% loss and they start complaining!. Figure 18 shows a plot how we could determine this value, which has the number of sources on the X-axis. For 5% loss our simulation gives us 60 sources or users, the mathematical model 64 and laboratory measurements 67 users. Since we have used 3 different techniques and arrived at similar results we could tell the operator they should not allow more than 70 users at this ingress router. In this kind of scenario our work could be used as the input to an admission control algorithm.

It soon becomes clear one could use the results in quite a few different manners. Should a system administrator not know how large to make a high priority queue in a router he or she could consult figures or graphs calculated for the particular network situation they are facing.

There are a number of further work items that we are currently addressing. The maximum delay is bounded by the buffer length in the system studied in this paper, but what is the resulting mean delay? We are also experimenting with higher bandwidth links. One challenge is to accurately generate enough sources. The next step is to measure a system which has multiple traffic classes in the style of diffserv [12]. How does different queue scheduling algorithms affect the dimensioning of traffic classes? Can the MMPP model presented in this paper be used to describe the loss and delay properties of a traffic class?

The ongoing work can be found at a web page[4] which has information about current experiments as well as data which was not directly relevant for this paper.

Acknowledgements

The authors would like to acknowledge the indispensable work of Henrik Abrahamsson in helping us calculating the IDI values presented in this paper. We would like thank Thiemo Voigt for his help in setting up Dummynet and adding extra debug statements to make our loss calculations considerably simpler. Finally we would like to thank Telia AB for their financial support in the early phases of this work.

References

[1] Anders Andersson. Capacity study of statistical multiplexing for ip telephony. Technical Report T2000:03, SICS – Swedish Institute of Computer Science, January 2000.

[2] D. Anick, Debasis Mitra, and M. M. Sondhi. Stochastic theory of a data-handling system with multiple sources. *Bell System Technical Journal*, 61(8):1871–1894, October 1982.

[3] Andrea Baiocchi, Nicola Blefari-Melazzi, and Aldo Roveri. Buffer dimensioning criteria for an ATM multiplexer loaded with homogeneous on-off sources. In J. W. Cohen and Charles D. Pack, editors, *Queueing, Performance and Control in ATM — Proceedings of the Workshop at the 13th International Teletraffic Congress (ITC)*, pages 13–18, Copenhagen, Denmark, June 1991. North-Holland. Volume 15 of the North Holland Studies in Telecommunication.

[3] These values were derived from an Athlon 600 Mhz PC with FreeBSD, a Fast SCSI-3 disk and 128 MB of RAM.

[4] http://www.sics.se/~ianm/Experiment/experiment.html

[4] Andrea Baiocchi, Nicola Blefari Melazzi, Marco Listanti, Aldo Roveri, and Roberto Winkler. Loss performance analysis of an ATM multiplexer loaded with high-speed on-off sources. *IEEE Journal on Selected Areas in Communications*, 9(3):388–393, April 1991.

[5] Kevin Fall and Kannan Varadhan. ns: Notes and documentation. Technical report, Berkeley University, 1998. Technical Report.

[6] Allan Gut. *An Intermediate Course in Probability*. Springer-Verlag, New York, 1995.

[7] Harry Heffes and David M. Lucantoni. A Markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance. *IEEE Journal on Selected Areas in Communications*, SAC-4(6):856–867, September 1986.

[8] International Telecommunication Union (ITU). Transmission systems and media, general recommendation on the transmission quality for an entire international telephone connection; one-way transmission time. Recommendation G.114, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, March 1993.

[9] Samuel Karlin and Howard M. Taylor. *An Introduction To Stochastic Modeling 3rd edition*. Academic Press, London, 1988.

[10] Steven McCanne and Van Jacobson. A BSD packet filter: A new architecture for user-level packet capture. In *Proc. of Usenix Winter Conference*, pages 259–269, San Diego, California, January 1993. Usenix.

[11] Ramesh Nagarajan, James F. Kurose, and Don Towsley. Approximation techniques for computing packet loss in finite-buffered voice multiplexers. *IEEE Journal on Selected Areas in Communications*, 9(3):368–377, April 1991.

[12] Kathie Nichols, Van Jacobson, and Lixia Zhang. A two-bit differentiated services architecture for the internet. Internet draft, Bay Networks, LBNL and UCLA, November 1997.

[13] I. Resnick, Sidney. *Adventures in Stochastic Processes*. Birkhauser, Boston, 1992.

[14] L. Rizzo. Dummynet: A simple approach to the evaluation of network protocols. *Computer Communications Review*, 27(1):31–41, January 1997.

[15] Kotikalapudi Sriram and Ward Whitt. Characterizing superposition arrival processes in packet multiplexers for voice and data. *IEEE Journal on Selected Areas in Communications*, SAC-4(6):833–846, September 1986.

[16] Zheng Sun. Capacity study of statistical multiplexing for ip telephony. Technical Report LiTH-MAT-EX-98-12, Department of Mathematics, Linkoping University, December 1998.

[17] Roger C. F. Tucker. Accurate method for analysis of a packet-speech multiplexer with limited delay. *IEEE Transactions on Communications*, COM-36(4):479–483, April 1988.

Addressing, Accounting and Deployment

IP Telephony Accounting and WAN Deployment Experience

Sven Ubik, CESNET
Prague, Czech Republic
ubik@cesnet.cz

Abstract—**Convergence of data and telephony networks is one of the current trends in telecommunications. It is supposed that all telephony traffic might be carried over a common data network infrastructure in the future.**

In accordance with this development, wide area IP telephony network has been deployed in the TEN-155 CZ network (located in Czech Republic), connected to the European network for research and education TEN-155.

In this article we describe our practical experience with deployment of IP telephony services over a wide area network. We particularly pay attention to the description of IPTA, the IP telephony accounting application, which has been developed as part of our project. IPTA differs from other IP telephony accounting applications by its modular structure, ability of using multiple accounting schemes in paralel and certain other features which we describe in this article.

Keywords—**IP Telephony Accounting, RADIUS, Voice Gateway.**

I. Introduction

IP telephony has been adopted as a way to carry telephone traffic over a common data network infrastructure. However, in order to replace classical telephony network with IP telephony, we also need means for call accounting. In this article we describe our experience with the deployment of IP telephony services over a wide area network and with the development of IP telephony accounting application.

II. IP Telephony in TEN-155 CZ

TEN-155 CZ is a national network for research and education in Czech Republic. It is connected to TEN-155, the European network for research and education. In mid-1999 we established an IP telephony group working in the TEN-155 CZ network. The aim of the group has been set to perform testing, measurement, development and practical deployment of technology for the transmission of voice over a data network. The group is financially and organizationally supported by CESNET, a not-for-profit organization which operates the TEN-155 CZ network.

The goal of the first phase of our project was to interconnect telephone exchanges of several universities across the country using IP telephony and to connect this IP telephony network to the public telephone network. We also wanted to acquire experience with the deployment of IP telephony over a wide area network and to investigate possibilities of integration of IP telephony with other services in the next phase.

III. Selecting technology

One of the requirements was to incorporate existing hardware - telephone exchanges in universities and routers in the TEN-155 CZ network. The telephone exchanges (PBXs) are of different brands and types: Ericsson MD110, Siemens Hicom 300, Alcatel and Matra, and they are equipped with different data interfaces. The network is based mainly on Cisco routers, running ATM and providing both ATM and IP services. We performed a series of experiments with various telephony equipment, particularly testing available PBX interfaces for their suitability for the connection to a data network. Upon completing these tests, we decided to use the VoIP (voice over IP) technology (as oposed to voice over ATM) and digital ISDN PRI/BRI and analog E&M and FXS interfaces.

Next, the choice of signalling had to be made. There are currently two major protocols used for connection management in IP telephony: H.323 [1] and SIP (Session Initiation Protocol) [2] together with SDP (Session Description Protocol) [3]. H.323 is currently prevailing in production implementations of VoIP services. A major advantage of SIP over H.323 is that it is simpler, messages are textual, rather than binary, similar in nature to HTTP. This makes implementation of VoIP software easier and the open architecture of SIP allows for easier integration with other services.

We performed a series of experiments with both H.323 and SIP on Cisco routers, using Microsoft Netmeeting and Cisco IP Phone 30 as H.323 clients and Columbia University sipc and Cisco IP Phone 7960 as SIP clients. We also used sipd (SIP proxy, redirect and registrar server) and SIP/H.323 gateway developed in Columbia University. While all software from Columbia University was working without problems, we ran into difficulties with Cisco IP Phone 7960 (with SIP firmware, release 1.0) and with configuration of SIP on Cisco routers. Using packet analyzer and protocol monitor in sipc we found that Cisco IP Phone 7960 was sending incorrect `From` and `To` headers in INVITE and REGISTER SIP messages. The problem was reported by Cisco as a known bug. Therefore, we decided to use H.323 initially, with hardware architecture allowing to switch to SIP later (mainly choosing routers reported to support both H.323 and SIP).

IV. Pilot project

At the end of the year 1999 the local telephone exchanges of CVUT in Prague and VSB-TU in Ostrava (two universities approximately 400 kilometers appart) were connected by the VoIP technology in a pilot project [4]. We connected universities in two other cities later. For compatibility with existing hardware deployed in our network, we inititally used Cisco AS5300 routers with the E1 ISDN/PRI interfaces to implement VoIP gateways. The configuration is shown in Fig. 1.

We did not have routers to be used as voice gateways in the other two other universitites (connected later, shown below in Fig. 1). Therefore, we took advantage of ATM connectivity between all connected universities. We configured voice gate-

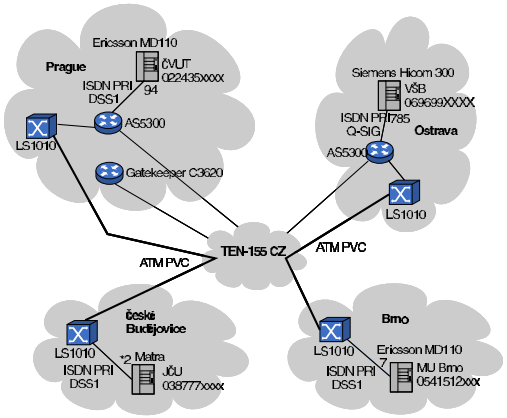


Fig. 1. Configuration of the pilot project

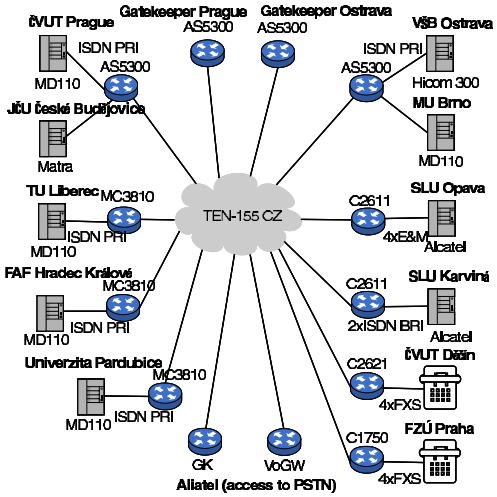


Fig. 2. Current IP telephony infrastructure in TEN-155 CZ

ways at the first two universities as having the PBXs in the latter two universities connected directly to their local ports, tunneling connections from these ports to remote PBXs through ATM PVCs. There is E1 connection between the voice gateway and ATM switch on one side and between the ATM switch and PBX on the other side.

To call from one university to another, users dial a specified prefix to escape into the IP telephony network, followed by the called number in E.164 format. We plan to enable transparent routing of calls through the IP telephony network using LCR (least cost routing) service, implemented in some PBXs. When calling from an extension in one PBX to an extension in another PBX, there is no telephone fee paid by either the caller or callee. As a part of the pilot project, an experimantal VoIP connection to CERN, Switzerland was also implemented and is currently used in both directions.

V. Current infrastructure

We are currently connecting several other universities to our IP telephony network. The infrastructure to be completed by May 2001 is depicted in Fig. 2. Although we were satisfied with Cisco AS5300 routers, we needed a less expensive alternative to build more voice gateways. Based on the requirements on voice interfaces in individual locations, we decided to use Cisco MC3810 or Cisco 2620 routers. As we intend to gradually switch from ATM to PoS on the network backbone, we plan to install separate voice gateways to two localities now connected through ATM PVC tunnels.

Currently we use G.711 codec on all VoIP connections. We also tested G.729 codec to see if people find the quality of sound acceptable. The quality seemed to be reasonable so we consider using G.729 when bandwidth becomes an issue for some localities.

We use two gatekeepers to provide fault tolerance in case one of the gatekeepers fails. This is made possible by setting two gatekeepers with different priorities in configuration of voice gateways. With some newer versions of the Cisco IOS it is possible to implement both a voice gateway and a gatekeeper on the same router. However, it requires so called enterprise feature set of the Cisco which is expensive and thus the collocation of a voice gateway and a gatekeeper on the same box could be more expensive than two separate boxes.

The IP telephony network is connected to the public telephone network and mobile phone network via the telephone exchange of CVUT in Prague. The service is provided by a telecommunication company. We have to pay to this company for the calls made from the telephone exchange of CVUT in Prague. It means that if someone from another city calls a number located in Prague, the call will be carried by the IP telephony network to Prague and it will be charged only as a local call in Prague.

VI. Accounting - a missing part

The telecommunication company currently charges us using a classical scheme, based on the call destination, duration, time of day and day of week. Even in case of another scheme, we would always need to keep track of calls initiated by all users connected to our IP telephony network. Using this information, we can determine the amounts which we will require to be paid by connected universities to settle the bill from the telecommunication company. It would be also useful to keep track about IP-only telephony calls (not continuing to the public telephony network) to check the utilization of VoIP interfaces and find potential problems.

As we did not find any existing solution satisfying all our requirements, we decided to develop a new IP telephony accounting application. We identified the following features which must be supported by the IP telephony accounting application:

- User-defined aggregations — flexible call listings from high-level summaries of aggregated VoIP traffic between connected

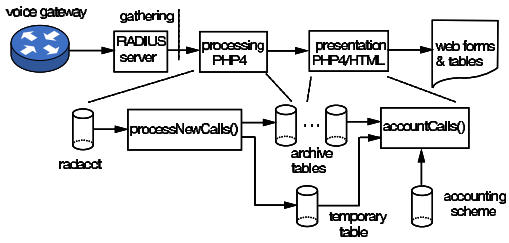


Fig. 3. Structure of the accounting application

organizations to detailed listings of individual calls

- Flexible accounting scheme — initially based on the location of the caller and callee, time of day and day of week, amount of transfered data and other parameters, with the posibility of easy modification
- Possibility of simultaneous accounting using several different accounting schemes - to evaluate alternative accounting approaches and gateway locations
- Reliability and scalability — resillience against changes of configuration of VoIP gateways, ability to store and search in reasonable speed records for calls made over several years

VII. Architecture

The structure of the accounting application is shown in Fig. 3. It consist of three parts - gathering of accounting information from voice gateways, processing of accounting information and presentation of the processed information to the user in a web-based interface. The three parts have defined interfaces between them and it is possible to replace one of the parts without the need to change the other parts. For example, it is possible to gather accounting information using different protocols from different voice gateways.

A. Gathering of accounting information

The routers which we used as voice gateways provide information about the routed calls via RADIUS accounting messages or UNIX syslog protocol. We decided to use RADIUS [5] because it provides accounting information in a more structural format than the syslog protocol, which is more suitable for further processing. RADIUS messages are sent from voice gateways, acting here as Network Access Servers (NAS), to the main and backup RADIUS servers. RADIUS is one of the standardized protocols for sending authorization and accounting information. However, the set of attributes included in the original specification do not cover all information needed in IP telephony accounting. There are two ways how to carry all necessary information.

First, the `AcctSessionId` attribute can be overloaded, carrying several pieces of accounting information in a series of strings separated by slashes [7]. An example `AcctSessionId` attribute value can look like this:

```
427/10:08:23.785 MET Thu Nov 23 2000/
Voice-OV.ten.cz/9662886D A66029F 0
42FD97B4/answer/Telephony/10:08:54.544
MET Thu Nov 23 2000/10:08:57.194 MET
Thu Nov 23 2000/10/195.113.134.240
```

Individual parts delimited by slashes mean: session identifier, call leg setup time, gateway id, connection id, call origin, call type, connect time, disconnect time, disconnect cause and remote IP address. We found that the exact format of the `AcctSessionId` attribute differs slightly in different Cisco IOS versions. However, the key parts are always present and it is possible to make parsing of this attribute resilient to small changes.

Another possibility is to use vendor-specific attributes [8]. Each piece of accounting information is carried in a separate attribute encapsulated in a vendor-specific (type 26) RADIUS attribute. An example set of vendor-specific attributes can look like this:

```
h323-gw-id=AS2-OV.ten.cz
h323-conf-id=
  4776D8CD B9A60177 0 37DBD1 74
h323-incoming-conf-id=
  4776D8CD B9A60177 0 37DBD174
h323-call-origin=answer
h323-call-type=VoIP
h323-setup-time=
  16:17:13.944 MET Mon Jan 22 2001
h323-connect-time=
  16:17:16.256 MET Mon Jan 22 2001
h323-disconnect-time=
  16:17:16.256 MET Mon Jan 22 2001
h323-disconnect-cause=10
h323-voice-quality=0
h323-remote-address=195.113.134.240
```

Some version of Cisco IOS can use only one of these two possibilities of carrying the accounting information, others can use any of them based on configuration. Therefore, the accounting application must accept both formats.

We decided to use ICRADIUS [6] server, because it has built-in support for storing RADIUS messages to a MySQL database. However, we had to modify the server in several ways. First, we added support for vendor-specific attributes, which was not included in the original server. Second, we had to change certain declarations so that the server now accepts long overloaded `AcctSessionId` attributes. The start and stop records (see below) now have to be matched based on the first part of the `AcctSessionId` attribute (the original session ID) rather than on the whole attribute value. Finally, we resolved some bugs in parsing of the configuration file.

The RADIUS server checks for duplicate messages, which can be received when the acknowledgement from the server sent back to the voice gateway gets lost, and it matches start and stop records sent by routers at the beginning and at the end of each call. However, it is sufficient to get only the stop records to obtain all needed accounting information. Therefore, we suggest that the voice gateways be configured to send the stop records only to preserve some bandwidth and processing on the server. The gatekeeper can also be configured to send RADIUS messages. However, although one of the tasks originally conceived to be performed by a gatekeeper was accounting, the RADIUS messages sent by the gatekeeper do not contain all the necessary

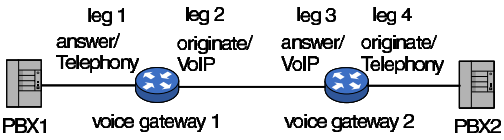


Fig. 4. Legs of a connection through two voice gateways

information about the calls. Information sent by voice gateways is more complete and sufficient, therefore we suggest that gatekeepers be configured not to send RADIUS messages.

Note the string `answer/Telephony` in the `AcctSessionId` value or `h323-call-origin` and `h323-call-type` vendor-specific attributes. These strings identify which call leg this message belongs to. Each call has either two legs, when passing through one voice gateway or four legs, when passing through two voice gateways (which is a typical case of calls from one PBX to another), see Fig. 4. Thus the RADIUS server normally receives either two or four messages for each call.

The parsed messages are stored in `radacct` table in the database, which serves as an interface between gathering and processing parts of the application.

B. Processing of accounting information

All processing and presentation of accounting information is done by PHP4 scripts. We normally use messages provided by the first voice gateway on the call route, namely the `answer/Telephony` leg for accounting of calls originated from a PBX and the `answer/VoIP` leg for accounting of calls originated by direct VoIP users (e.g., using Microsoft Netmeeting). The other two legs (`originate/Telephony` and `originate/VoIP`) are often missing some important information. An exception are calls from CERN, Switzerland where we could not configure the voice gateway (because of administrative reasons) to send RADIUS messages. Therefore, we use the `answer/VoIP` leg records provided by the second voice gateway on the call route to account these calls. However, sometimes the normally used message for a certain call leg is not available (the server did not receive the message) or it is broken in some way. In such cases we try to account the call using another leg by storing a found broken leg in a cache, looking for a correct leg of the same call (which has the same connection ID). This feature significantly improves reliablity of the accounting application. Detailed information about calls that had broken legs and about calls that could not be accounted even with assistance of the cache is logged via syslog enabling analysis of network problems.

Once a day `dailyTask()` script is automatically executed to process records in `radacct` table stored during the last 24 hours. Unfortunately, voice gateways sent messages for all attempts of calls, that is not only for calls that were made through, but also for cases when the called part did not respond. Therefore, the messages corresponding to these not-answered

```
tt CREATE TABLE tCallsDec2000 (
id INT NOT NULL AUTO_INCREMENT,
caller VARCHAR(32) NOT NULL,
callee VARCHAR(32) NOT NULL,
connectTimestamp INT NOT NULL,
duration int NOT NULL,
inputOctets int(12),
outputOctets int(12),
gwIp varchar(15),
remoteIp varchar(15),
callerMatched TINYINT,
calleeMatched TINYINT,
PRIMARY KEY (id),
UNIQUE (caller, callee, connectTimestamp)
);
```

Fig. 5. Structure of the table to store processed calls

calls are identified and deleted from the `radacct` table by `deleteNotAnsweredCalls()` function.

Then, `processNewCalls()` function is called. As a first step, it identifies the caller and callee based on the IP address of the voice gateway the message came from, the identification strings sent by the voice gateway and the rewriting tables in the database. This step is necessary because in some cases the identification of the caller or callee sent by the voice gateway depends on the configuration and capabilities of the PBX and need to be rewritten to some normalized form for further processing. The normalized form we use is `<area code><number>` for users within the country and `<country code><area code><number>` for international users. Some PBXs, for example, Ericsson MD100 do not send the extension number (within an organization) of the calling user to the voice gateway by default. It has to be enabled by issuing a command separately for each extension. It can be automated by a script.

As the next step, the time of the beginning and end of the call is parsed and the duration of the call is computed. Finally, the call is inserted into one of the archive tables used to store processed calls. There is one such table for each month. This storage configuration scales well in terms of the number of records in a table and the number of tables in a database and files in the operating system. It also allows for easy backup of old tables by copying the corresponding files at the filesystem level. Definition of one of these tables is shown as an example in Fig. 5.

The cost of individual calls is not stored in the database. Only accounting schemes are stored in the database. The cost of calls is computed dynamically at the presentation time (see below). The cost can be based, for example, on the location of the callee, the duration of the call, time of day and day of week.

C. Presentation of accounting information

Accounting information is presented to the user in a web interface created dynamically also by PHP4 scripts. The user can browse through summaries at various levels of aggregation on either the caller or callee side, directly specify the caller and callee and the beginning and end of the accounting period. For example, it is possible to select a summary of all calls from CVUT (university in Prague) to numbers beginning 060, which

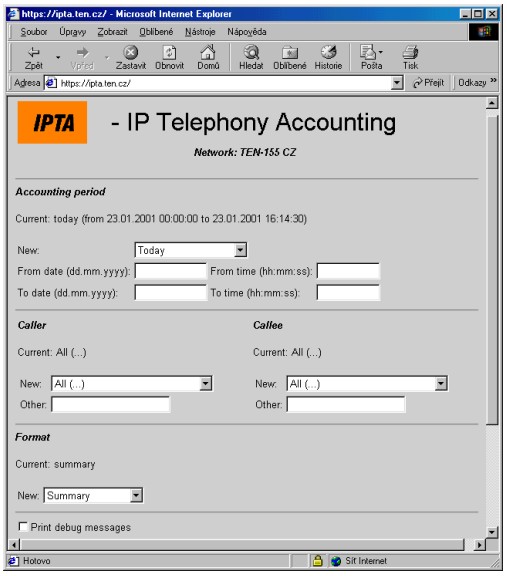


Fig. 6. Specifying accounting parameters

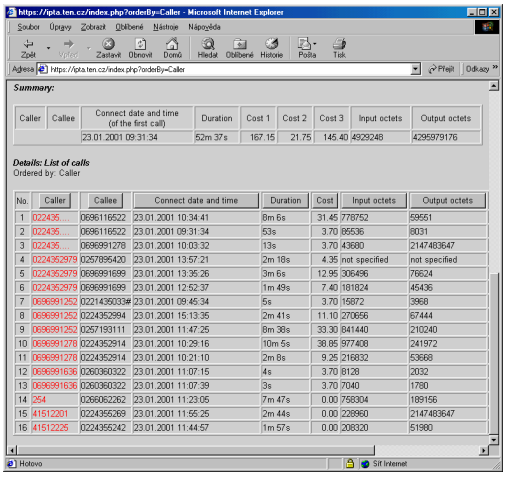


Fig. 7. Example of accounting information listing

are possible prefixes of local mobile phone operators. The form for the specification of accounting parameters is shown in Fig. 6. An example of a detailed listing of calls is shown in Fig. 7.

The user can specify the accounting scheme to be used. It is possible to account calls by several accounting schemes in parallel and the difference in cost can be presented. By combining aggregate listings and accounting schemes we can find patterns suggesting the most economical location of the gateway to the public telephone network.

The presentation part selects already processed calls from tables for individual months. Depending on the required range, calls from several tables may need to be aggregated. If the required range includes the current day, the `processNewCalls()` function is also called to obtain information about the recent calls, which have not yet been moved to the archive tables.

VIII. Conclusion and Future Directions

We successfully set up a wide area IP telephony network using Cisco routers and PBXs of different brands and types. The network is connected to PSTN and to the mobile phone network. Our experience is that H.323 is currently more reliable when using IP network based primarily on Cisco routers. However, development of SIP in Cisco IOS is under way and we plan to use SIP later to benefit from its advantages. We are currently continuing in tests of SIP-based communication using Cisco 3600 routers.

We developed an IP telephony accounting application which provides detailed information about individual calls as well as high-level summaries. The flexible structure of the accounting application allows using different methods of gathering accounting information from different voice gateways and accounting calls using multiple accounting schemes in parallel.

The application successfully gets over changes in RADIUS messages caused by changes in router configuration and it copes with missing or damaged RADIUS messages using a call leg cache. Another feature which should be available soon at least for some users (depending on the type of interface connecting the PBX and the used signaling) is the indication of the caller's name (in addition to the number) on the display of the callee's phone. We successfully tested this feature.

We plan to make experiments testing the behaviour of VoIP services running over the network with controlled QoS parameters in the cooperation with the QoS in IP working group that is implementing diffserv-based QoS in the TEN-155 CZ network.

References

[1] International Telecommunication Union, "Packet Based Multimedia Communication Systems", Recommendation H.323, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, February 1998.

[2] Handley, M., Schulzrinne, H., Schooler, E., Rosenberg, J. "SIP: Session Initiation Protocol", *Request for Comments 2543*, Internet Engineering Task Force, March 1999.

[3] Handley, M., Jacobson, V. "SDP: Session Description Protocol", *Request for Comments 2327*, Internet Engineering Task Force, April 1998.

[4] Voznak, M., Bohac, M. "Experience with Operation of IP Telephony and its Integration into Current Telecommunication Infrastructure", COFAX conference, Bratislava, Slovakia, May 2000.

[5] Ligney, C. "RADIUS Accounting", *Request for Comments 2139*, Internet Engineering Task Force, June 2000.

[6] *ICRADIUS server*, http://icradius.hislora.com.au.

[7] Huff, T. "Configuration Guide for AAA Billing Features in Cisco Voice-Enabled Routers and Access Servers", Cisco Systems, 1999.

[8] "RADIUS Vendor-Specific Attributes Voice Implementation guide", Cisco Systems, 1999.

Attribute Based Addressing for SIP

Vlasios Tsiatsis †*, Jyh-Cheng Chen ‡, Prathima Agrawal ‡, Mani Srivastava †

† Electrical Engineering Department, University of California at Los Angeles
Los Angeles, CA 90095-1594
{tsiatsis, mbs}@ee.ucla.edu

‡ Applied Research, Telcordia Technologies, Inc.
Morristown, NJ, 07960-6438
{jcchen, pagrawal}@research.telcordia.com

***Abstract*—The Session Initiation Protocol is used for setting up multimedia sessions among users in such a way that personal mobility is handled by the use of one identifier to distinguish among users. This identifier in spite of being convenient and simple does not fully describe the communicating entities in terms of their capabilities and their characteristics. In this paper we address the problem of describing the communicating parties with multiple attributes in the context of SIP. We present the architecture of the prototype system and its integration with the SIP protocol. The SIP protocol regards the location service as an orthogonal part of the architecture and does not deal with this issue. However the architecture of the location server plays an important role in the operation of the protocol. Based on the experience gained from the operation of prototype system we present some enhancements in terms of the interoperability of the location service and the SIP server.**

***Index terms*—session initiation protocol, attribute based addressing.**

A. Introduction

The Session Initiation Protocol [8] utilizes a single user identifier similar to an e-mail address in order to distinguish among the different users. The user contact information for the session setup consists of a user name, the machine name and port number where the SIP client runs on. The machine name can be either its IP address or its fully qualified domain name. Thus the SIP contact information is already a collection of attributes with the machine name being the one, which may need a translation from a fully qualified domain name to an IP address. The need for this duality in machine identifiers is a result of the difficulty faced by humans to associate a device with a string of numbers (IP address) and to be able to use this address when referring to this machine. The fully qualified domain name enables a more natural description of a device, which is easy to memorize and associate as well.

This naming scheme greatly reflects the location of the networked entities because the primary concern in the early networks was to route information from one entity to another. And since there were not many networked entities the IP address naming scheme worked perfectly all right.

Moreover, the naming scheme used today although simple[10] gives little information about the services that the networked entities provide. The main concern of a user is to use the network, which is attached to his/her machine to perform a task. As a result the user focuses on the functionality that the network can provide and is not willing to make yet another association of an identifier to a desired service.

With the advent of wireless networked machines the need to keep track of the machine location led to patching the existing naming and routing schemes with location directories. These are essentially associations of the unique machine identifiers with new identifiers, which reflected the current machine location.

As the computers and networked devices became smaller and smaller and they also become wireless, the need for new naming schemes became evident. This is because on the one hand the existing schemes are running out of names, and on the other hand they were designed to facilitate the communication of a relatively small number of machines.

Location is only one attribute of an entity that might change. Several other entity attributes may change over time, but existing naming and addressing schemes do not have efficient ways of addressing this problem.

In this paper we address the naming problem providing a naming scheme and a location service for the Session Initiation Protocol [8]. The proposed scheme enables a more flexible and natural user discovery by taking into account more information about the user and his/her context than only a single identifier. This is done by describing a user with a list of attribute value pairs along with the SIP user and contact information.

The paper is organized as follows: Section B describes the motivation behind the proposed architecture. Section C introduces the architecture for our location server. Section D elaborates on the implementation. Section E describes some related work. Section F includes the future work while section G concludes the paper.

B. Motivation

There are three categories in which we can identify the benefit of using an attribute based naming scheme to describe a user:

1. First of all the number of communication devices that are connected on a network as well as the number of users who tend to be mobile is increasing. As a result the identifiers used to distinguish either the devices or the people will become too large that other people will be unable or unwilling to use them because they don't know them or they don't want to know them. All that people

* Part of the work was performed when the author was with Telcordia Technologies, Inc.

want is to be able to address other people or devices by what they are or represent (e.g. PDAs with audiovisual capabilities, nearest policeman).

2. Often people want to communicate with other individuals or groups of people but they only know some characteristics of either an individual or the group. It is more natural to address these people with a description rather than an identifier (e.g all interns in a company).
3. Finally, what it is desirable to have, is not only a service that plays a role similar to the Domain Name System Service. On the one hand it would be good to have a network of servers, which have the ability to resolve network identifiers from high-level descriptions of entities. This kind of service would be used for name resolution for path setup for call setup protocols over IP networks (e.g. SIP) or it could be used for maintaining profile information for users with changing profiles. On the other hand the need for a service that enables users to push their own data to other users that have some common characteristics is gaining interest in the research community. This is what we call profile-based services. One example of this service is location dependent queries for user discovery. Another example is short messages sent to users that satisfy some criteria.

C. ARCHITECTURE

1st. Overview of SIP

Since our attribute-value based naming system assists the Session Initiation Protocol (SIP) we present a brief introduction to the protocol.

The SIP is an application level protocol that can establish, modify and terminate multimedia sessions or calls. SIP enables the concept of *personal mobility* by providing name mapping and redirection services. Personal mobility is the ability of users to originate and receive calls on any terminal in any location and the ability of the network to identify end users as they move [12].

The main SIP architecture includes two major components. The first component is the SIP client, which is an application that provides the user with the ability to initiate (INVITE requests) and terminate (CANCEL requests) calls to other SIP clients as well as to register (REGISTER requests) to a SIP server. A SIP Server maintains the SIP registration records for a limited amount of time (soft state registrations) and can setup connections between clients. When a SIP client registers with a SIP server it indicates the user name with which it is going to be known to the SIP protocol along with contact information used for setting up the connection. This contact information includes the user name that runs the SIP client, the machine name and the port number that the SIP client application runs on.

Although two SIP clients can communicate directly without the interference of a SIP server, the server is needed for user registration and call setup. A SIP Server can be configured to operate in one of two modes:

a) Redirect Server Mode (Figure 1)

In this mode a client john@hol.gr sends an invitation request to the server indicating the SIP identifier URI (uniform resource identifier) of the other communicating party helen@ucla.edu. The SIP server contacts a location server and determines if there is a registered user under the requested SIP identifier. If such a user exists, the SIP server returns the contact information of client helen@ucla.edu to client john@hol.gr. In general the SIP identifier and the contact information are not identical. This is due to the fact that a user may want to be known under the SIP protocol by one name while his/her contact information is changing according to his/her current location. When finally the client john@hol.gr receives this information it initiate an invitation request to client helen@veria.ee.ucla.edu directly. If the other party accepts the call an acknowledgement is sent back to the originating client and a real-time session can begin between the two parties. The type of initiated session depends on the device or the SIP capable software. For hardware SIP phones [18] only audio sessions can be initiated while the Columbia software SIP client [19] supports the initiation of real-time audio and video sessions.

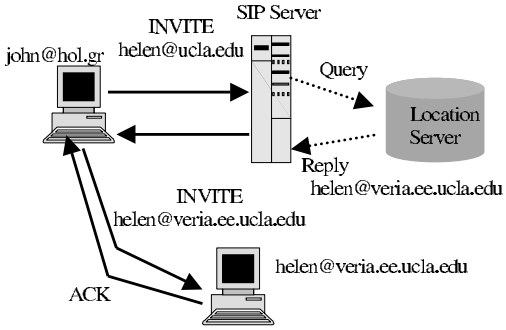


Figure 1. SIP Redirect Server

b) Proxy Server Mode (Figure 2)

In this mode the SIP server does not return the contact information of client helen@ucla.edu to client john@hol.gr but instead sends an INVITE request to client helen@veria.ee.ucla.edu on behalf of client john@hol.gr after contacting the location server. When client helen@veria.ee.ucla.edu accepts the invitation for a connection an acknowledgement of the acceptance is sent to client john@hol.gr. After that, the two communicating parties are ready to initiate a real-time session between each other.

One of the main operations that SIP supports for multimedia real-time setup is location discovery for users. The SIP documentation explicitly mentions that the interaction of the SIP server with the location server is not in the scope of the protocol. However this does not mean that the location server is not an important part of the SIP architecture. We judge that enhancing the intelligence of the location server is crucial for locating users with specific characteristics and this is main the focus of this paper.

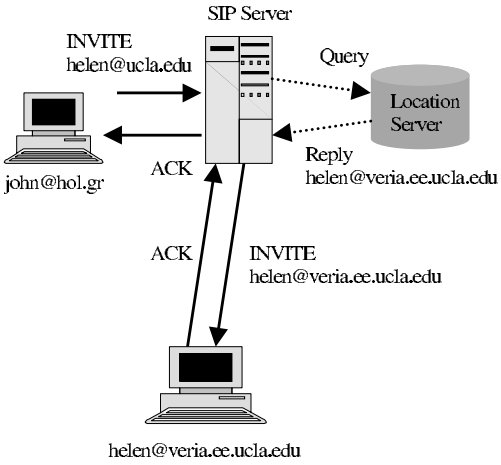


Figure 2. SIP Proxy Server

2nd.Attribute Based Location Server

We considered the implementation of our attribute based location server in two forms. A centralized one, which is used by all the SIP servers to register the SIP clients, and the distributed version, which consists of multiple location servers interconnected together. Each SIP server receives location services from only one distinct location server.

1) Centralized Approach

The overall architecture for a centralized name resolution server of this system is shown in Figure 3. The name server consists of a Database Engine, and a number of Client Communication Components. The number of client components depends on the number of active clients since each time a new connection occurs, a new client component is spawned to provide attribute based naming resolution services to the requesting client. The clients request services from the server using messages formatted in a specific way described later in the paper. The client components communicate with the database engine to update and query the attribute value pair database.

The database engine consists of a Client Request Engine, which is responsible for client queries and a Soft State Manager, which is responsible for maintaining the soft state of each database record. For each record in the database, along with the attribute value pairs that characterize a network entity, there are additional record fields, used by the entities to communicate with other entities. In the case of the SIP protocol the additional information is the SIP contact identifier.

The centralized approach is suitable for isolated environments where the communication traffic to the server is light; otherwise the server becomes the bottleneck of the SIP protocol.

The more interesting approach to architect the location server is the distributed case.

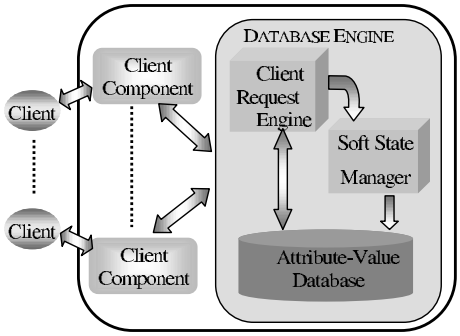


Figure 3. Centralized Server

2) Distributed Approach

In the distributed version of the name resolver different resolvers are connected with each other to provide distribution of the database information and higher availability to the user. The database information can be either replicated in every server or in a set of servers. The replication occurs when an entity registers and updates the registration information. The registration and update messages are then forwarded to every eligible server and result in changes in the database of each server. An eligible server is the one within the scope of a registration or update message. The types of messages as well as their formats are described later in the paper. The way the different servers are connected together depends on the configuration that each server obtains by contacting a server that maintains neighborhood information.

When each server is started, it contacts another server that maintains the connectivity list of the distributed network. The connectivity list is used for forwarding the subsequent messages to the rest of the network. The internal server architecture is depicted in Figure 4.

As in the centralized server case, there is a client component, which is responsible for servicing client requests and a database engine responsible for accessing the local database of attribute value pairs. The additional components shown in Figure 4 enable the server-to-server communication.

There is one packet receiving and one packet sending component for each server. The messages exchanged between the servers carry the appropriate source and destination identification to assist those two components in their operation.

The Duplicate Elimination Tables are tables that maintain information about the processed packets already received or sent so that duplicate messages are not processed again. Duplicate packets can be received when more than one other name servers can contact one server. The list of the name servers that the current one can contact is maintained in the Neighbor Table.

The Open Client Connections is a table that maintains inoformation about the client query requests in progress. The query processing in the distributed case resembles the

recursive query processing in DNS [11]. All the DNS servers are hierarchically connected together according to the domain name they reside in. When a DNS server cannot resolve the requested domain name, it forwards the query to another DNS server that can handle the request. In the meantime the client waits until the query reply comes back through the reverse path. In our case the query is also recursive and as a result all the information about the waiting client must be stored in the Open Client Connection Table so that the query reply is forwarded to the appropriate client.

More details about the server-to-server communication will be given in the next section.

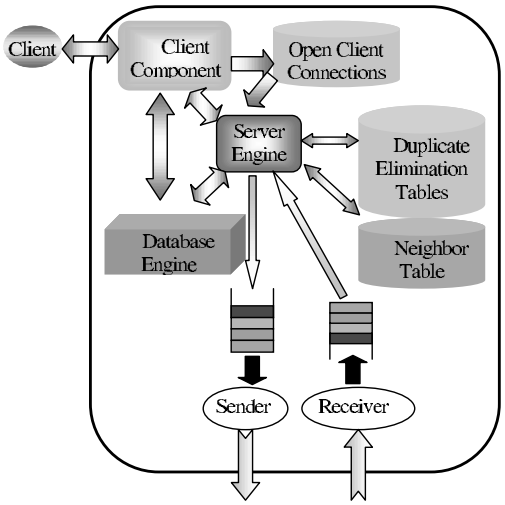


Figure 4. Distributed Server Case

D. Implementation

The implementation of the attribute based naming server was done in Java in order to provide a portable proof of the concept of attribute based naming of networked entities. The integration of the naming server as a location server for SIP was performed using the SIP client and server implementations from Columbia University [19]. Our goal was to provide an implementation that would help us understand the problems faced by introducing such a service not only to SIP but other environments as well[14].

1st. Database structure

The database of the entity descriptions consists of a list of records, which contain the following information: an entity identifier which acts as a handle for update operations, a list of attribute value pairs, some entries for database maintenance (record number, timestamp, expiration time), the IP address and the port number of the entity and a user name, which is associated with this entity (in our case the entity is a SIP user).

The structure of a record is shown in Figure 5. The list of attribute value pairs consists of a list of triples (AttributeName, Operator, Value). In the current implementation every triple has the equality operator ("=") as its operator. However an entity can also have an attribute described by using other operators. An example is the range operator used to describe that a user's work hours are from nine to five.

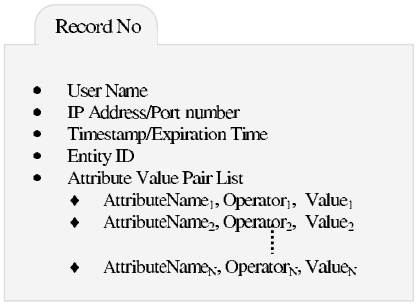


Figure 5. Database Record

2nd.Message Types

A client connecting to an attribute based name server can perform the following operations:

1. Registration
2. Signoff
3. Description Modification
4. Query
5. Application Level Forwarding

Clients request each of the above operations by connecting to the name server and sending a message formatted in a specific way. In the current implementation the message is send as a string of characters without any coding, for compression or encryption purposes. The first part of the message contains the identification of the source of the message in terms of a user name, an IP address, and a port number and a keyword that distinguishes the type of the message. The next part is a field (time-to-live field) that facilitates the scoping of the messages in terms of number of server hops that the message can travel. The structure of the rest of the message depends on the message type.

1) Registration

A sample registration message is shown in Figure 6(a). This type of message is characterized by the keyword **register,** which is followed by the list of attribute value pairs that the requesting entity possesses. When the registration of the entity is completed the name server gives the client a handle for that entity. This handle acts as a certificate that the client shows to the name server when the former requests an update of the registration record. This handle is a string of hashes concatenated together. These hashes are produced by the fully qualified domain name of the server, the originating user information, the register request string and a server sequence number. The possible types of updates are the signoff of an entity from the database, and the modification or deletion of a set of the entity attributes.

2) Update Operations

Examples of a signoff message and attribute deletion and modification messages are also shown in Figure 6(b)-(d). The only information that a signoff message has to carry is the record handle returned by the registration request. An attribute deletion message carries the identification of the user whose description needs to be changed and the names of the attributes that need to be removed. An attribute modification message contains the user record handle and a list of attribute value pairs. The attribute names of the pairs may or may not exist as part of the original user description. In the former case the attribute value is changed, while in the latter case the new attribute value pair is inserted in the user description.

```
(a) src(vlasios@veria.ee.ucla.edu:4443)
    ttl(2)
    register
    #a=1#b=2#c=3#
(b) src(vlasios@veria.ee.ucla.edu:4443)
    ttl(255)
    signoff userid (1345_345_224_1)
(c) src(vlasios@veria.ee.ucla.edu:4443)
    ttl(255)
    del_attr userid(1345_345_224_1)
    #a#c#
(d) src (vlasios@veria.ee.ucla.edu:4443)
    ttl(255)
    mod_attr userid (1345_345_224_1)
    #a=3#d=4#
```

Figure 6. Message examples

3) Query

A client can issue a query, which is, in general, a combination of primary queries. A primary query consists of an attribute name, a matching operator and an attribute value. An attribute value can be either a number or a string of characters.

The primary query operators are:

1. exact match operators(Attribute = Value),
2. range operators ($A_1>V_1$, $A_2>=V_2$, $A_3<V_3$, $A_4<=V_4$, A_5 @[V_5-V_6]),
3. set operators (A @[V_1, V_2, .., V_N]) and
4. the wildcard operator (=*)

The notation A_5 @[V_5-V_6] means that the primary query will be satisfied for those entities that have the attribute A_5 in the range [V_5, V_6]. The notation A @ [V_1, V_2,.., V_N] means that the primary query will be satisfied for those entities that have an attribute A belonging into the set [V_1, V_2, ..., V_N]. A number of primary queries can be combined together, using AND/OR operators together with parentheses to construct composite queries. An example of a composite query is presented in Figure 7. A client that issues this query is looking for a policeman or a firefighter at Zimbabwe.

A query message has the keyword **query** followed by a query modifier and the actual composite query. The query modifier modifies the output of the actual query. The output of the actual query is a set of records that satisfy the query. Out of these records the user may choose to filter these records so as to get any of the records, or all the records or the record, which describes an entity, that is nearest to the client that issues the query. The modifier used in each case is ANY, ALL, NEARESTFROM respectively. In the example in Figure 7 the user requests that all the matching records be returned in the query reply.

```
(a) src(vlasios@veria.ee.ucla.edu:4443)
    ttl(255)
    query
    all
    location = Zimbabwe
    AND
        (occupation=firefighter
        OR
        occupation= policeman)
```

Figure 7. Query Example

In order for the NEARESTFROM modifier to have a meaningful output the users must also register their locations. In our prototype implementation for demonstration purposes we assume that the location of a user is given as a triple of the x, y and z coordinates of a coordinate system whose origin is at the center of a city where the SIP network of servers is employed. Without loss of generality the location of a user can be an arbitrary character string (e.g. Room 1045) as long as there is a well-defined ordering function among those strings in terms of the location.

The location attribute is useful when the entities are spatially distributed over an area of interest or the client is interested in having location dependent services as in the case of a query with the NEARESTFROM modifier.

Query → SetModifier BooleanQuery

- SetModifier →
  - any
  - all
  - nearestfrom(x,y,z)
- BooleanQuery → ANDquery (OR ANDquery)*
- ANDquery → PrimaryQuery (AND PrimaryQuery)*
- PrimaryQuery →
  - (BooleanQuery)
  - ID (= |> |>= |< |<=) (ID | NUM)
  - ID @ [(ID |NUM) - (ID |NUM)]
  - ID @ [(ID |NUM) , (ID |NUM), ... , (ID |NUM)]

ID : character string, NUM : number

Figure 8. Query Grammar

3rd. Query Language

The query grammar is depicted in Figure 8. A query is preceeded by a query modifier called SetModifier because it determines the output of the query, which is in general a set of matching records. The actual query can be viewed as an expression tree with the operators AND and OR as the non-leaf nodes and PrimaryQueries as leaf nodes. An example of an expression tree is shown in Figure 9. A PrimaryQuery can be one of the four types of primary queries depicted in Figure 8.

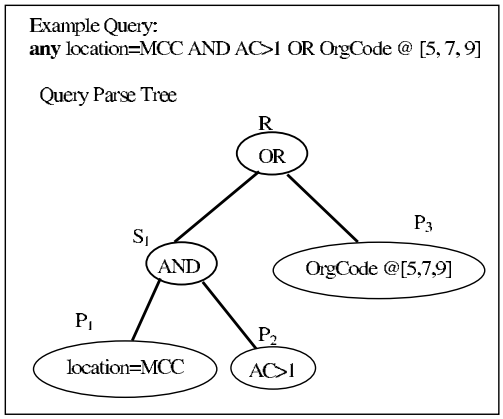


Figure 9. Query Parse Tree

4th. Lookup Algorithm

After the descriptive name server receives a query, a query parse tree is formed based on the query string. An example of a query and a query parse tree is shown in Figure 9. The first part of the message that contains the **query** keyword and the source identification is omitted is Figure 9. The query targets the users that work in location MCC and have authorization code greater than 1 or their organization code lies in the set [5,7,9].

According to the query grammar the parse tree will have a root (R) of an OR operation with two subtrees (S_1 and P_3) which correspond to one AND operation and one primary query. The AND operation will have two primary queries as leaves. As a result, primary queries P_1 and P_2 will produce the result sets Set_1 and Set_2. Because of the fact that P_1 and P_2 are leaves of a subtree which has an AND tag the result of the operation that this subtree corresponds will be the intersection ($Inter_1$) of Set_1 and Set_2. The result of the right subtree query will be Set_3. Finally the result of the tree R will be the union of $Inter_1$ and Set_3 ($Union_1$). Because of the query modifier ANY the result of the original query will be any random record of the $Union_1$ set.

5th. Application Level Forwarding

Application level forwading in our service is the ability of the location service to support forwarding of packets from one user to another using the location server network. The forwarding of these packets is subject to conditions that resemble a query.

An application-level-forwarding-request message consists of two parts, apart from the source identification part and the TTL field, which are carried on every request. The first part is a condition expressed in terms of a query and the second part consists of the data that the requesting client sends to other entities that satisfy the query. The data portion of the message is encoded in base 64 encoding since the whole protocol is character string oriented. An example of an application forward message is shown in Figure 10.

```
(b) src(vlasios@veria.ee.ucla.edu:4443)
    ttl(255)
    app_lev_forw
    filter (location = Zimbabwe
        AND
                (occupation = firefighter
                OR
                Occupation = policeman)
    )
    packet (Base64 encoded message)
```

Figure 10. Application level forwarding example

6th. Reply Format

The replies each server sends back to the requesting client are given in the form of XML [2]. XML is the perfect candidate for this kind of a naming system. It is first of all a string-oriented language in order to be compatible, portable and easy to debug. Secondly structured information can be easily represented in XML. On the one hand entity registration and signoff, attribute deletion and modification can result either in success or failure, which can be represented by just one success or error message. On the other hand a typical query reply will result in a list of entities, which in turn have their own lists of attribute value pairs. This kind of information fits perfectly into the XML specification. Figure 11(a) presents a successful registration reply, which includes the entity handle for future updates. Figure 11(b) shows a query reply for which the query was not satisfied, while Figure 11(c) shows a query reply which includes all the entity information in a structured way.

7th. SIP Integration

For the purposes of a proof of concept implementation the Columbia SIP client and server was used while the first author was with Telcordia Technologies, Inc. The SIP client is written in Tcl/Tk while the SIP server is written in C. The operations that the SIP client allows are user registration, user signoff and refresh of user registration because of the soft state registration. The attribute value naming system provides similar operations in addition to attribute value pair modification and attribute value pair addition and deletion operations. With these additional operations an entity can have a variable behavior in terms of the services this entity provides

and causes the queries of the other entities to have variable outcomes. Therefore we integrated these additional operations in the SIP client as well as the SIP server.

SIP provides a mechanism for its extensibility by means of stating in a SIP packet if the packet needs special handling ("Require" packet header). Therefore providing a new user location service is smoothly integrated with the SIP protocol by using the special SIP packet header. Every SIP packet originating from a SIP client to a SIP server carries this special header if the attribute based naming system is used. Along with the "Require" header another header ("Abea-name") was added. The additional header serves the purpose of carrying the information specific to the attribute based naming system. This information is the actual message (registration, signoff, attribute modification/deletion and application level forwarding). In the reply packets the same header operates as the reply from the location server. All the necessary User Interface components for the additional operations were added to the Tcl/Tk SIP client and all the demultiplexing code for the new location service was incorporated in the SIP server. These include components for registration and modification of an entity description, as well as for the query construction. The registration, the signoff and attribute modification/addition/deletion operations are encapsulated in SIP REGISTER requests, while the user query is encapsulated in SIP INVITE requests.

```
<reply>
     <userid>AF134BC_CD975_1</userid>
</reply>
                    (a)
<reply>
     <error>User Not Found!</error>
</reply>
                    (b)
<reply>
  <list>
    <entity>
      <user>
        helen@veria.ee.ucla.edu:4500
       </user>
       <attribute_value_list>
         <item>a=1</item>
         <item>b=new york</item>
         <item>d=876</item>
       </attribute_value_list>
    </entity>
  </list>
</reply>
                    (c)
```

Figure 11. XML Reply Examples

After the SIP server receives a SIP packet that requires special handling the attribute based naming server is contacted to take care of the special operation. This server processes the request and returns an XML reply to the SIP server. If the request was a registration/signoff/attribute modification addition and deletion then the reply does not need to be further processed by the SIP server. The XML reply is subsequently sent to the SIP client for being processed. If the request was a query request then the SIP server processes the XML reply, formats the reply according to the SIP protocol (list of contact information) and returns the reply to the SIP client. No further processing is need on behalf of the client because the reply does not contain any special handling directive.

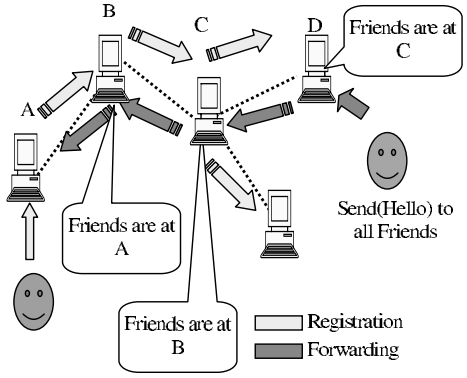


Figure 12. Application level forwarding

8th. Information Distribution and Request Processing

In the case of an entity registration, entity signoff and entity description modification the requests are forwarded to the whole distributed system or to a set of servers. This is accomplished by scoping the messages using the time to live message field.

In the case of registration and update messages the Server Engine updates the local database, determines the neighboring server(s) and forwards the requests to the set of qualifying servers (neighbors with the message scope). They, in turn, update their databases and forward the requests to their neighbors.

In the case of a client query the Server Engine searches the local database and if any registration records are found, it returns them to the requesting client. Otherwise the necessary information for the open client connection is stored in the Open Client connection table and the query request is forwarded to the neighbors of the current name server. This form of a query request follows the recursive query principle, which is also followed by the DNS servers. If the query reaches a terminal server (no neighbors other than the one that send the request) and no records satisfy the query, no negative reply is sent back. In this way the network does not fill up with unnecessary negative repiles. The purpose of a query is to find some entities satisfying the query and not to receive negative replies. This means that the originating server implements a mechanism to send a negative reply to the client after a timeout

period. If negative replies were tracked, additional state should be preserved in each server that forwards the query, so that it gathers all the replies from the other servers (downstream servers) and send one reply back to the server that it got the request from (upstream server). In this case the timeout mechanism would also be necessary for each upstream server.

When an application-level forwarding message enters a name server, the name server searches the local database to find all the records that satisfy the query. If some records are found, the server determines if the entities described by the records have registered to the current server directly or via a registration forward message from another server. If the entity has registered directly with the current server the data part of the message is sent to the matching entities using UDP packets. If the entity has registered directly with another server the application level forward packet is sent to that server. Thus if the registration message has gone through a series of servers then the application level forward packet will go through the reverse path (Figure 12). If no records were found in the local database the message is forwarded to the neighbors of the current server and the procedure above is repeated.

E. RELATED WORK

1st. Content based routing

In [4] the authors have implemented a content based addressing and routing architecture based on the communication model of an event notification service. This model clearly differs from our approach in the proposed architecture. They base the content routing on the event notification service called SIENA [3], which is in principle, a publish/subscribe service. The users interested in some piece of information subscribe their interests in terms of constraints on attribute values. Each interested user registers a filter similar to our query and every piece of published information (notification) is filtered through the user subscription (filter). If a notification matches the subscription then the notification is forwarded to the interested party. The philosophy of this approach is clearly different from our approach in which users issue the filters that scan the information database of an attribute based location server. However our approach could benefit from the techniques for merging subscriptions when one subscription can be logically inferred from another. These techniques could be used for implementing an indexing scheme for more efficient searching through our distributed system.

2nd. Lightweight Directory Access Protocol, WHOIS and WHOIS++ Service

The Lightweight Directory Access Protocol (LDAP)[17] is a protocol for accessing directory services like X.500 [7]. However it is not extensible and flexible enough to provide a location service that can do more than just binding of user descriptions to low level addresses. The WHOIS and WHOIS++[9][6] services are designed to provide information about the Internet users similar to our scheme but the supported query language is limited. It supports only equalities, regular expressions and wildcards combined with the AND, OR and NOT boolean operator, thus making the matching procedure purely string oriented. Our scheme provides a richer collection of operators and matching mechanisms.

3rd. Service Discovery Protocols

The Service Location Protocol [15] is a protocol for discovering network services with a different query language that firstly indicates the type of service that is being searched and then the attributes of this service in a structure-free manner. The services register with a central directory agent that maintains all the necessary information about the available services and handles the client requests. Our approach is not restricted to only network services and it does not include a centralized repository of user registration information. Moreover our query language has more operators and a query modification machanism that is applied on the result of the query.

The Secure Service Discovery Service [5] proposes an architecture of hierarchically connected directory servers that maintain the descriptions of network services in XML [2]. Additionally the servers exchange indexing information in the form of lossy compressed summary service records. The query language [16] used by SDS is based on the XML, and it is less expressive than our query language since it is oriented to search through XML structured information.

4th. Intentional Name Resolution

Intentional Name Resolution [1] from MIT addresses the descriptive name resolution and application level packet forwarding problem by creating a distributed system of descriptive name resolvers (INRs). These resolvers are also responsible for disseminating routing information about the registered intentional names. Networked entities are described with attribute value pairs that are hierarchically structured in order to narrow down the search space. However this approach has the drawback that the queries must also be structured in the same way that the information is structured. Therefore the entities issuing queries must know the exact information structure so that their queries are correctly resolved. Our flat naming scheme does not impose this limitation and it enables us to also incorporate information similarity indicators that can be used for evaluating the relevance of a descriptive name.

F. FUTURE WORK

The current status of the implementation enables a SIP user to invite another user issuing a query with the following modifiers: a) ANY: return any record that matched the query, b) ALL: return all records that matched the query and c) NEARESTFROM(x,y,z): Return the records that are nearest to the given point d) modify and delete attributes thus changing the behavior of the query matcher. If an entity d changes a type of service, it can use this mechanism to update its registration.

The scope of the registration messages can be selected by the client so as to minimize the amount of the forwarded registration messages and the amount of storage for each server. This solution however calls for an indexing scheme since the unsatisfied queries in one server will be virtually

broadcasted in the whole network of the location servers. This problem can be solved by having the registration and update messages forward the attribute names beyond the scope of the message. In this way each neighbor in the neighbor table will have a list of attribute names that the server knows something about. The server either has at least one record that includes this attribute name or it knows another server that has similar information. In this way the queries are forwarded to the neighbors that have some information about some attribute names. In addition to maintaining attribute names each neighbors can have indexes of attribute names as well as the ranges of the values of attributes. Moreover an indexing scheme similar to the one used in [4] can also be employed.

With the attribute based naming system in which the queries are an important part (for the INVITE requests and for the application level forwarding), the SIP users can implement some interesting types of services like the following:

1. The users can register with the following attributes: i) DISCLOSE_USER = [userlist] which means that if the user that issues the query is in the user list then and only then a matched record is disclosed and it is taken in account for the filtering process that the modifier imposes, ii) DISCLOSE_MACHINE = [machinelist]; in this case the records are disclosed if the originating client issues the query from a machine listed in the machinelist, iii) DISCLOSE_TIME[FROM-TO]; this attribute is used by a user that requests from the system to disclose her information if the query is issued in the time interval [FROM-TO].
2. A message system can be implemented using the application level forwarding ability of the location server. This can be done by encapsulating the application level forwarding message inside an INVITE request by using the "Require" keyword in the SIP packet. A unified messaging system was proposed by [13], which involves SIP and RTSP (Real Time Streaming Protocol). Our system enables a different architecture for implementing a messaging system using the added functionality of the location service.
3. Since a query can be forwarded in the network of the location servers it would be good if the location server had the ability communicate with SIP clients and servers. The intuition behind this is that a query may reach a destination user's network and then return to the originating user. After that the originating user sends an INIVTE request. If the location server is able to play the role of a proxy it can initiate a call thus eliminating one roundtrip message exchange.

G. Conclusion

The Session Initiation Protocol is used to setup real-time sessions among users that register with a SIP Server using user names that resemble an e-mail address. The location server utilizes the user information in order to find the user's contact information, which serves as the other communicating party's SIP address. In this paper we deal with the problem of making the INVITE requests address a user with a more natural way by using attribute value pairs. We have presented the overall architecture for two types of location servers, a centralized and a distributed one. A distributed location server enables a user to register locally while it can be found by any other SIP server that uses another location server connected with the former one. This can be done either with full replication of the location information or with scoped registration and indexing of the attribute names. Along with the ability to locate mobile users a messaging service based on the application level forwarding feature can be implemented.

The attribute based naming system proposed for the location server enables the SIP users to initiate calls to users whose exact SIP addresses may not be known. The only requirement is that the users register with enough attribute value pairs so that other users can discover them using the proposed attribute based location service.

H. References

[1] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, J. Lilley, "The Design and Implementation of an Intentional Naming System" Operating Systems Review, vol.33, no.5, Dec. 1999. pp. 186-201.

[2] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C Recommendation, October 2000, http://www.w3.org/TR/REC-xml.

[3] A. Carzaniga, D.S. Rosenblum, A.L. Wolf, "Achieving Expressiveness and Scalability in an Internet-Scale Event Notification Service", Proceedings of the Nineteenth ACM Symposium on Principles of Distributed Computing (PODC2000), Portland OR. July, 2000.

[4] A. Carzaniga, D. S. Rosenblum,, A. L. Wolf "Content-Based Addressing and Routing: A General Model and its Application", Technical Report CU-CS-902-00, Department of Computer Science, University of Colorado, January, 2000.

[5] S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, R. H. Katz, "An Architecture for a Secure Service Discovery Service", Proceedigs of Fifth Annual International Conference on Mobile Computing and Networks (MobiCom '99), pp. 24-35.

[6] P. Deutsch, R. Schoultz, P. Faltstrom, C. Weider, "Architecture of the WHOIS++ Service", RFC 1835, August 1995.

[7] The Directory: Overview of Concepts, Models and Service. CCITT Recommendation X.500, 1988.

[8] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg, "Session Initiation Protocol", RFC 2543, March 1999.

[9] K. Harrenstein , M. Stahl., E. Feinler, "NICNAME/WHOIS", RFC 954, October 1985.

[10] P. Maniatis, Mary Baker, "IdentiScape: Tackling the Personal Online Identity Crisis.", Technical Report CSL-TR-00-804, Stanford University, June 2000.

[11] P. Mockapetris, K. J. Dunlap, "Development of the Domain Name System", Proceedings of SIGCOMM '88, pp. 123-133.

[12] R. Pandya, "Emerging mobile and personal communication systems", IEEE Communications Magazine, vol. 33 , pp. 44-52, June 1995].

[13] K. Singh, H. Schulzrinne, "Unified Messaging using SIP and RTSP", IP Telecom Services Workshop, Sept. 11, 2000. Atlanta, Georgia.

[14] D. Tennenhouse,"Proactive Computing" Commun. ACM 43, 5 (May. 2000), pp. 43-50

[15] J. Veizades, E. Guttman, C. Perkins, S. Kaplan, "Service Location Protocol", RFC 2165, June 1997.

[16] Xset Database and Query Engine, http://www.cs.berkeley.edu/~ravenben/xset/

[17] W. Yeong, T. Howes, S. Kille, "Lightweight Directory Access Protocol", RFC 1777, March 1995.

[18] http://www.3com.com/products/sip/index.html

[19] http://www.cs.columbia.edu/~hgs/sipc/

1

Internet Telephony Traversal across Decomposed Firewalls and NATs

Jiri Kuthan
kuthan@fokus.gmd.de

GMD-Fokus

April 2001

Abstract

We introduce a decomposed architecture that accomplishes traversal of Internet telephony sessions across firewalls and Network Address Translators (NATs). As opposed to building monolithic firewalls and NATs with Application Layer Gateways (ALGs), we suggest placing application logic out of routers into signaling servers. Expected benefits of decomposition are better maintainability, lower costs, higher performance and enabling security in application-layer hop-by-hop signaling. The missing piece needed to build the proposed architecture is a protocol that connects individual elements of the decomposed architecture. Minimum required functionality of the protocol is opening/closing pinholes in firewalls and maintaining address translations in NATs. We call this protocol *Firewall Control Protocol* and discuss design issues related to it.

***Index terms*: Internet telephony, Firewalls, NAT, Firewall Control Protocol**

A. INTRODUCTION

Internet telephony is a novel application with numerous appeals to both users and service providers. In beginnings of Internet telephony, the main driver was economical transport of long-distance phone calls. However, it is being recognized that the major appeal is open service model, programmability and ability to integrate voice with numerous Internet services. Click-to-call, instant messaging, video conferencing, gaming and virtual reality sessions are only a few of numerous examples.

The major components of Internet telephony architecture are media transport and signaling protocols. The primary purpose of a signaling protocol is session control between two or more participants. The session protocol is used to locate prospective participants, negotiate session endpoints and session parameters such as audio codecs used. Currently, there are two signaling protocols: Session Initiation Protocol (SIP [1]) developed by IETF and H.323 [2] developed by ITU-T. Problems addressed in this text are very similar for both protocols. We focus only on SIP which we believe to be the real enabler of integrated services[1].

Unfortunately, firewalls and NATs seriously inhibit operation of Internet telephony. Static filtering policy prevents dynamic media packet flows from traversing firewalls. Network address translation causes a mismatch between translated packet flows and not-translated addresses conveyed in signaling and results in a failure to establish a session as well. As a result, user behind firewalls/NATs cannot communicate with users on the Internet. It is hard to estimate how many users are affected by presence firewalls and NATs for many reasons, however it is not unreasonable to say that the problem size is big. Brian Carpenter's recent hand waving estimate[2] was 40% of users (160 Mio.) were behind a firewall and/or NAT.

In the remainder of this paper, we suggest a new architecture that improves existing solutions primarily in terms of maintainability. Section B refreshes the notion of firewalls and NATs and explains why they inhibit Internet telephony sessions. Our solution, decomposed firewalls/NATs, is introduced in Section C. Section D reviews design issues related to central mechanisms of our proposal, Firewall Control Protocol. Related work dealing with this problem is reviewed in Section E.

B. BACKGROUND

1st. Firewall Refresher

Generally, firewalls serve the purpose of centralized network security administration. In networks with many hosts, it is infeasible for administrators to maintain all of them and check them for bugs or malicious code threatening security of the device, its users or the network. Thus firewall administrators define a restrictive traffic policy that typically allows only traffic from and to processes considered trustworthy. This policy is

---

[1] Readers interested in comparisons of both signaling protocols may want to check the following webpage: *http://www.fokus.gmd.de/glone/projects/ipt/sip.html*

[2] *http://www.fokus.gmd.de/usr/jku/private/fcp/email-notes/bc1.html*

enforced centrally in a firewall through which all traffic passes.

An inherent problem of such centralized enforcement points is they are also centralized points of failure and subjects to performance bottlenecks. Alternative solutions that rely on pushing the policy to all administered hosts have been proposed [3]. Software based on this scheme is also implemented though not widely deployed. We focus only on traditional centralized firewalls in this document.

Though firewalls operate at network/transport layer application-layer policy may also be enforced. The construct used deploys proxies, i.e. trusted processes that relay application messages and enforce application-layer policy on them. Firewalls are configured to permit application traffic identified by port numbers only to and from the proxies.

Firewall policy is also frequently used to prevent internal network users from "doing bad things" such as looking at entertainment web-sites during office hours and wasting network bandwidth. It should be noted that though unknowledgeable users are prevented from doing so, motivated inventive users are not. With some effort they always can build tunnels that meet security policy but contain traffic whose presence was unwanted. Software for subverting firewall policy is publicly available. (Check "Linux Firewall Piercing"[3] or Http Tunnel [4].)

2nd. NAT Refresher

NATs[4] were primarily introduced to conserve exhausted IP space. NATs allow hosts to share IP address. They operate transparently without hosts being aware of them.

Meanwhile, NAT has found also other uses -- it is deployed for load balancing, address translation between IPv6 and IPv4 realms[5] or transparent recovery from access failures in multi-homed networks. It is also believed that NATs increase network security by obfuscating addresses of hosts behind them. However this belief does not seem to have a reasonable justification.

Network Address Translation is known to cause numerous troubles[6,7]. T. Hain summarized architectural implications of NATs in [8]. He identifies the major drawback of NAT in loss of transparency of end-to-end connectivity. (Carpenter explains Internet transparency, the end-to-end model and threats to it in [9].) Transparency loss leads to numerous problems such as inconsistent address translations due to parallel NATs and alternating routing, higher overhead and lower, inhibiting security protocols relying on value of IP address (*ipsec, dnssec, Kerberos, ssh)*, inhibiting applications relying on session control protocols, etc. K. Moor, a dedicated NAT opponent, maintains a webpage with a collection of arguments against using it[5].

Regardless of NATs' drawbacks it is unlikely they will disappear from the Internet in the near future. An ultimate solution to the lack of address space would be deployment of IPv6 however it is unclear in which time frame it will be deployed if at all. Hope still lives that mass deployment of IP-only cell phones will encourage IPv6 deployment.[6]

3rd. VoIP Incompatibility with Firewalls/NATs and Solution Space

Unfortunately, both firewalls and NAT are a serious obstacle to deployment of Internet telephony.

The source of problems with firewalls is their packet filtering policies are static whereas Internet telephony introduces dynamic conditions. Session signaling protocols are used to negotiate dynamic IP addresses and port numbers, signaling may be initiated by third parties, call parties may change IP address during a session, etc. Static IP/transport layer policy simply does not capture the dynamic nature and complexity of Internet telephony. In particular, SIP signaling messages convey dynamic IP addresses and UDP port numbers of media receivers in their SDP payloads. Static, SIP-unaware filtering policy may permit signaling itself but fails to permit media streams to receivers described in SDP payload.

To enable firewalls to understand application needs while still keeping their filtering policies as restrictive as possible, a real-time link between them and applications is needed. This link enables firewalls to learn what applications currently need and adapt its filtering policy accordingly. Two ways for connecting application with IP/transport layer in firewalls can be used: *explicit* and *transparent*. With an explicit link, applications tell firewalls what they need using a signaling protocol like SOCKs[10]. The application-awareness responsible for signaling to firewalls may be living in end-devices as well

---

[3] http://www.linux.org/docs/ldp/howto/mini/Firewall-Piercing.html

[4] http://www.nocrew.org/software/httptunnel.html

[5] Keith Moore: What NATs break; *http://www.cs.utk.edu/~moore/what-nats-break.html*

[6] Fred Baker posted an interesting e-mail on IPv6 deployment to the IETF mailing list. A copy may be found at *http://www.fokus.gmd.de/usr/jku/private/fcp/email-notes/fb1.html*

as in application relayers. The other way is to build-in application-awareness (called Application Layer Gateways, ALGs) in firewalls, let it inspects application payload and adapt rule-bases transparently.

With NATs, problems are caused by a mismatch between original addresses conveyed in signaling and translated addresses in packet headers of media streams. If a SIP caller or callee behind a NAT advertises a private, non-routable address of its receiver in her signaling message, media streams sent by the other call party to this address will never reach its destination. Details of how NAT affect SIP are well described by Rosenberg, Drew and Schulzrinne in [11].

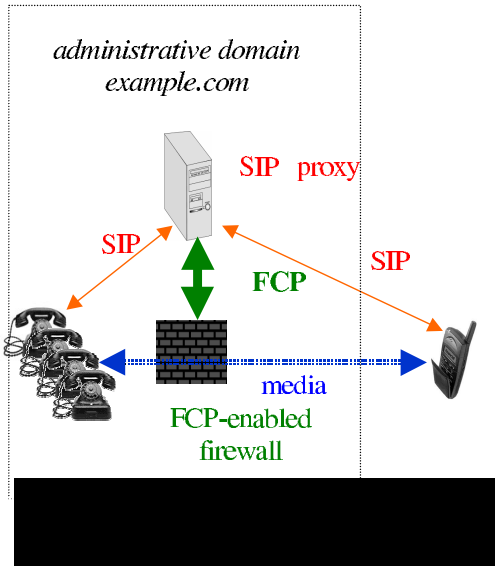
Like with firewalls, two solutions are possible: explicit and transparent. RSIP[12] is an experimental explicit solution developed in IETF. It allows hosts to borrow routable addresses temporarily. ALGs are commonly used to attack the problem in the transparent way.

The choice between explicit host-driven signaling and transparent ALG mechanisms is a trade-off between better end-to-end compatibility and deployment easiness. Transparent ALG mechanisms need no updates of TCP/IP stack or even applications in end-devices. On the other hand, application awareness residing in network devices is a departure from end-to-end model and retains some incompatibilities. In particular, outdated ALGs may not understand new applications, NAT and firewall ALGs cannot inspect encrypted signaling, and NAT ALGs cannot change authenticated signaling.

We argue there is a compromise that merges advantages of both solutions: It does not force upgrading end-device while it significantly softens architectural drawbacks of embedded ALGs. The idea is to reuse application awareness living in call signaling servers and allow it to manipulate firewall/NAT packet processing. The construct relies on the decomposition principle: "divide et impera". The architecture is described in the following section.

C. DECOMPOSED FIREWALLS/NATS

The intent of our architectural proposal is to reuse logic of complex application servers (SIP proxies, H.323 gatekeepers, etc.) as opposed to embedding additional application stack in network devices. In a typical scenario (see Figure 1), an IP telephone sends and receives SIP signaling via a SIP proxy responsible for the administrative domain to which the phone belongs. The SIP proxy inspects signaling and checks it against some application-layer policy. The policy may require the local call party to authenticate, it may drop call requests coming from well-known spam domains, etc. When a call request is approved and both parties agree to set up a call, the SIP proxy instructs the firewall belonging to its administrative domain to punch holes for media streams belonging to the approved call. A protocol, which we call *Firewall Control Protocol (FCP)*, acts as the link between application and IP/transport layers. It allows applications to determine transport/IP layer processing in firewalls, NATs and possibly other network devices. Eventually, both call parties send audio packet streams directly via the dynamically generated pinholes.



Note: A new term capturing a broad class of network devices residing in network and performing functions beyond IP routing is gaining acceptance: *middlebox*. B. Carpenter wrote taxonomy of middleboxes[13]. We focus explicitly on firewalls and NATs in this paper however it could be possible that our proposal is applicable to problems caused by other middleboxes as well.

This construct is quite close to traditional embedded ALGs as it assumes application-awareness residing in the network. However, it fixes many shortcomings from which embedded ALGs suffer.

A major advantage for middlebox vendors is *reduction of development costs*. Reusing application awareness

residing in existing servers relieves middleboxes from understanding and implementing various application protocols and dialects/versions of them. It is responsibility of application protocol developers to bind their applications to them. Instead of developing NxM embedded ALGs for N applications and M middleboxes, N "pluggable" ALGs can be coupled with M middleboxes resulting in N+M protocol stacks.

A great advantage for middlebox operators is they are relieved from *vendor dependency*. Once they are given a generic open interface they will be able to use ALGs supplied by third parties. They even may create their own ALGs. In neither case are they at the mercy of vendors and do not have to wait until a vendor implements an ALG for its middlebox.

Allowing external application servers to control middleboxes also allows for *hop-by-hop signaling security model* that is not feasible with embedded ALGs. This model relies on the notion of transitive trust. It assumes applications in which end-devices are explicitly aware of next-hop server to and from which they relay application messages. The servers propagate them to further servers until they reach a destination end-device. Communication between adjacent hops is secured using a transport or network layer security protocol. All involved hops have unlimited access to messages but the communication between hops is secured against man-in-the-middle attacks. This scheme allows the hops to inspect/modify application messages and control middleboxes accordingly. Though one might have doubts about benefits of hop-by-hop security as compared to end-to-end security, it at least eliminates a subset of security threats. Clearly, end-to-end security does not work with firewalls/NATs at all.

Moving application-awareness out of middleboxes is also *likely to improve their performance*. Being knowledgeable of applications requires keeping and processing application state. On many platforms, application-layer inspection results in interrupts that increase packet processing delay. We expect that if middleboxes speak a simple control generic protocol as opposed to processing a number of complex applications, their packet processing performance will improve.

The essential component of the middlebox architecture is the Firewall Control Protocol. This name refers to a concept rather than to a specific protocol. It is quite likely, that an existing protocol such as SNMP, Cops or Diameter will be used for this purpose. We describe protocol design issues in the next chapter.

D. FIREWALL CONTROL PROTOCOL: DESIGN ISSUES

4th. Functionality Scope

The primary objective of the Firewall Control Protocol is to link application logic residing in application servers to IP/transport layer logic residing in firewalls and/or NATs. As firewalls and NATs are both technologies oriented on packet-flow processing and are mostly implemented in same devices, it seems reasonable to design a single protocol for dealing with both problems. The protocol must be able to open/close pinholes in firewalls and allocate/release NAT translations.

One could also think of other uses of such a protocol. Suggestions appeared to use it to instruct edge routes to set DiffServ bytes[7] to particular flow or use it by management tools to provision middleboxes in a standardized manner. Especially the latter example would require numerous additional protocol features such as dealing with rule conflicts, support for reflexive rules, ability to query rules, adding notion of direction to the rules, support for keeping track of matched packets, etc. However, as we are primarily interested in solving the firewall/NAT problem and believe in power of simple tools as opposed to all-disease-cures[8] we leave such uses out of scope.

On the other hand we do not want to make prospective future extensions impossible. Particularly, the ability to extend rule language and associate new attributes, for example traffic limitations or expiration period, with pinholes seems to be essential for possible future uses. Nonetheless, care must be paid to usefulness of the specific attributes. An example of little use extensions are attributes whose values are the same across all flows -- such attributes can be easily replaced by a global configuration option. Another example of questionable FCP extensions is attempts to solve end-to-end problems such as QoS by controlling state in a middlebox. As end-to-end problems are believed to be best served by end-to-end protocols[14], manipulating state in the middle of data path seems to be of limited use. Eventually, adding attributes that describe application (e.g., by codec names or application ids) beats the decomposition objective.

---

[7] Some mailing list comments explaining why such scenarios are rather of limited use may be found at http://lists.bell-labs.com/pipermail/sip/2000q2/000688.html and *http://lists.bell-labs.com/pipermail/iptel/2000q4/000486.html*.

[8] Also nicknamed as "Wunderwaffen", "Mega-weapons" or "ocean boiling protocols".

There are also other issues we recommend to leave out of FCP scope. Setting up an operational relationship between a middlebox and FCP controllers is an issue orthogonal to FCP functionality. In common case, there will be a few of trusted controllers maintaining manually configured security associations with nearby middleboxes. Lear examined issues related to middlebox discovery in [15].

We also suggest not to support the ability to re-route application messages to servers that are not on the application path explicitly. Though potentially useful, this feature would introduce additional complexity (e.g,, encapsulation of re-routed packets) and is not needed in SIP scenarios where signaling is typically routed via signaling servers.

5th. Layering

A fundamental design decision on which the decomposed architecture relies is layering, i.e., separating processing logic in independent layers. FCP makes the cut between application logic and IP/transport layer logic in a similar way socket API does. Both layers get reconnected via the notion of packet flows. Packet flows are sequences of packets from same source to the same destination. They are described by packet selectors. Typically, packet selectors include source and destination IP addresses, protocol number, and transport layer port numbers. They may also describe flow aggregates by including wildcards in some fields.

Rules in middleboxes consist of packet selectors and optional attributes associated with the selectors and describing details of packet flows. In the simplest case, rules could include zero attributes and their sole presence would imply specific behavior such as permitting a flow to traverse a firewall.

It is important to emphasize that IP/transport layer flows are an application-independent notion. This allows arbitrary applications to use FCP to manipulate per-flow state in middleboxes so that middleboxes are relieved from understanding individual applications. Applications translate their needs to a generic language.

6th. Decomposed Firewall Security Model

Splitting firewalls into multiple pieces introduces new vulnerability places that were not present with monolithic firewalls. We discuss the security model of the decomposed architecture in this chapter. We focus primarily on firewalls as these are used to make networks more secure and breaking them would be a considerable security concern.

Firstly, we assume that middlebox controllers belong to the same administrative domains as middleboxes themselves and are trusted, administrator-maintained entities. Though this is not dictated by FCP, it best preserves security model of traditional firewalls that are completely under central administrative control and treat end-devices as untrustworthy. Alternative solutions delegating the control to end-devices would constitute a considerable departure from this model and also beat the "easy-to-deploy" objective.

The next assumption is FCP clients use their application knowledge carefully to decide if pinholes should be opened or not. For example, a SIP proxy may require a local call participant to authenticate herself and both call parties to agree on setting up a call before it punches holes for media streams.

Firewall Control Protocol has to be protected against malicious attacks necessarily. If attackers had the possibility to "impersonate" a trusted controller or modify its requests, they could easily corrupt the firewall policy and exploit it for further attacks. It is a must requirement that only unaltered FCP messages from authorized parties may be received in either direction. Authentication and message integrity mechanisms are desperately needed. Generic security protocols operating bellow application layer (ipsec, TSL) are likely to be used for this purpose. Though light-weighted protocol security, e.g., authentication by IP address and no message integrity, could be sufficient in networks configured spoofing-proof, FCP-enabled products should be build for general conditions and have support for authentication and message integrity.

It can never stressed be enough that protocol security is a necessary but not satisfactory condition for security of the entire system. Naïve software implementations and careless administration have been proven to be the major source of security gaps.

Last but not least, firewalls should not blindly trust what they are told by FCP clients. Even trusted, administrator-maintained signaling servers may include bugs that result in FCP request compromising system's security. Thus we recommend that FCP-enabled firewalls also maintain Access Control Lists (ACLs) against which incoming FCP requests are checked. For example, such an ACL may permit an authenticated SIP proxy to open only pinholes from and to a phone pool and forbid port numbers below 1024.

7th. Performance and Scalability Considerations

The performance of the Middlebox architecture is primarily determined by the fact it relies on per-flow processing. The penalty is increased packet processing overhead that grows with number of rules and affects packet latency. Dynamic rule control may result in a considerable amount of rapidly changing rules (Four rules are needed for a typical telephony session: RTP and RTCP rules in both directions.) and disqualify packet classification algorithms relying on the ability to pre-build well-performing rule bases.

There are two properties of dynamic firewall control that may be exploited to improve performance. The first is that order of subsequent firewall holes (i.e., rules that result in identical action) may be arbitrarily shuffled without changing security semantics of the rule base. The other feature is the holes may be associated with additional attributes gained from applications. These attributes may be used to build an efficient rule-base.

For example, if an application aware of what codecs are used communicates expected bandwidth along with rules to a firewall, the firewall may place high-bandwidth rules in front of low-bandwidth rules. Though such a method is not very fair to bandwidth-saving codecs, it results in lower average packet-processing delay as most frequently hit rules are processed first.

Optimization of dynamically created rule-bases remains subject to further work.

8th. Miscellaneous Issues

There are further more or less serious design issues that we do not describe in this paper. Interested readers may find discussion of them in [23]. The issues include dealing with multiple middleboxes, packet fragmentation, controller failures, etc. Discussion of related topic also takes place on the mailing list of the IETF midcom working group[9].

9th. SIP-specific Issues

There are a couple of SIP-specific issues that have to be considered when using FCP. In particular, outbound proxies, funnel pinholes and FCP failure handling.

In general, SIP callers can send signaling requests directly to destination. Unfortunately, if callers behind NATs or firewalls do so their signaling would miss local SIP server controlling a firewall/NAT. Callers wanting to traverse a middlebox must configure their User Agents to route signaling via an outbound proxy. They may learn address of the proxy using DHCP[16] or another configuration protocol.

Another issue is SDP payload conveyed in SIP messages indicates only destination addresses. Firewall rules that can be generated using this knowledge resemble funnels in shape as they allow arbitrary senders to inject packets to a permitted destination. However, implementers of FCP-enabled proxies who fear such rules are not restrictive enough may use source guessing and limit source addresses to IP addresses at which call peers listen. Though this method is not general it is likely to cover a majority of telephony scenarios.

There is also a question of validity of rules over time. It seems desirable to use soft-state techniques for rules to prevent accumulation of state under erroneous conditions. Especially with firewalls, a failure to release rules might result in a relaxed filtering policy. It should be however noted that straightforward implementation of a keep-alive mechanism in a SIP proxy may result in stale state as well. The reason is that session state maintained in a SIP proxy and used to refresh middlebox state may itself become stale if a call party fails to terminate session properly. To deal with this issue, using SIP-session soft-state [17] and propagating SIP keep-alive messages to middleboxes would be needed.

Other tricky issues implementers of FCP-enabled SIP proxies have to keep in mind are re-INVITEs that may result in changed middlebox rules and early media [18] that is sent and need to traverse middleboxes before session establishment completes.

E. Related Work

IP telephony's need for a firewall/NAT control interface has been recognized and documented by Rosenberg[11] and Shore [19]. Molitor proposed a firewall control API [20] and Srisuresh proposed a NAT API [21]. An early attempt to use a protocol for this purpose was made by Huitema [22].

We presented an initial framework and set of requirements for a control protocol [23] at 47th IETF meeting. Since that time, forming a new working group has been discussed. After long (and somewhat controversial) discussion a new working group 'midcom' was set up. At the time of submission of this paper, the first working group documents were released.

Besides these standardization efforts, there are proprietary solutions. Authors are aware of solutions by Aravox and Marconi. The Aravox protocol was submitted to ETSI.

[9] Midcom WG:
http://www.ietf.org/html.charters/midcom-charter.html

Solutions for firewall/NAT control from end-devices have also been proposed. RSIP[12] deals with NAT. Roedig suggested using RSVP to deal with firewalls in [24]. Neither solution requires rebuilding applications and they are both closer to the end-to-end model. On the other hand, they require TCP/IP stack in end-devices to be rebuilt. We are also unsure if it is a good idea to deal with "evil middleboxes" by adding explicit compatibility protocols to end-devices.

Early SIP and H.323 ALGs start to appear in some firewall products. (Both SIP[25] and H.323[10] NAT ALGs are available for Linux.)

Rosenberg and Schulzrinne, frustrated by lack of deployable solutions and disappointed users being inhibited from enjoying Internet telephony, suggested tunneling Internet telephony protocols through https ports in [26]. This suggestion belongs to the class of policy subversion mechanisms. These mechanisms attempt to transport application protocols through channels explicitly permitted by firewall policy.

Nonetheless, we expect that easiness of deployment will be the factor to decide which solution family will be used. We anticipate operators of access and corporate networks will be attracted by solutions requiring minimum changes to their infrastructures. From this point of view, embedded ALGs and external FCP-enabled ALGs seem to have greatest chance to get deployed.

F. CONCLUSIONS

We described how firewalls and NATs break communication between Internet telephony users and reviewed solution space. The solutions fall into the following categories: junking firewalls and NATs, embedding ALGs in firewalls/NATs, controlling them from end-devices and controlling them from proxies.

We advocated the last approach, as it seems to be the easiest one from the deployment point of view. It requires only change to firewalls/NATs and signaling servers; end-devices remain completely untouched.

We know that it is a horrible, horrible hack. Linking application to transport/IP layers anywhere else than in end-devices breaks the Internet end-to-end model. However, it should be noted that this link was already presented by ALG patchwork and our primary ambition is to make ALGs easier to live with rather than to migrate to a radically different solution.

The other alternatives do not seem be better off. The junking approach is unrealistic. Embedded ALGs put device operators on mercy of vendors. End-device controlled firewalls constitute a considerable change in security model and require updates of end-devices.

The FCP approach also features other advantages. It is likely to perform better and result in lower development costs. Additionally, it allows for hop-by-hop signaling security.

In spite of these apparent advantages, we expect embedded ALGs to dominate in the early stages. There is currently no standardized FCP solution and middlebox traversal needs to be solved rapidly. In the long run, we anticipate wide support for FCP. Feedback from vendors indicates frustration from having to implement every new application stack in their devices. This frustration will be most likely the major driving force behind adoption of the FCP construct.

[10] H.323 ALG: *http://www.coritel.it/coritel/ip/sofia/nat/nat.html*

G. BIBLIOGRAPHY

1 M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg: "SIP: Session Initiation Protocol", RFC 2543, IETF, March 1999.

2 ITU-T Recommendation H.323. "Packet-based Multimedia Communications Systems," 1998.

3 S. Bellovin: "Distributed Firewalls", appeared in the November 1999 issue of "login", pp. 37-39.

4 P. Srisuresh, K. Egevang: "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, IETF, January 2001.

5 G. Tsirtsis, P. Srisuresh: "Network Address Translation - Protocol Translation (NAT-PT)", RFC 2766, IETF, February 2000.

6 M. Holdrege, P. Srisuresh: " Protocol Complications with the IP Network Address Translator", RFC 3027, IETF, January 2001.

7 D. Senie: " NAT Friendly Application Design Guidelines", Internet Draft, IETF, July 2000. Work in progress.

8 T. Hain: "Architectural Implications of NAT", RFC 2993, IETF, November 2000.

9 B. Carpenter: "Internet Transparency", RFC 2775, IETF, February 2000.

10 M. Leech: " SOCKS Protocol Version 5", RFC 1928, IETF, March 1996.

11 J. Rosenberg, D. Drew, H. Schulzrinne: "Getting SIP through Firewalls and NATs", Expired Internet Draft, Internet Engineering Task Force, Feb. 2000. Work in progress.

12 M. Borella et al: " Realm Specific IP: Framework", Internet Draft, IETF, July 2000. Work in progress.

8

13 B. Carpenter: "Middle boxes: taxonomy and issues", Internet Draft, IETF, January 2001. Work in progress.

14 J.H. Saltzer, D.P. Reed, D.D. Clark: " End-To-End Arguments In System Design", ACM Transactions on Computer Systems, pages 277288, 1984."

15 E. Lear: "A Middle Box Architectural Framework", Internet Draft, IETF, January 2001. Work in progress.

16 G.Nair, H.Schulzrinne: "DHCP Option for SIP Servers", Internet Draft, IETF, January 2001.

17 J. Rosenberg, S. Donovan: "The SIP Session Timer", Internet Draft, IETF, November 2000. Work in progress.

18 J. Rosenberg et al: "SIP 183 Session Progress Message", Expired Internet Draft, IETF, October 1999. Work in progress.

19 M. Shore: "H.323 and Firewalls: Problem Statement and Solution Framework", Expired Internet Engineering Task Force, Feb. 2000. Work in progress.

20 S. Mercer, A. Molitor, M. Hurry, T. Ngo: "H.323 Firewall Control Interface", Expired Internet Draft, IETF, November 1998. Work in progress.

21 P. Srisuresh: " Framework for interfacing with Network Address Translator", IETF, November 2000, Work in progress.

22 C. Huitema, F. Andreasen: "Media Gateway Control Protocol (MGCP) Support for Packet Relays", Expired Internet Draft, IETF, February 1999. Work in progress.

23 J. Kuthan, J. Rosenberg: "Middlebox Communication: Framework and Requirements", Internet Draft, Internet Engineering Task Force, November 2000. Work in progress.

24 U. Roedig, M. Goertz, M. Karsten, R. Steinmetz: "RSVP as Firewall Signaling Protocol", 2000.

25 B. Biggs: "A SIP Application Level Gateway for Network Address Translation", Expired Internet Draft, IETF, March 2000, Work in progress.

26 J.Rosenberg,H.Schulzrinne: "SIP Traversal through Residential and Enterprise NATs and Firewalls", Internet Draft, IETF, November 2000. Work in progress.

Potpourri

ECLIPSE Feature Logic Analysis

Gregory W. Bond, Franjo Ivančić, Nils Klarlund, Richard Trefler
{bond, trefler, klarlund}@research.att.com
ivancic@gradient.cis.upenn.edu

Abstract—**ECLIPSE is a virtual telecommunications network based on IP. It is the result of an ongoing research project at AT&T Labs – Research that is investigating next-generation telecom service architectures. The ECLIPSE Statecharts language was developed to simplify feature (service) development, for example call waiting, by supporting a smooth transition from design to implementation and by supporting automated semantic analysis. The modular nature of ECLIPSE features necessitates that they utilize well-defined protocols for communicating with one another. If an individual feature fails to obey the protocol then it is likely that subscribers to the feature will be unable to complete calls. This paper describes a tool that uses the Mocha model checking tool to analyze ECLIPSE feature modules to ensure that they satisfy the specified protocols.**

Keywords— **DFC, Distributed Feature Composition, telecom services, voice over IP, VoIP, UML Statecharts, Mocha, model checking, Java**

I. Introduction

ECLIPSE is a virtual telecommunications network based on IP. It is the result of an ongoing research project at AT&T Labs – Research that is investigating next-generation telecom service architectures. The ECLIPSE network is intended to support multimedia telecommunication services involving voice, video, and text in seamless composition. The ECLIPSE network is designed to be device-independent to accommodate today's range of soft and hard devices such as cable phones, Microsoft Netmeeting™, AOL Instant Messenger™, as well as multiple external networks such as the public switched telephone network (PSTN). ECLIPSE provides a framework for rapid development and deployment of telecom services. It also provides a framework for managing "feature interaction," a problem that has hampered customization and rapid innovation of services in traditional telephony. ECLIPSE is an instance of Jackson and Zave's Distributed Feature Composition (DFC) virtual architecture [1]. DFC provides a framework for exposing and managing feature (service) interactions in multi-party, multi-feature (service) and multi-media "calls" in telecom networks. ECLIPSE implements DFC in an IP setting.

The ECLIPSE Statecharts language, hereafter referred to as "ECLIPSE Statecharts", was developed to meet the needs of ECLIPSE feature developers. Before ECLIPSE Statecharts was developed, ECLIPSE features were implemented using a general programming language (Java). It became clear early on that using a general programming language was inadequate for this purpose since it was easy to introduce faults into the feature logic, even for the simplest of features. For example, developers would forget to account for possible feature states, they would neglect to account for messages that might be received from the feature's environment, and they would respond incorrectly to messages received from the environment. For a more complicated feature like call waiting, which involves multiple parties, the problems were worse since the number of states and possible interleavings of messages exchanged with the environment were much greater. ECLIPSE Statecharts was designed to address these problems.

G.W. Bond, N. Klarlund and R. Trefler are located at AT&T Labs – Research, Florham Park, NJ, USA; F. Ivančić is located at University of Pennsylvania, Philadelphia, PA, USA

ECLIPSE Statecharts is a customized version of the Unified Modeling Language (UML) Statecharts behavioral description language [2], [3]. The UML Statecharts language, hereafter referred to as "UML Statecharts", is a graphical language for describing hierarchically structured state machines. Since it is a graphical language based on state machines it is well suited for describing the high level behavior of system structures. The language supports hierarchically structured state machines so it is possible to describe complex behavior with simple diagrams. The language also supports a number of concepts that are useful for describing timed, reactive systems, for example concurrent state machines, timed transitions, and a number of inter-object and inter-state machine communication mechanisms. In addition to being a powerful behavioral description language, UML Statecharts is part of the Object Modeling Group's UML standard for object-oriented system modeling. For this reason, a growing number of tools are available or in development to support the language.

As a design language UML Statecharts might have sufficed for describing the high level behavior of ECLIPSE features. However, by incorporating a number of ECLIPSE concepts into the language, it is possible to formally translate an ECLIPSE feature design to an implementation. Indeed, experience has shown that a feature described with ECLIPSE Statecharts needs very few additional implementation details.

Since ECLIPSE Statecharts are based on finite state machines they are suitable for automated analysis. The ability to analyze ECLIPSE feature logic is desirable for a number of reasons. A consequence of the underlying architecture of ECLIPSE is that the failure of any feature module involved in a call ("feature box" in DFC) can cause the entire call to fail. Moreover, since the ECLIPSE architecture is open we expect third-parties to develop features for use in ECLIPSE networks. For these reasons it is important to ensure that each feature module deployed in an ECLIPSE network satisfy certain minimal integrity constraints.

One way to ensure that these constraints are met is to use run-time monitoring of individual features. This approach, which is still used in parts of the current ECLIPSE network, imposes run-time overhead which we would prefer to avoid. A complementary approach is to analyze feature logic prior to its deployment in the ECLIPSE network. Using this approach, features that satisfy the constraints no longer require run-time monitoring.

In the current ECLIPSE system, ECLIPSE Statecharts exist as a set of Java classes (i.e. state classes, transition classes and an interpreter class). The Java compiler is used to perform syntax checking and type checking. However, the compiler cannot

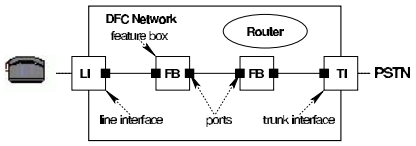


Fig. 1
A DFC NETWORK.

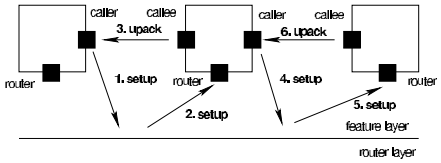


Fig. 2
CONSTRUCTING A USAGE IN DFC.

detect domain-specific semantic errors. To do this we have developed a tool to perform simple static analysis functions similar to the C language's lint tool, as well as to perform more complex model checking tasks on the code to ensure that a feature interacts correctly with its environment. The actual model checking task is performed using the Mocha model checking tool [4], [5].

II. DFC

The DFC architecture is an instance of the "pipes and filters" architectural design pattern. As shown in Figure 1, a DFC network consists of instances of a small number of component classes: Line Interface (LI) boxes which connect a single device to a DFC network, for example, a black phone; Trunk Interface (TI) boxes which connect another network to a DFC network e.g. the PSTN; Feature boxes which implement feature logic e.g. call waiting.

When a call is initiated from an LI or TI box, the router finds the destination LI or TI and then finds the feature boxes that are to be inserted based on user subscription data and precedence rules. Figure 2 shows how boxes establish connections between each other by exchanging DFC messages between peer ports according to a protocol defined as part of DFC. The overall graph of boxes that is constructed over the course of a call is called a usage.

III. INTER-PORT MESSAGING

In ECLIPSE, the behavior of an individual feature box is defined using an ECLIPSE Statechart. A feature box can communicate with its environment only via its ports which are connected to ports on peer boxes. A box's Statechart defines how the box reacts to messages it receives on its ports. The actions performed by a box in response to a message may include sending messages out its ports. Message exchange between peer ports is asynchronous and each port has its own message queue for incoming messages. This form of messaging is a refinement of one form of messaging specified by UML Statecharts.

In ECLIPSE Statecharts, as in UML Statecharts, transitions have labels of the form: $event[guard]/action$, where each label component is optional. Events are message receive operations on a port, guards are arbitrary boolean expressions, and actions are arbitrary expressions, which often include send operations on ports. A transition is enabled (fireable) if there is a message available in the specified port's queue and the guard evaluates to true. In ECLIPSE Statecharts we use the following notational short forms (borrowed from the CSP language[6]): $port!message$ to send, and $port?message$ to receive.

For the reader familiar with UML Statecharts, you should note that, unlike UML Statecharts where each object possesses a single queue for incoming asynchronous messages, ECLIPSE feature boxes potentially possess multiple queues: one for each port associated with the box.

IV. PORT PROTOCOLS

In order to support feature logic modularity in the context of a pipes and filters architecture, DFC requires that box ports obey well-defined protocols. These protocols ensure that a box can insert itself into a usage as it is being constructed, and remove itself from a usage when the usage is torn down. Once a box is inserted into a usage, a box is able to effect changes on the signaling and media associated with the "call" via its port connections.

There are four classes of box ports defined by DFC: router ports, which receive messages from a router; caller ports, which are only able to initiate connections to peer boxes; callee ports, which are only able to receive connections from peer boxes; and dual ports, which can behave as either a caller or callee port for the lifetime of a connection with a peer box.

A box programmer is responsible for ensuring that these protocols are correctly implemented for each port employed by a box. Typically a box uses a number of ports. For example, the ECLIPSE Statechart defining the feature logic for the call waiting feature, shown in Figure 3, employs 4 ports: a router port (labeled 'box'), two dual ports (labeled 'dual1' and 'dual2') and a callee port (labeled 'callee'). This Statechart utilizes the UML Statecharts notions of nested state machines and history pseudostate, as well as semantic refinements to UML Statecharts involving transition priority based on message class and nesting level, and port aliases. Port aliases permit indirectly specifying the identity of a port, analagous to how a pointer indirectly specifies the identity of a variable. Three port aliases are used in the call waiting feature: 'sub', 'conn', and 'wait'. These aliases are used to refer to the roles that actual ports play at any time during the feature's execution. For example, the roles of the ports representing the connected participant ('conn') and the waiting participant ('wait') are exchanged when the subscriber ('sub') signals the feature with a flash-hook.

V. FEATURE LOGIC ANALYSIS

Feature logic analysis addresses the following two questions:

- Did the programmer of the feature box consider all possible input messages that the environment—the peers associated with the feature box—can send to the feature box?
- Does the feature box output only those messages to its environmental peers that are expected by those peers?

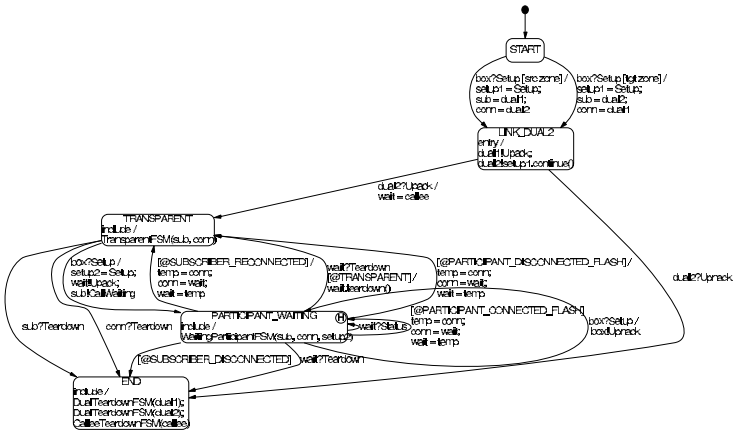


Fig. 3
THE CALL WAITING FEATURE LOGIC

Performing this analysis is a two-step process. The first step consists of translating the feature logic expressed as ECLIPSE Statecharts code into a model suitable for use by the Mocha model checking tool. The analysis of the model using Mocha is performed in the second step.

Model checking ECLIPSE feature code takes place in a test environment set up by ECLIPSE2Mocha within the modeling language framework of Mocha, called Reactive Modules (RM). That is, given a feature box B, ECLIPSE2Mocha translates B into RM and combines B with the RM versions of the standardized environmental peer entities with which B expects to communicate. Finally, ECLIPSE2Mocha adds a distinguished *bad* state to the RM model of B and its peers. The RM test environment behaves exactly like B combined with its peers except in the case when either B sends a message to a peer which the peer cannot accept or a peer sends a message to B which B cannot accept. In either case the test environment transits to the *bad* state.

Model-checking then consists of checking whether there is an execution of the test environment from the initial state of B to the *bad* state. This check can be easily embedded in the temporal logic which Mocha uses to evaluate RM models.

VI. TRANSLATION

The translation of ECLIPSE Statecharts feature logic code to a RM model consists of the series of steps shown in Figure 4.

The feature logic code – written in a subset of Java – is first parsed. The next step identifies the Eclipse Statecharts instructions, such as `addState` or `addTransition` and produces an abstract model of the code expressed as a hierarchical state machine. The hierarchical state machine is flattened, and then port aliases are instantiated. Some preliminary checks are performed on the resulting model and then a model is output in RM. These steps are explained in more detail in the following sections.

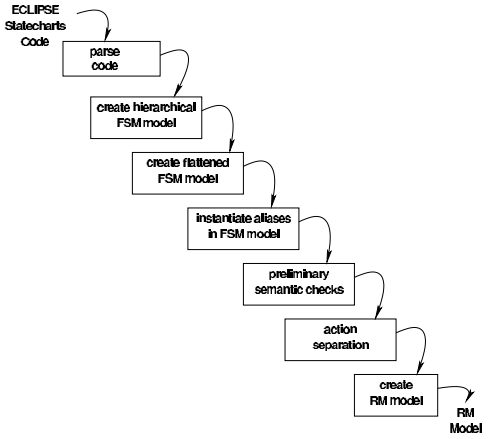


Fig. 4
TRANSLATION OF ECLIPSE FEATURE LOGIC TO A MOCHA MODEL

A. Parsing and Creating the Hierarchical FSM Model

The ECLIPSE Statecharts language is implemented in Java. To define an ECLIPSE Statechart a programmer subclasses the main FSM class and, in this class's constructor, creates instances of state and transition objects and invokes methods to add them to the FSM. The parent FSM class, the transition classes and the state classes comprise the ECLIPSE Statecharts language and its interpreter.

Programmers can define action methods for transition or state instances that will be executed by the FSM interpreter when a transition fires or when a state is entered or exited. Similarly,

programmers can define guard methods for transitions that will be invoked by the FSM interpreter to determine if a transition is enabled.

The parsing step parses the Java code that defines an ECLIPSE Statechart. The parser grammar is customized to recognize the declarations of transitions, states, and their associated action or guard methods, as well as declarations of ports associated with the box. The resulting parse tree contains the elements of the Java code that are necessary for constructing the hierarchical FSM model.

B. Flattening

Since Mocha's RM language does not support the notion of hierarchical state machines, these are flattened to simplify translation to the RM model. Also, the semantics of Eclipse Statecharts are easily expressed in the flattened state machine. This is particularly true for the transition priority rules used in Eclipse Statecharts. Another reason for favoring a flattened state machine is that checking other properties not checked by Mocha, like "Are all possible status messages in this state covered?", is much easier in a flattened state machine.

It should be noted that the flattening phase does not increase the number of states in the state machine. On the contrary, it may actually reduce the number of states in the state machine. However, the flattening phase normally increases the number of transitions. Furthermore, the number of states will increase exponentially relative the to original program size when state machines are hierarchically nested.

C. Instantiation

Eclipse Statecharts permit the use of port aliases—that is, variables that range over the ports of the feature box. Mocha's RM does not directly support variable aliasing so the instantiation step explicitly instantiates occurrences of port aliases with their possible values. Instantiation results in adding conditions to transition guards. In general, instantiation will also increase the number of transitions.

D. Preliminary Semantic Checks

In this step we check certain semantic properties that are easily checked in the flattened state machine. Currently, we are checking whether a state that accepts a specific status message on a given port also takes care of all other possible status messages on that port. This is a nice by-product of flattening the state machine, because certain properties can be checked easily in a state-by-state fashion.

E. Action Separation

The RM model is not able to directly express the case of a feature box sending more than one message to the same peer during one transition. If such behavior is detected, the sending actions in the same transition are separated from each other by introducing a so-called micro-state μ. By introducing a sequence of micro-states, each with exactly one incoming transition and one outgoing transition, we handle the fact of sending a sequence of messages to the same peer. So, for example, if an action sends n messages to the same peer, we introduce $n-1$ new micro-states $\mu_i, i=1,\ldots,n-1$.

VII. Creating the Reactive Modules Model

The final translation step shown in Figure 4 creates the RM model for the Mocha model checker. In order to understand the mapping from the flattened FSM model to the RM model, it is necessary to provide some background information on Mocha and the RM language itself.

A. The Model Checker Mocha

Model checking is emerging as a practical tool for automated debugging of embedded software. In model checking, a high-level description of a system is compared against a logical correctness requirement to discover inconsistencies. Since model checking is based on exhaustive state-space exploration, and the size of the state space of a design grows exponentially with the size of the description, scalability remains a challenge. The model checker Mocha is based on the idea of exploiting modular design structure during model checking. Instead of manipulating unstructured state-transition graphs, it supports the hierarchical modeling framework of Reactive Modules.

The language Reactive Modules is a modeling and analysis language for heterogeneous concurrent systems with synchronous and asynchronous components. This is accomplished by the notion of time rounds. As a modeling language it supports high-level, partial system descriptions, rapid prototyping, and simulation. As an analysis language it allows the specification of requirements either in temporal logic or as abstract modules. Finally, as a language for concurrent systems, it allows a modular description of the interactions among the components of a system.

The behavior (executions) of a reactive system can be visualized in a message sequence charts (MSC) like fashion by using the simulator. To run the simulator, the user selects a module and the submodules/variables to be traced. For each selected variable, a vertical line shows its evolution in time. The value of a variable is displayed only when it changes. The same format is used to display the counter-examples generated by the model checkers during failed verification attempts. The simulator can be used either in automatic or in manual mode.

Mocha allows the specification of requirements in a rich temporal logic called alternating temporal logic (ATL). By far the most common requirements are invariants, and thus it is of utmost importance to implement invariant checking efficiently. With this in mind, Mocha provides both fine-tuned enumerative and symbolic state search routines for invariant checking.

B. The RM Model

The flattened state machine is translated into a single RM module. The environmental peers are instances of predefined RM modules. The combination of these RM modules constitutes the RM model that is used for model checking.

The operational semantics of the state machine are explicitly translated into the RM model. The communication between a box's Statechart and its environmental peers is accomplished using the following sub-round structure of one RM time round:

1. First a peer is chosen to send at most one message and update its internal state.

2. The feature box model will receive either no message or exactly one message from one of its peers. If it does not receive a message, it will not do anything. If it receives a message, it will update its internal state, and it might also send messages to its peers. Note that the use of micro-states constrains a feature box model to send at most one message to any peer.
3. The peers will receive the messages from the feature box. If a peer receives a message from the feature box, it updates its internal state.

Whenever a peer receives a message that it does not have an explicit transition for, the peer model fires an implicit transition into a special "bad state", that indicates the feature box is incorrect. Similarly, whenever the box model receives a message that it does not have a transition for, the box model fires an implicit transition into a bad state.

The approach to modeling we have adopted avoids explicitly modeling the queues between the feature box and its environment. Therefore, a message that is sent from a peer (and potentially changes the internal state of the peer) has to be handled immediately by the feature box model. In reality a feature box might actually enqueue a peer message for a while before it looks at it. So to avoid flagging an error in the feature box because it received a message that it was not expecting at this point (the programmer merely decided not to bother with this peer in this particular state), we have to ensure that the peer does not send this message in the first place. Therefore, we have introduced enabling flags that signal a peer whether the feature box model is ready to accept any message the peer might want to send at a given time.

We can avoid modeling the queues involved in the real system by modeling them implicitly in the environment of the feature box. The peers are allowed to skip a round where they are supposed to send a message, which basically models the fact that the previous message has not arrived at the peer yet. The possibility to skip a message can also be interpreted as a delay of the message from the peer to the feature box. After careful consideration of the environmental models it is clear that all possible message sequences that the environment in the original setting might send can be sent in the RM model.

The last step of our translation is the output of a RM model of the feature box. The peers have predefined models that are based on the state machines shown in Figures 5, 6, 7, and 8. If a feature box sends out instances of status message subclasses, rather than just instances of the parent status message class, it is necessary to add transitions into the peer models. Consider the case that a feature box sends out the status messages subclass $m_1, \ldots, m_n$ to its peers. For each transition in the peer model that is labeled with "`?status`", we will include a transition for each message $m_{i,i=1,\ldots,n}$ with the label "$?m_i$".

As mentioned before, we translate the flattened state machine as one RM module. The RM model maintains a variable called `currentState` that ranges over all states, and keeps track of the state that the flattened state machine is in. Each transition is translated to one update rule in the model. An additional rule covers the case that no message arrives from the environment in a time round. This rule ensures that the state of the state machine does not change. If a message arrives, but no update rule is applicable, then we will enter a "dead state" and flag

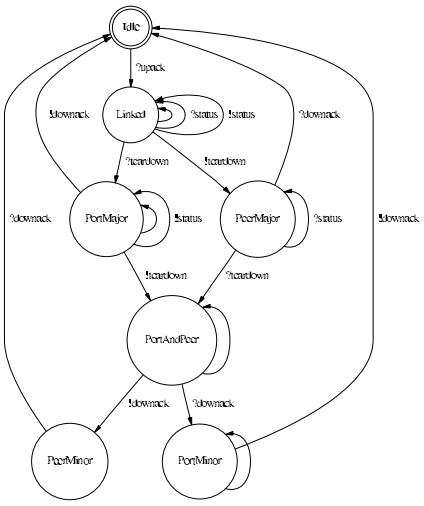


Fig. 5
DFC CALLEE PORT PEER PROTOCOL

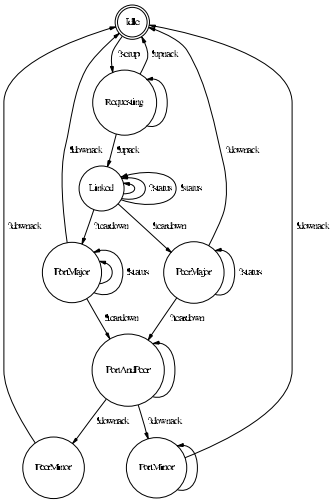


Fig. 6
DFC CALLER PORT PEER PROTOCOL

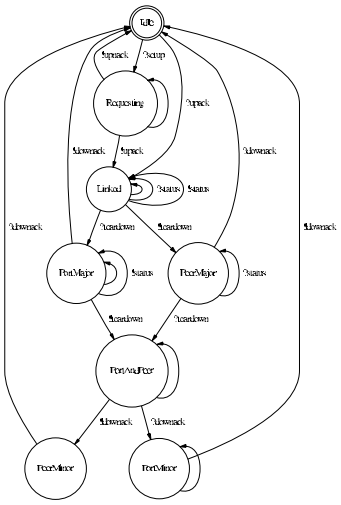


Fig. 7
DFC DUAL PORT PEER PROTOCOL

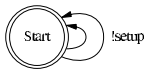


Fig. 8
DFC ROUTER PORT PEER PROTOCOL

an error. This covers one of the problems we are looking for, because it basically means that the programmer of the feature box has forgotten to take account for this particular message.

Figure 9 shows a sample transition in the flattened state machine. Its source state is A, and its destination is state B. It is enabled if the guard $alias = p \wedge g$ is true, and if the message m from the peer of port p arrives at the feature box. If this transition is taken, then the actions $p_1!m_1$, $p_2!m_2$, $alias = p_3$, are executed.

We translate this transition into RM in the following manner:

```
currentState = A & signalFromP? &
messageFromP' = m & alias = p & g ->
signalToP1!  ; messageToP1' := m1 ;
signalToP2!  ; messageToP2' := m2 ;
alias' := p3 ;
currentState' := B ;
```

When we send a message to a peer, we update the corresponding value, but we also have to make sure that it is realized that we

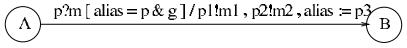


Fig. 9
TRANSITION FROM A TO B IN THE FLATTENED STATE MACHINE

updated the value. We therefore issue a Mocha event by saying `signalToP1!`. To check whether there has been a message send from P in this round, we check the corresponding event by `signalFromP?`. To receive the message that was sent in this round, we have to ask for `messageFromP'` instead of `messageFromP`, which holds the value of the previous time round.

VIII. MODEL-CHECKING ECLIPSE STATECHARTS

In Figure 10, we give an example of an ECLIPSE Statecharts feature box we have analyzed. This feature behaves like a buffer after it has been set up between a left and right neighboring box. Upon proper initialization according to the DFC protocol, the box is in state 'linked', where it reads messages from its right hand neighbor on the calleePort and sends them to its left hand neighbor on it callerPort and vice versa. The states 'linked', 'transparent1' and 'transparent2' explain this behavior. Here, we have assumed that the messages are atomic; in reality, the messages contain values that are temporarily stored in the feature box. The remaining states are necessitated by the DFC protocol.

If Mocha does determine that the state *bad* is reachable from the initial state, a debugging mechanism in Mocha is available to reconstruct the sequence of events leading to the bad state. In Figure 11 we show a screen shot of Mocha displaying such a trace. The error was generated by altering the program in Figure 10 so that the feature box sends two consecutive teardown messages: we changed the 'unlink5' to 'unlink6' transition so that a teardown message is placed on the callerPort instead of a downack. The trace, which is shown only partially, reflects that error by taking a path in the protocol that produces two teardown messages.

During the programming in Java of this trivial feature box, we introduced several little errors as typically happens, mostly due to misspellings. All but one were caught by the parser of the ECLIPSE2Mocha tool. (Some would also have been caught by the Java compiler.) The one that was not caught was discovered through model checking. The model checker approved of the Java code, but even a positive answer must be taken with a grain of salt. For example, in our setting the model checker does not check for liveness properties like "does the feature box always eventually acknowledge a teardown request?". Thus, it is a reasonable sanity check to willfully introduce errors in the feature box program that is purported to be correct. When we did this, we discovered that the program sometimes, unexpectedly, still was passed by the model checker. As a result, we discovered a misspelling of a method name that issues a message to a port. This Java error would not have been caught by a compiler since the erroneous name appeared in the initializer for an object of an anonymous, inner class.

When we originally programmed the call waiting box in Figure 3, we struggled with three insidious programming errors, all of which we later presented to our tool. They were all correctly identified through error traces.

A. Correctness of analysis

To give a complete account of what the correctness of our analysis is would be a huge task. For example, we would need a formal description of the semantics of ECLIPSE Statecharts,

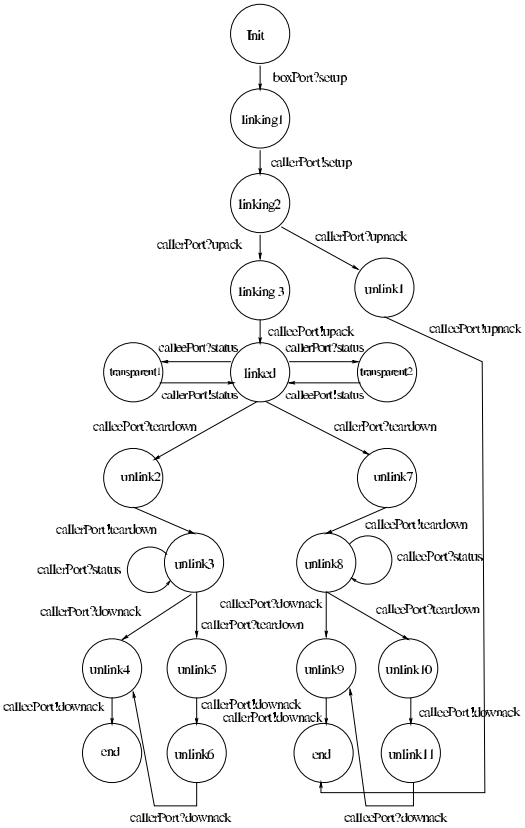


Fig. 10
TRANSPARENT FEATURE BOX

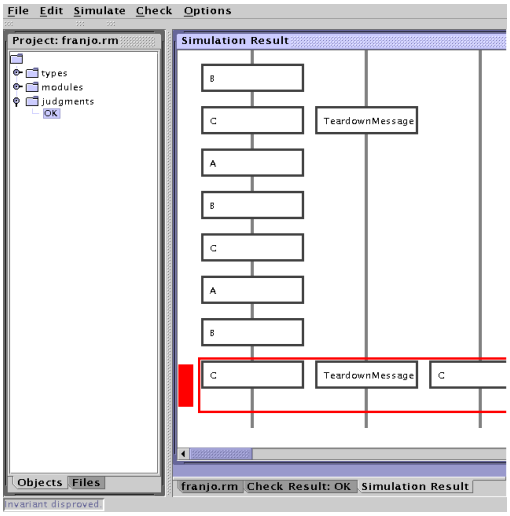


Fig. 11
MOCHA ERROR TRACE

the semantics of the translation, and the semantics of the Mocha language. Also, we would need to carefully explain the abstractions that are inherent to our analysis. Instead, we will give an informal statement that reflects our belief that we have correctly implemented the ECLIPSE Statecharts semantics through the translation to Mocha. Thus, we will have to relate errors found during runtime to errors discovered by our tool. Our concept of error is that of the Section V: an error occurs if either the environment or the feature box is unable to process a message. We say that there is an error in the ECLIPSE Statecharts feature if there exists an environment that follows the peer protocols and for which the composite system may enter a situation where a port is unable to process a message. Note that such a situation is characterized by a trace (history) of communication events. In general, traces involve buffering of messages—something that our Mocha model does not accommodate. Therefore, we say that a trace is synchronous if the event following a send message is the accept of the message. Moreover, we assume that the only communication events that involve the feature box are those that are recognized as such by the ECLIPSE Statecharts parser (thus, communication events invoked through auxiliary method definitions are not allowed). Also, we assume that all Boolean guards on message transitions are true. Then, we believe the following to be true.

- (Soundness) Any error found in the ECLIPSE test environment by the model checking procedure is an error in the ECLIPSE Statecharts feature.
- (Completeness under synchronization assumption) Assume that an ECLIPSE Statecharts feature exhibits an error in a synchronous trace. Then, the ECLIPSE Statecharts will not pass the model checking procedure.

In a more advanced tool, it would sometimes be relatively straightforward to analyze Boolean guards if they involve local variables. The issue of analyzing queued system is generally undecidable, since queues tend to resemble tapes of Turing machines.

IX. RELATED WORK

Research is currently very active and diverse in the area of model checking Statecharts. Space does not permit us to provide a comprehensive overview of this activity. Instead, we will address the current work most closely related to our own. Similar to our own work, [7], [8], [9], [10] have developed approaches to model checking properties of systems defined in various Statechart dialects. Only one of these approaches ([9]) addresses queued, asynchronous messaging between Statechart objects—similar to the way that an ECLIPSE feature box interacts with its environmental peers. However, in their approach they assume bounded queues between the environment and an object.

Instead of arbitrarily bounding queue length, our approach abstracts away the queues by exploiting properties of the environmental peer protocols and the semantics of the Mocha RM modeling language. Although the approach we use is not general enough to be applied to all possible environmental behavior, it is suitable for the environmental behavior defined by the DFC architecture.

In practice, the customized parser that we built for Java programs turned out to be very useful by itself for writing ECLIPSE Statecharts. The parsing step has revealed domain-specific semantic errors in finite state machines that the Java compiler deems to be error-free. Moreover, our experience with this tool validates the use of statically-checked constraints that formalizes software architectures. Several general tools for expressing such architectural constraints on code have been proposed; see [11] for references and the description of CoffeeStrainer, a tool for checking Java programs.

X. Conclusions and Future Work

We have built a tool, ECLIPSE2Mocha, for analyzing the communication behavior of an ECLIPSE feature and its immediate environment. ECLIPSE2Mocha is capable of detecting subtle semantic errors of ECLIPSE feature code and using the Mocha reporting features ECLIPSE2Mocha is well suited as a debugging aide for ECLIPSE features. Our translation of ECLIPSE feature code to RM code relies on a crucial abstraction – namely, the modeling of asynchronous communication via unbounded queues by synchronous communication. However, because we restrict the types of properties analyzed, errors detected by ECLIPSE2Mocha can be translated into errors of the ECLIPSE feature code.

For the future we see several interesting directions to take this work. Firstly, we see a need for an intermediate language between the Java code of ECLIPSE Statecharts and RM. Such an intermediate language would make the use of other analysis tools far easier and remove the need for a direct mapping between ECLIPSE Statecharts and RM. Secondly, we would like to explore the use of model checkers of hierarchical models [12] to avoid the flattening phase currently used by ECLIPSE2Mocha. Thirdly, we are interested in incorporating ECLIPSE2Mocha and the use of Mocha directly within a domain specific compiler for ECLIPSE Statecharts.

Finally, we would like to enhance the class of properties checked. This can be done by enlarging the type of environmental entities used in the analysis and by more faithfully modeling the unbounded queues and asynchronous communication of ECLIPSE. These enhancements would allow us to check significantly more feature interaction properties.

Acknowledgments

The authors would like to thank the other members of the ECLIPSE project at AT&T Labs – Research for their feedback during the development of ECLIPSE Statecharts and the analysis tool: Eric Cheung, Andrew Forrest, Michael Jackson, Hal Purdy, Chris Ramming, Xiaotao Wu (Columbia University) and Pamela Zave.

References

[1] Michael Jackson and Pamela Zave, "Distributed feature composition: a virtual architecture for telecommunications services," *IEEE Transactions on Software Engineering*, vol. 24, no. 10, pp. 831–847, Oct. 1998.

[2] David Harel and Eran Gery, "Executable object modelling with Statecharts," *IEEE Computer*, July 1997.

[3] Object Management Group, *OMG Unified Modeling Language Specification, version 1.3*, Object Management Group, June 1999, Available at `ftp://ftp.omg.org/pub/docs/ad/99-06-08.pdf`.

[4] Rajeev Alur, Thomas A. Henzinger, F.Y.C. Mang, Shaz Qadeer, Sriram K. Rajamani, and Serdar Tasiran, "Mocha: Modularity in model checking," in *Proceedings of the Tenth International Conference on Computer-aided Verification (CAV)*, 1998, number 1427 in Lecture Notes in Computer Science, pp. 521–525, Springer-Verlag.

[5] L. de Alfaro, R. Alur, R. Grosu, T. Henzinger, M. Kang, R. Majumdar, F. Mang, C. Meyer-Kirsch, and B.Y. Wang, *Mocha: Exploiting Modularity in Model Checking*, August 2000, Available at `http://www-cad.eecs.berkeley.edu/~mocha`.

[6] C.A.R. Hoare, *Communicating Sequential Processes*, Prentice-Hall, 1985.

[7] S. Gnesi, D. Latella, and M. Massink, "Model checking UML Statechart diagrams using JACK," in *Proceedings of the 4th IEEE International Symposium on High-Assurance Systems Engineering*, 1999, pp. 46–55.

[8] E. Mikk, Y. Lakhnech, M. Siegel, and G.J. Holzmann, "Implementing Statecharts in PROMELA/SPIN," in *Proceedings of the 2nd IEEE Workshop on Industrial Strength Formal Specification Techniques*, 1998, pp. 90–101.

[9] J. Lilius and I.P. Paltor, "vUML: a tool for verifying UML models," in *Proceedings of the 14th IEEE International Conference on Automated Software Engineering*, 1999, pp. 255–258.

[10] Chonlawit Banphawatthanarak and Bruce H. Krogh, "Verification of stateflow diagrams using SMV: sf2smv 2.0," Tech. Rep., Dept. of Electrical and Computer Engineering, Carnegie Mellon University, June 2000, Available at `http://www.ece.cmu.edu/~krogh`.

[11] B. Bokowski, "Statically-checked constraints on the definition and use of types in Java," in *Proceedings of ESEC/FSE'99*, 1999, vol. 1687 of *LNCS*, pp. 355–375.

[12] R. Alur and M. Yannakakis, "Model checking of hierarchical state machines," in *Proceedings of the Sixth ACM Symposium on the Foundations of Software Engineering*, 1998, pp. 175–188, Available at `http://www.cis.upenn.edu/~alur`.

Centralized Conferencing using SIP

Kundan Singh, Gautam Nair and Henning Schulzrinne
Columbia University
{kns10,gnair,hgs}@cs.columbia.edu

Abstract— **Multiparty conferencing is an important telephony service, provided in the PSTN by conference bridges. Internet telephony can enhance this basic service by video conferencing and collaborative work. The Session Initiation Protocol (SIP) can support many different conferencing architectures, including the centralized conferencing server model.**

We describe design issues and challenges in implementing a SIP-based centralized conferencing server and discuss the architecture and performance of our implementation, sipconf.

Keywords— **Centralized conference server; dial-in conference bridge; SIP; RTP mixer; packet audio; packet video; Internet telephony; sipconf**

I. Introduction

The Session Initiation Protocol (SIP) [1] defines how to establish, maintain and terminate Internet sessions including multimedia conferences. SIP supports various multi-party conferencing models [2], ranging from mixing in end systems to multicast conferences. When multicast is not available, centralized mixing, transcoding and filtering of media can be used to create multiparty conferences. In centralized mixing, a server receives media streams from all the participants in a conference, mixes or filters these based on pre-defined policy and distributes the streams to the participants. Different types of media streams need to be handled differently, for example, audio streams are typically summed, while video streams are selected, e.g., to present only the active speaker.

The main functions of a conference server is the mixing and redistribution of media streams. Typically, Internet audio streams are added ("mixed"), while video streams and other media are simply replicated. However, a video mixer can also create a new composite video image [3]. For audio, the server needs to ensure that a participant does not receive a copy of his own media in the mixed stream. RTP [4] allows a sender to indicate which sources have been combined in a single media packet. When summing, the server should absorb the jitter in packet arrival times while introducing minimum delay ("playout buffer").

For replication, the server should not need to be aware of the media formats. The RTP SSRC indication [5] ensures that the receiver can distinguish different sources addressed to the same network destination.

For either summing or replication, it is desirable if each participant can use different media types and packetization intervals, to accommodate heterogeneity of end systems and access bandwidths. Implementations need to scale to large numbers of conferences as well as large numbers of participants per conference.

A media mixing module with a SIP interface can act as a conferencing server component in the distributed application server (AS) component architecture [6]. Advanced system can bundle this functionality with other services, such as interactive voice response (IVR) and a web-based user interface.

This paper explores the centralized conference server design issues in detail and describes challenges in implementing such a system. We also explore advanced usage scenarios of the conferencing system in a real-world Internet telephony environment. We are currently collecting performance data on our implementation of the SIP based conference server, sipconf[1], and present some initial results of our experiments.

A. Outline of the rest of the paper

Section II explains the role of SIP in centralized conferencing systems. Section III discusses and compares different conferencing models. Design issues are described in Section IV. We provide an overview of our implementation and performance figures in Section V. The usage scenarios in various Internet telephony and multimedia communication applications are discussed in Section VI. Section VII lists some of the related work. Finally, we summarize and point to future work in Section VIII.

II. Background

Many PSTN carriers offer conference bridges which allow users to take part in a voice conference by dialing a telephone number and possibly access code. We can use the same concept for Internet-based conferencing: The conference can be identified by a destination address, and participants can join the conference by making a call to that address, thus requiring no modifications in end systems. There are currently two Internet telephony signaling protocols, IETF's SIP and ITU-T's H.323 [7]. SIP identifies the destination via a SIP URI of the form *sip:user@domain*, while H.323 uses AliasAddress data structures, which can assume many forms, including URLs.

There are two different aspects of Internet based conferencing, signaling and media. Either SIP or H.323 can be used as a signaling protocol for taking part in a conference. Both SIP and H.323 use the Real-time Transport Protocol (RTP [5]) for carrying real-time media traffic, such as audio and video. H.323 defines a multi-point control unit (MCU) for handling multiparty conferences. An MCU consists of a multi-point controller (MC), which can also be part of a terminal, to handle signaling and control exchanges with every participant in the conference. An optional component, the multi-point processor (MP), handles mixing and filtering of different media streams. SIP does not define any conferencing entity as such, as these entities are easily modeled as SIP user agents. The core SIP specification supports a variety of conferencing models [2]. In the server-based models, RTP media streams are mixed or filtered by the server and distributed to the participants. There is a standard point-to-point signaling relationship between each participant and the conferencing server.

[1]More information at http://www.cs.columbia.edu/~kns10/software/sipconf

The conference is identified by the SIP URI, e.g., `sip:discuss@server.com`. The standard user location and routing mechanisms in SIP forward all calls to the appropriate conference server at `server.com` without requiring any extension to the protocol. The SIP message routing entities (SIP proxies) need not be aware that the request URI corresponds to a conference and not to an individual person.

The Session Description Protocol (SDP [8]) is used to indicate media capabilities and media transport addresses. The participant sends the information about his media capabilities (PCMU) and the transport address where he wishes to receive RTP packets. In the message body of the 200 success response, the server sends the transport address to which the participant should send his PCMU RTP packets. More advanced scenarios can be accomplished using the SIP REFER method. For example, an existing participant can invite another user to join the conference. These conferencing models can be found in [2].

SIP-based authentication can be used to prevent unauthorized participants to join a conference. The server can support both pre-arranged conferences as well as ad-hoc conferences by assigning special meaning to the user field in the request URI. For example, participants who wish to join `sip:ietf.arranged@office.com` will need to set up the conference before hand, while those who wish to join `sip:library-discuss.adhoc@office.com` do not need to setup the conference in advance. The conference state is maintained as long as at least one participant is part of the conference. Participants find out about the conference URL via external means, such as email or a web page.

III. Conferencing models

Conference models can be distinguished based on the topology of signaling and media relationships. Conferences with a central server are easier to handle for end systems and simplify keeping track of the conference participants. On the other hand, network-layer multicast is more scalable for large-scale media distribution and allows a "loose" model of conference membership [9], where each member has only an approximate view of the group roster.

Table I summarizes the different types of *media distribution models* in multimedia conferencing. The table compares the scaling properties, depending on the the number of active senders, M, and the total number of participants, N. Given that M is almost always one for typical audio conferences, most of these models scale similarly in terms of processing and bandwidth requirements. Note that the centralized model performs better with higher M if inputs are summed.

Centralized: In the centralized model, a server receives media streams from all participants, mixes them if needed, and redistributes the appropriate media stream back to the participants (See Fig. 1). Since senders would have difficulty subtracting out their own contribution, the server needs to create a customized stream for each of the currently active M senders and a common stream for all $N - M$ listeners, assuming that they can all support the same media format. The server needs to decode audio streams before mixing, as mixing can only be performed on uncompressed audio. Decoding M and encoding $M+1$ streams limits the amount of active sources or conferences, while the number of participants limits the total conference membership to the available outbound network bandwidth.

The central server model has the advantage that clients do not need to be modified and do not have to perform media summing. In addition, it is relatively easy to support heterogeneous media clients, with the server performing the transcoding. For example, this allows a conference consisting of participants connected through high-bandwidth networks and modems, each receiving the best possible quality. At the cost of increased inbound bandwidth, silence detection can be delegated from clients to the server. This is helpful as many current IP telephones do not support silence suppression.

Also, the server can enforce floor control policies and can control the distribution of video based on audio activity. Compared to a distributed model, a central server can readily provide a consistent view of the complete conference membership.

Full mesh: In a full mesh, each active participant sends a copy of its media stream to all participants via unicast, without a central server. End systems sum the incoming audio streams; since most of the time, only one speaker will be active, the CPU overhead is modest as long as silence suppression is implemented everywhere, but it fails if the access bandwidth of some participants is just large enough for a single stream. For video, full mesh does not scale unless, for example, only currently active speakers send video. In a full mesh, each pair of participants must share a common codec.

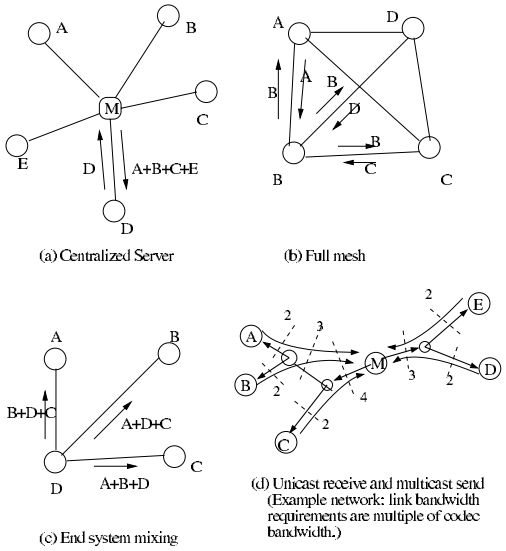


Fig. 1. Types of media distribution model

Properties	centralized	full mesh	multicast	unicast rx, multicast tx	end mixing
Topology	Star	full mesh	m-cast tree	star and m-cast tree	ad-hoc
Server processing	O(M+N)	n/a	n/a	O(M+N)	n/a
Endpoint processing	O(1)	O(M)	O(M)	O(1)	variable
Server bandwidth	O(M+N)	n/a	n/a	O(M) based on m-cast tree	n/a
Endpoint bandwidth	O(1)	O(M)	O(1)	O(1)	variable
Scaling	medium	medium	large	large	medium
Heterogeneous endpoints	yes	yes	no	no	yes (partially)
Get back your media	no	no	no	yes	no

TABLE I
TYPES OF CONFERENCES; M IS THE NUMBER OF ACTIVE SENDERS AND N THE TOTAL NUMBER OF PARTICIPANTS

Multicast: Network-layer multicast is ideally suited for large-scale conferences. A multicast address is allocated for each media stream, and every participant sends to that address. As in the full mesh, participants receive packets on the same address from all other participants, and need to sum or select streams. While the incoming bandwidth is the same as in a full mesh, each system only needs to generate one copy of the media stream.
Unfortunately, native multicast is not widely available outside network testbeds such as Internet2. Also, all receivers must share a common set of codecs.

Unicast receive and multicast send: This scheme combines some of the benefits of the server and multicast models. Participants send their media stream using unicast to the conferencing server, which sums them and sends them out on a pre-established multicast address. Thus, unlike pure multicast, end systems do not have to filter or mix media streams. Every participant receives the mixed stream, which includes his own stream. Unless a sender maintain a buffer of the data sent and there is a means of aligning time scales, it will have difficulty removing its own audio content from the mixed stream. The gain in bandwidth efficiency is largest if the number of simultaneous senders is small compared to the total group size. This approach lends itself well to single-source multicast [10], [11].

Endpoint mixing: Instead of in a server, mixing can take place in one of the participating end systems. For example, if A and B are in a call, A can also invite C. A sends the sum of A and B to C, and the sum of A and C to B. B and C do not need to be aware of the service performed by A, but can in turn mix other participants.
Cascading mixers increases the delay on some of the media paths. Another problem is that the conference dissolves when the participant who is acting as a mixer leaves the conference. This model is likely to be suitable only for small conferences of three or four parties.

Besides these, one can imagine a replication model, where the server sends a copy of each incoming media stream to all the participants using unicast. The mixing is done at each end system. This might be useful for media path authentication as every end system exchanges media packets only with the server's IP address. The CPU overhead is modest as long as silence suppression is implemented. The server however is less loaded than in the case of the centralized conference since it is now freed from the task of mixing audio streams. This is the model used in the case of video and text based conferences, since there is inherently no mixing required.

Media and signaling can use different models in the same conference. For example, one could combine centralized signaling with multicast media distribution, where the server maintains a one-to-one signaling relationship with each of the participants. Unfortunately, this requires cooperation from the end system. The server can indicate a multicast address in its SIP success response, causing the end system to send media streams via multicast, but the end system will still expect to receive media via unicast. More sophisticated session description formats may address this issue.

Also, different media streams can use different models. For example, audio could be mixed by a central server and redistributed, while video can be sent point-to-point between every pair of participants as in full mesh.

Thus, as long as multicast is not widely available, server-based conferences will continue to be the only viable model for mid-size conferences of tens to hundreds of participants.

IV. DESIGN OF A CONFERENCE SERVER

A conferencing server consists of a signaling and a media mixing module. The signaling module receives SIP or H.323 requests to join and leave conferences, while the media mixing module receives and sends RTP media streams from and to participants. Replicating video packets is straightforward; below we describe the operations needed for mixing audio.

A. Audio mixing

Fig. 2 shows how an audio mixing module can be implemented. Participant A support G.711, B DVI ADPCM and C both GSM and G.711. Participants list the codecs they support in their INVITE requests. The server selects an intersection of the algorithms supported by the participant as well as by the server. This selection is returned in the signaling success response to the participant. These algorithms are listed in order of preference in the SDP of the INVITE or its response.

The mixing algorithm follows a *decode-mix-encode* sequence. When an audio packet arrives at the mixing module, it is decoded into 16-bit linear samples and appended to the per-

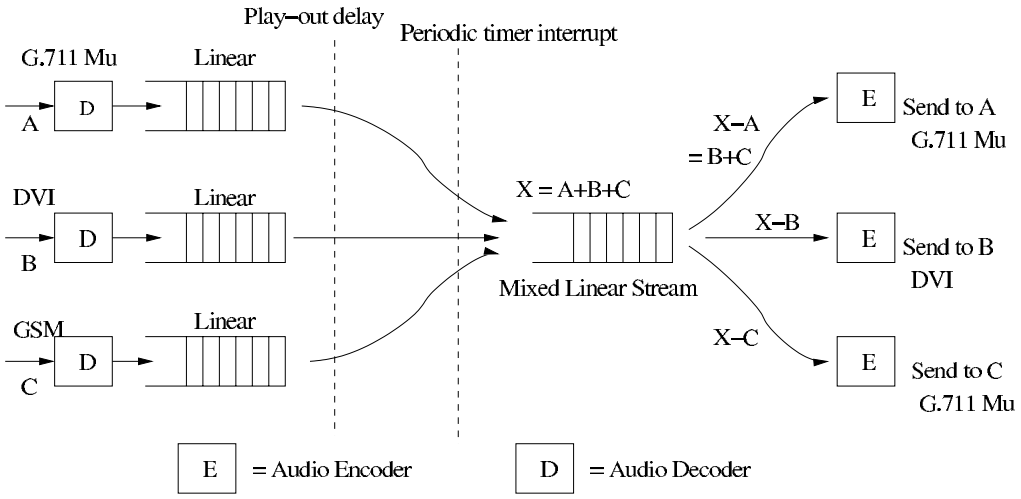


Fig. 2. Audio mixing

participant audio buffer queue. Each buffer is labeled with the corresponding RTP timestamp. The jitter in packet arrivals is absorbed by a play-out delay algorithm. Every outbound packetization interval, a timer triggers a routine that mixes a range of the samples from one of more input buffers from each active participant into a combined packets by simple addition of the sample values. The timer intervals are shortened and lengthened to account for earlier slight variations in timer invocation times and processing delay. (A simple operating system timer that fires after a delay would yield an output rate that is typically slightly below the desired rate.)

To allow input and output packets to have different packetization intervals, the mixer routine can grab samples from one or more input buffers. (Using a chained list of buffers saves memory compared to a circular buffer and makes it easier to detect when a particular source is silent.) Then, for each of the participants, the linear sample values from the per-participant queue (e.g., A) is subtracted from the mixed data (X) and the resulting data ($X - A$) is encoded using the preferred audio compression algorithm. The encoded data is packetized and sent to the participant. If there are M participants, then both mixing and redistribution will take M additions and M subtractions. Note that the receive and transmit audio algorithms need not be same for each participant.

While the *decode-mix-encode* sequence is the most straightforward approach to implementing an audio mixer, there are alternative approaches. For instance, one can build an addition or subtraction table for G.711 samples, so that conversion to linear is not required to do mixing. This only works for G.711, not for codecs with cross-sample dependencies such as G.723.1 or GSM.

Also, instead of subtraction, one could create $M+1$ different streams directly, one for each talker and one for the listeners. However, that requires M^2 additions.

B. Playout delay algorithm

Playout delay algorithms help absorb the jitter in network packet arrival due to network congestion. Adaptive playout delay further allows an application to adapt to changes in the amount of jitter, thus giving minimum delay in the audio stream. Playout delay compensation takes place before mixing, stretching or shrinking silence periods between talkspurts to adjust the time between arrival and mixing [12], [13]. (In the absence of silence periods, time stretching or companding can be used, albeit at much greater computational cost.) We have used Algorithm 1 from [12], with $\alpha = 0.95$, for our implementation. The algorithm is basically a linear recursive filter. The adapted delay at any instant depends on the measured delay (using RTP timestamps) plus the previous adapted delay, with a weighting factor α. The playout delay depends on both the adapted delay and the variation in the adapted delay.

V. Implementation

We have implemented a simple SIP conference server based on the above design. It can support some of the common audio algorithms, including G.711 A and μ-law, DVI ADPCM, and GSM. The Columbia SIP C++ library is used for all the SIP and SDP related functionality in the conference server. When a SIP user agent connects to the server the signaling is managed by the routines in this library. The mixer module forms the core of the conference server. We use Columbia RTP Library for implementing the RTP functionality. It sets up and manages the audio transport with the participants. There is a thread for receiving packets from every participant. Another thread (per conference) takes care of sending the mixed stream to the participants.

Below, we discuss design and implementation issues and

present initial performance data.

A. Design issues

Packetization interval: Although RTP implementations are supposed to handle a wide range of packetization intervals, we found 20 ms to be the only one that worked across a range of media clients such as rat [14] or Microsoft NetMeeting. End systems permitting, it may be useful to dynamically change the packetization interval for outgoing packets, as smaller packetization intervals decrease delay, but increase network bandwidth and computational effort.

Scaling: For large conferences, scalability is limited primarily by outbound bandwidth, copying of data between buffers and encoding. If many smaller conferences are to be supported, scaling depends as much on inbound bandwidth and decoding. While simple codecs like G.711 require very little encoding and decoding effort, they impose a heavier burden on buffer copying and bandwidth.

To scale to very large conferences using conferencing servers, a network of servers can be deployed (Section VI). To scale to a large number of smaller conferences, a SIP proxy server can act as a load-distribution system and direct incoming requests for new ad-hoc conferences to different servers. Alternatively, the conference server itself can redirect a request to an alternate server.

Instead of using general-purpose computers, one could also build DSP-based customized hardware at lower per-port cost. However, in many environments, there are enough idle cycles on workstations and servers that can be drafted into service for occasional large conferences.

Inactivity detection: The system should be able to detect if a particular participant becomes inactive, e.g., due to user agent failure. Failures can be detected by observing ICMP errors or sudden discontinuation of RTCP reports.

B. Performance measurements

We are currently measuring performance of our software on a range of platforms. Initial results are below. We characterize server load by processor and memory utilization. As discussed above, both the number of conferences and the total number of participants affect load, assuming that the average number of active senders per conference is one.

Table II summarizes the server load depending on how many simultaneous participants are present in a single conference. There was no optimization done at compilation time. There were only one or two active speakers and all others were listeners. The server was running on a Sun SPARC Ultra 10 with 256 MB RAM and a 360 MHz CPU. All participants were in the same 100 Mb/s LAN as the server and used G.711 with a 40 ms packetization interval from server to participant and 20 ms from participant to server. The bandwidth includes IP, UDP and RTP headers, and for a typical 100 Mb/s LAN, is not a limiting factor.

Load figures are obtained using the Unix command `top`. Memory is the amount of resident memory (RES). The audio quality was good up to 80 participants in the single conference, tolerable with 100 participants and very poor for 120 participants.

Participants	CPU (%)	memory (MB)	bandwidth (Mb/s) inbound	outbound
2	< 0.1	2.7	0.08	0.07
20	< 1	6.0	0.08	1.37
40	2-3	9.6	0.08	2.81
60	5	13	0.08	4.25
80	10-15	17	0.08	5.69
100	35-50	22	0.08	7.13
120	50-70	26	0.08	8.59

TABLE II
SERVER LOAD AS FUNCTION OF NUMBER OF PARTICIPANTS IN SINGLE CONFERENCE

Table III summarizes the server behavior depending on the number of simultaneous three-party conferences where every participant is an active speaker. All other parameters are the same as before. Audio quality was good up to 15 three-party conferences, but deteriorated to poor with 18 conferences.

Conferences	participants	CPU (%)	memory (MB)	bandwidth (Mb/s) inbound	outbound
3	9	< 0.4	4.1	0.72	0.65
6	18	< 2.0	5.7	1.44	1.30
9	27	7-13	7.3	2.16	1.94
12	36	15-20	9	2.88	2.60
15	45	25	10	3.60	3.24
18	54	30	12	4.32	3.89

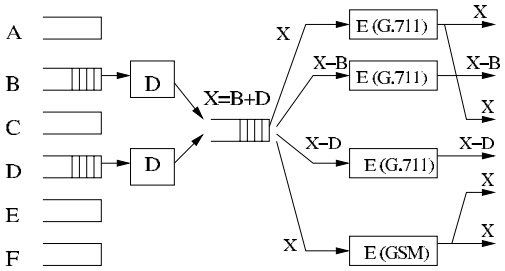
TABLE III
SERVER LOAD AS FUNCTION OF NUMBER OF THREE-PARTY CONFERENCES

The memory requirement depends on the number of participants and seems to increase linearly. For instance, memory requirements for 15 three-party conferences (45 participants) is almost the same as that for 40 participants in a single conference. Secondly, the CPU utilization starts increasing drastically at about 30-40 participants.

CPU load is the primary bottleneck in our test environment. The other factors: memory and bandwidth do not seem to cause problem for a hundred participants. Various factors contribute to the CPU load, e.g., thread switching and list traversal. This bottleneck can be removed by using "Multi-state conferences" (Section VI) or by dedicated hardware for encoding and mixing, for example.

It may be possible to optimize the mixing logic. One such scheme is shown in Fig. 3, combining the encoding step for the output streams that have same mixed audio data and use the same encoding algorithm. For all the participants who did not speak in the last timer interval and who have a common subset of supported receive audio algorithm, we can call the encoder only once. However, if a stream stops being active, it will receive the general listener packet stream rather than its own version, so that the predictor will be wrong. It is not clear how much this would matter in practice.

Scaling may also be limited by the available number of



A–D support G.711; E and F support GSM.

Fig. 3. Possible optimization in decode-mix-encode sequence

threads. Our implementation allocates a thread for every conference as well as for every participant. With 1000 threads allowed per process, the server can support 250 three party conferences (with 750 participants), for example.

VI. Conferencing as part of an overall VoIP architecture

This section describes enhancements to the simple centralized conference system and how it can fit into a more complex Internet telephony and multimedia communication environment.

A. Multi-protocol conference server

A simple enhancement (Fig. 4) is to use a SIP-H.323 gateway and SIP-PSTN gateway to provide a unified conferencing server which can be contacted from any of the SIP, H.323 or PSTN networks. To integrate PSTN users, some form of interactive voice response (IVR) is required, e.g., to prompt for pass codes.

B. Network of conference servers

For larger conferences, it is possible to create a tree of conference servers, where each server appears as a participant in the server at the aggregation level above it (Fig 5). In the figure, S2, S3, and S4 act as participants for the conference server S1. Such a tree adds packetization and playout delay, but can approximate the bandwidth scaling benefits of network-layer multicast if participants select the closest server. Since it is common that corporate conferences consist of a large number of participants spread across a relatively small number of facilities, having a server in each LAN is likely to be a common mode of operation.

C. Integration with other services

A conferencing server can be integrated with a text-to-speech and speech recognition system to allow text-only participants in an audio session. A conference server could also include an RTSP client that can stream media to a recording server.

VII. Related work

Most of the conference servers in the market today are based on H.323. These include MeetingPoint from CUseeMe Networks, Sametime from Lotus and Microsoft Exchange 2000

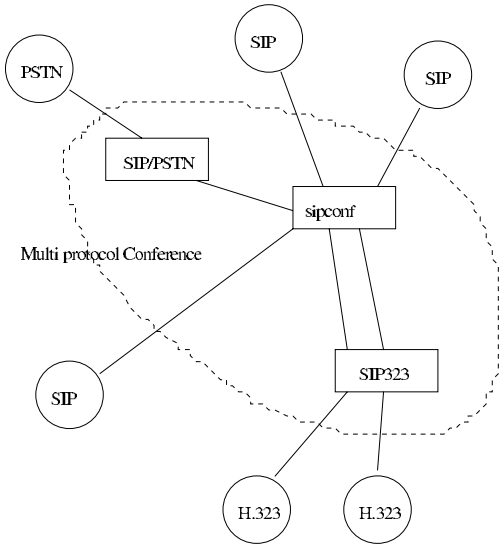


Fig. 4. Multi-protocol conference server

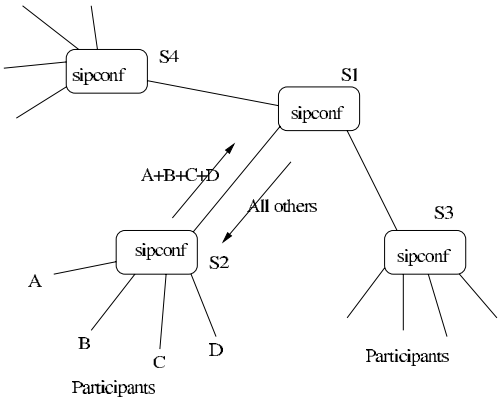


Fig. 5. Multi-stage conference servers

Conferencing Server. These support T.120 for application sharing and whiteboards. MeetingPoint has mechanisms to link servers together so that conferences can be shared and load-balancing can be done.

VideoTalks [15] by AT&T Labs is a comprehensive multimedia conferencing system intended to provide a variety of Internet services such as video conferencing and low cost video-on-demand. It is not based on SIP.

A number of tools (e.g., RAT and NeVoT) support multicast "light-weight" conferencing, without explicit signaling support [9]. Etherphone [16] is probably one of the earliest systems supporting multimedia conferencing.

Most of these work talk about conferencing in general or a specific implementation of a conferencing system. Here, we compare different models and present performance numbers for a real implementation.

VIII. Conclusion and future work

Based on our implementation, SIP provides a suitable multimedia conferencing platform that allows advanced scenarios and services without requiring that end systems are conferencing-aware. It is possible to build medium scale conferencing servers in software. Our implementation supports up to 100 participants in a single conference using G.711 audio and only one active speaker on a Sun Sparc Ultra-10 platform. It can also support up to 15 three party conferences where all participants are speaking simultaneously. Scalability can be improved by ensuring that the clients support silence suppression at their end.

In addition to audio and video conferences, various other services can be provided at the conference server such as whiteboard applications and multi-user games. This may lead to a distributed conferencing server architecture with different components handling different services. Participants can also join conferences from the PSTN if they use SIP to PSTN gateways. SIP-H.323 gateways [17] exist that can permit the participation of H.323 clients in the conference.

We plan to enhance our prototype in a number of ways:

- We need to gather performance data for different codecs in a heterogeneous conferencing environment, on different computing platforms, including those with multiple processors. We also plan to measure the delay and jitter at the client side as the server load increases.
- Video support will be added to the server. This involves defining policies on which video stream to distribute and possibly merging streams into screen quadrants. (With windows-based end systems, this is not required as the end system can arrange multiple video windows.)
- We plan to add additional codecs, beyond the G.711 μ-law, G.711 A-law, GSM and DVI ADPCM currently supported.
- Additional SIP features such as dial-out and authentication can be added to the server. This allows the server to invite participants to the conference and keep unauthorized participants out. Conferences could also be bounded in duration; however, since the resource consumption of inactive conferences is very small as long as media streams are muted, it is quite feasible to set up permanent conferences in work groups, for hoot-and-holler applications. The transition from centralized conferences to full-mesh and multicast conferences, as well as hybrid solutions, need to be supported.

IX. Acknowledgments

We would like to thank Jonathan Rosenberg and Weibin Zhao for their valuable and insightful comments.

References

[1] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: session initiation protocol," Request for Comments 2543, Internet Engineering Task Force, Mar. 1999.

[2] J. Rosenberg and H. Schulzrinne, "Models for multi party conferencing in SIP," Internet Draft, Internet Engineering Task Force, Nov. 2000. Work in progress.

[3] Z.-Y. Shae and M.-S. Chen, "Mixing and playback of JPEG compressed packet videos," in *Proceedings of the IEEE Conference on Global Communications (GLOBECOM)*, (Orlando, Florida), pp. 245–249 (08B.03), IEEE, Dec. 1992.

[4] H. Schulzrinne, "RTP profile for audio and video conferences with minimal control," Request for Comments 1890, Internet Engineering Task Force, Jan. 1996.

[5] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications," Request for Comments 1889, Internet Engineering Task Force, Jan. 1996.

[6] J. Rosenberg, P. Mataga, and H. Schulzrinne, "An application server component architecture for SIP," Internet Draft, Internet Engineering Task Force, Nov. 2000. Work in progress.

[7] International Telecommunication Union, "Packet based multimedia communication systems," Recommendation H.323, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Feb. 1998.

[8] M. Handley and V. Jacobson, "SDP: session description protocol," Request for Comments 2327, Internet Engineering Task Force, Apr. 1998.

[9] M. Handley, J. Crowcroft, C. Bormann, and J. Ott, "The internet multimedia conferencing architecture," Internet Draft, Internet Engineering Task Force, July 2000. Work in progress.

[10] S. Bhattacharyya *et al.*, "A framework for source-specific IP multicast deployment," Internet Draft, Internet Engineering Task Force, July 2000. Work in progress.

[11] H. W. Holbrook and D. R. Cheriton, "Ip multicast channels: EXPRESS support for large-scale single-source applications," in *SIGCOMM Symposium on Communications Architectures and Protocols*, (Cambridge, Massachusetts), August/September 1999.

[12] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Toronto, Canada), pp. 680–688, IEEE Computer Society Press, Los Alamitos, California, June 1994.

[13] J. Rosenberg, L. Qiu, and H. Schulzrinne, "Integrating packet FEC into adaptive voice playout buffer algorithms on the internet," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Tel Aviv, Israel), Mar. 2000.

[14] A. Sasse, V. Hardman, I. Kouvelas, C. Perkins, O. Hodson, A. Watson, M. Handley, J. Crowcroft, D. Harris, A. Bouch, M. Iken, K. Hasler, S. Varakliotis, and D. Miras, "Rat (robust-audio tool)," 1995.

[15] M. R. Civanlar, G. L. Cash, R. V. Kollarits, B.-B. Paul, C. T. Swain, B. G. Haskell, and D. A. Kapilow, "VideoTalks: A comprehensive multimedia conferencing system," in *Proc. of Packet Video*, (Sardinia, Italy), May 2000.

[16] H. M. Vin, P. T. Zellweger, D. C. Swinehart, and P. V. Rangan, "Multimedia conferencing in the Etherphone environment," *IEEE Computer*, vol. 24, pp. 69–79, Aug. 1991.

[17] K. Singh and H. Schulzrinne, "Interworking between SIP/SDP and H.323," in *Proceedings of the 1st IP-Telephony Workshop (IPtel 2000)*, (Berlin, Germany), Apr. 2000.

An Adaptive Mechanism for Real-time Secure Speech Transmission over the Internet

Alessandro Aldini, Roberto Gorrieri, Marco Roccetti
Dipartimento di Scienze dell'Informazione
University of Bologna, Italy
E-mail: { aldini, gorrieri, roccetti }@cs.unibo.it

Abstract—
The Internet offers a best-effort service over public networks which do not guarantee privacy. Because of this the provision of secure real time audio applications has received increasing interest and was an active research area in last years. We propose an adaptive packet audio control mechanism, originally designed for controlling and adapting the audio applications to the network conditions, and now enriched with cryptographic features in order to support secure, unicast, voice-based communications over the Internet. We take advantage of the characteristics of the adaptive mechanism, which meets the real time constraints needed by audio transmission applications, in order to realize a lightweight security infrastructure which offers privacy, authenticity and integrity assurances in a simple way and at a negligible cost. Finally, we show the performance of the proposed mechanism and we contrast it with those of other well-known tools designed for the secure audio transmission over the Internet.

Keywords—**Internet, Multimedia Applications, Security, Real Time.**

I. Introduction

SUPPORTING real time audio applications over wide area networks has been the subject of continuous research during the past recent years. Sophisticated applications of Internet multimedia conferencing will become increasingly important only if the speech quality and privacy provided by the communications will be perceived as sufficiently good by their users.

From a performance standpoint, the feasibility and the expected QoS of audio applications over public networks have to be carefully considered, if we wish those applications be successful in spite of the possibly restrictive resources (e.g. bandwidth) they are constrained to work with. In particular a significant issue in interactive sound transmission dealing with restrictive network resources is the problem of minimizing the latency due to each step of the communication. The main problem is that over the Internet only a flat, classless, best-effort service may be offered. For instance, as far as the transmission delay is concerned, the IP model does not consider the provision of QoS guarantees with the proper intensity. As a consequence, real time audio traffic experiences unwanted delay variation (known as jitter) with peaks on the order of 500/1000 ms for congested Internet links [9]. On the contrary, it is well accepted that telephony users find round trip delays larger than 300 ms more like a half-duplex connection than a real time conversation. In addition, too large audio packet loss rates (over 10%) may have a tremendous impact on speech recognition ([15], [6]). These observations put in evidence the importance of the trade-off between the stochastic end-to-end delay of the played out audio packets and the packet losses, especially when dealing with the problem of unpredictable jitter typical of environments providing a best-effort service. Hence the most important metric affecting the user perception of audio is represented by the average packet audio playout delay vs. the packet loss rate, where with the term playout delay we refer to the total amount of time that is experienced by the audio packets from the time instant they are generated at the sender site to the time instant they are played out at the receiver site ([1], [16], [18]). The problem of obtaining the optimal trade-off between these two aspects and facing the constraints on strict delays and losses tolerated in an unfavourable platform is addressed by adaptive packet audio control algorithms (see e.g. [16], [18], [22]), which adaptively adjust to the fluctuating network delays of the Internet in order to guarantee, when possible, an acceptable QoS.

In this work we consider the adaptive packet audio control mechanism proposed by one of the authors ([22], [21]) and we cope with the problem of adding security to the audio data flow pipeline generated by such a mechanism. This is done because the problem of considering security constraints when using public networks raises, as real time audio communications are a much less secure service than most people realize. Indeed, it is relatively easy to eavesdrop phone conversations, and the situation is even worse in the case of audio applications over the Internet, because anyone with a PC and an access to the public network has the possibility to capture the network traffic, potentially compromising the privacy and the reliability of the applications. Hence, it is mandatory for audio applications to guarantee authentication, confidentiality, and integrity of data (see e.g. [17] about the authentication problem in the case of video streaming).

The adaptive nature of the mechanism we adopt is particularly suitable to include with a negligible effort an adequate and lightweight security infrastructure. Such an enriched mechanism allows the two trusted parties to have a private conversation by employing a stream cipher, whose cryptanalysis is made much more difficult by the particular behavior of the adaptive algorithm. In particular, it allows the parties taking part into the audio communication to agree on a sequence of session keys, where the lifetime of each key is limited to a temporal interval not greater than one second of conversation (corresponding to less than 2^{12} bits of transmitted data), whereas the best known attacks of some stream ciphers based on linear feedback shift register require in the best cases 2^{20} to 2^{33} ciphertext bits (with complexity 2^{59} to 2^{21}, respectively) [12], [8].

Summarizing, our scheme offers a minimal per-packet communication overhead, and, thanks to the original adaptive mechanism, also arbitrary packet loss and delay tolerated. Moreover, it provides the receiver with a high assurance of secrecy, integrity, and authenticity, as long as the underlying cryptographic

assumptions are enforced. The negligible cost of such an integration is supported by some experimental results, presented also with a comparison with the performance of other tools.

To the best of our knowledge, in the literature the tools based on adaptive playout adjustment schemes either do not consider security services (see e.g. FreePhone [2]) or simply enable encryption of data by using the well-known DES block cipher [24] and a key prefixed by the two parties (see e.g. NeVot [26], and rat [13]). In addition, some other software packages working at the application layer are proposed to offer a secure audio communication over the Internet (see e.g. Nautilus [11], PGPfone [29], and Speak Freely [28]), but they do not include mechanized adaptive playout adjustment schemes.

The paper is organized as follows. In Sect. II we first discuss how to approach the problem of guaranteeing real time secure audio communications over IP, and then we specify the characteristics of the system (and the adversary) which the mechanism we propose is able to cope with. In Sect. III we present the mechanism and we describe the security properties which are met by our algorithm. In Sect. IV we analyze the performance of the mechanism together with the performance of well-known software tools designed for the secure audio transmission over the Internet. Finally, in Sect. V some conclusions are drawn.

II. Real-time Secure Audio Transmission

In this section we briefly describe how to get over the problems introduced when taking into account real time and security requirements, and then we fix the features of the models of (i) the system we rely on and (ii) the adversary we should cope with.

In this work we consider an adaptive audio control software mechanism, originally designed for controlling and adapting the audio application to the network conditions [22]. The mechanism has been passed through intense functional and performance analysis [1], which revealed its adequacy to guarantee real time constraints, and it has been recently implemented in a software tool called BoAT [23]. The motivation under the development of this kind of mechanism is that in the absence of network support to provide quality guarantees of Internet voice software, an interesting alternative to deal with jitter and high packet loss is to use adaptive control mechanisms. In fact, jitter-free, in-order, on-time packet delivery rarely, if ever, occurs in today's packet-switched networks. The provision of a synchronous playout of audio packets at the receiver site, in spite of stochastic end-to-end network delays, is typically achieved by buffering received audio packets and delaying their playouts, so that most of packets will have been received before their scheduled playout times. At the sending site, packet audio mechanisms operate by periodically gathering audio samples, packetizing them, and transmitting the packets to the receiving site. On the other site, received packets are queued into a smoothing buffer and the playout of packets is adaptively delayed. A strict connection exists between this additional delay introduced by the receiver buffer and the number of lost packets due to late arrivals, and the goal of these mechanisms consists in achieving the optimal trade-off.

Actually, these adaptive packet audio control mechanisms do not consider all the security problems, in that they do not guarantee confidentiality of the audio conversation, integrity of the transmitted data, and authentication of the involved parties. The need to consider such problems when modeling applications over IP is well accepted. The Internet Protocol underlies large academic and industrial networks as well as the Internet. IP's strength lies in its easy and flexible way to route packets; however, its strength is also its weakness. Indeed, the way IP routes packets makes large IP networks vulnerable to a range of security risks, e.g. spoofing (meaning that a machine on the network masquerades as another), and sniffing (meaning that a third party listens in a transmission between two other parties). In order to protect sensitive communications in such a scenario several approaches have been suggested. For instance, the Internet Engineering Task Force (IETF) has developed the IP Security (IP-Sec) protocol suite [14], a set of IP extensions that provide security services (such as access control, integrity, data origin authentication, and confidentiality) at the network level. IP-Sec technology seeks to secure the network itself, instead of the applications that use it; just as IP is transparent to the average user, so are IP-Sec based security services. Some key issues that need to be addressed concern the level of interoperability reachable by this standard and the computational performance costs imposed by the use of IP-Sec. In particular, these costs are associated with the memory needed for IP-Sec code and data structures, the computation of integrity check values, encryption and decryption, and added per-packet handling. The per-packet computational costs are manifested by increased latency and, possibly, reduced throughput. This is due to the increase in the packet size resulting from the addition of IP-Sec dedicated headers, and the increased packet traffic associated with key management protocols. In general, IP-Sec cannot be optimized for special-purpose applications. These considerations are emphasized especially in the case of transmission of real time audio packets.

In the same line of the above discussion, in this paper we consider an application-level extension of the audio mechanism of [22] in order to provide the audio communication over IP with all the main security services. Our approach in equipping this mechanism with security modules is said to be lightweight, because the used cryptography infrastructure exploits the particular adaptive audio control scheme in order to make secure the application with a minimal computational cost.

Before presenting the characteristics of the extended algorithm, we need to define (i) the environment in which the mechanism is expected to work and (ii) the threat model such a mechanism should deal with, which basically reflects the assumptions of the Dolev-Yao model [10].

A. The System Model and the Threat Model

An ideal network can be expected to provide some precise properties; for instance it should (i) guarantee message delivery, (ii) deliver messages in the same order they are sent, (iii) deliver at most one copy of each message, and (iv) support synchronization between the sender and the receiver. All these properties are favourable in order to support real-time applications such as multimedia conferencing over wide area networks.

However, the underlying network upon which we operate has certain limitations in the level of service it can provide. Some

of the more typical limitations of the network we are going to consider are that it may:

- drop messages;
- reorder messages;
- deliver duplicate copies of a given message;
- limit messages to some finite size;
- deliver message after an arbitrarily long delay.

A network with the above limitations is said to provide a *best-effort* level of service, as exemplified by the Internet. This model adequately represents the Internet as well as shared LANs, but not switched LANs. All the dissertations and the results presented in the next sections are obtained under such a model of the network.

As far as the adversary model is concerned, we argue that our mechanism is secure also in the presence of a powerful adversary with the following capabilities:

- the adversary can eavesdrop, capture, drop, resend, delay, and alter packets;
- the adversary has access to a fast network with negligible delay;
- the adversary computational resources are large, but not unbounded. He knows every detail of the cryptographical algorithm, and he is in possession of encryption/decryption equipment. Nonetheless the adversary cannot guess secret keys or invert pseudorandom functions with non-negligible probability.

III. The Mechanism

The mechanism proposed in [22] has been originally designed to dynamically adapt the playout delay of the received audio packets to the network conditions assuming neither the existence of an external mechanism for maintaining an accurate clock synchronization between the sender and the receiver, nor a specific distribution of the end-to-end transmission delays. Such a scheme relies on a periodic synchronization between the sender and the receiver in order to obtain, in periodic intervals (at most 1 second), an estimation of the upper bound for the packet transmission delays experienced during an audio conversation. Such an upper bound is periodically computed using round trip time (RTT) values obtained from packet exchanges of a handshaking protocol performed among the two parties. In this work we exploit the handshaking protocol for a twofold goal:

- it allows the receiver to generate a synchronous playout of audio packets, in spite of stochastic end-to-end network delays;
- it allows the two authenticated parties to agree on a sequence of exchanged keys such that a third party cannot know it.

In the paper we adopt the following notation: S is the sender, R is the receiver, M_j is a chunk of audio conversation contained in a packet, P_j denotes a packet composed of a timestamp and an audio sample M_j. We denote with K_0 a symmetric key agreed during a preliminary authentication phase (e.g. by using a regular digital signature scheme such as RSA [19]), and with K_i any subsequent session key agreed among the two authenticated parties. Moreover, we assume that the packets of the handshaking phase are encrypted with K_0 by using any one of the block ciphers for the symmetric cryptography (such as RC6 and Blowfish) [24].

A. An Adaptive Playout Control Algorithm

As previously explained, one of the goal of the handshaking synchronization protocol consists of providing an adaptive control mechanism at the receiver site in order to properly playout the incoming audio packets. This is typically achieved by buffering the received audio packets and delaying their playouts, so that most of packets, in spite of stochastic end-to-end network delays, will have been received before their scheduled playout times. This is achieved as follows. The first handshaking protocol precedes the audio conversation and then is carried out every second along the conversation lifetime. As a proof-of-concept, before detailing the protocol, we present the handshaking phase as follows:

Direction	**Message Type**	**Contents of packet**
$S \rightarrow R$	*probe*	sender time t_s
$R \rightarrow S$	*response*	sender time t_s
$S \rightarrow R$	*install*	RTT computed by the sender
$R \rightarrow S$	*ack*	RTT computed by the sender

As shown in the above scheme, the sender begins the packet protocol exchange, by sending a *probe* packet timestamped with the time value shown by its own clock (t_s). At the reception of this packet, the receiver sets its own clock to t_s and sends immediately back a *response* packet. Upon receiving the response packet, the sender computes the value of the RTT by subtracting the value of the timestamp t_s from the current value of its local clock. Then it sends to the receiver an *installation* packet, with attached the calculated RTT value. Upon receiving this packet, the receiver sets the time of its local clock, by subtracting from the current value of its local clock the value of the transmitted RTT. At the end of this protocol, the receiver is provided with the sender's estimate of an upper bound for the transmission delay that can be used in order to dynamically adjust the playout delay and buffer. Based on the value of the time difference (called Δ) between the two system clocks imposed by the protocol the following strategy may be followed. The sender timestamps each emitted audio packet P_j with the value of its local clock t_s at the moment of the audio packet generation. When an audio packet arrives, its timestamp t_s is compared with the value t_r of the receiver clock, then a decision is taken according to the following rules:

Condition	**Effect on the packet**	**Motivation**
$t_s < t_r$	discarded	it is arrived too late to be played out
$t_s > t_r + \Delta$	discarded	it is arrived too far in advance of its playout
$t_r \leq t_s \leq t_r + \Delta$	buffered	it is arrived in time for its playout

Using the same rate adopted for the sampling of the original audio signal at the sender site, the playout process at the re-

ceiver site fetches audio packets from the buffer and sends them to the audio device for playout. More precisely, when the receiver clock shows a value t_r, the playout process searches in the buffer the audio packet with timestamp t_r. If such a packet is found, it is fetched from the buffer and sent to the audio device for immediate playout.

In essence, a maximum transmission delay equal to Δ is left to the audio packets to arrive at the receiver in time for playout. In particular, the playout instant of each packet arrived in time is scheduled after a time interval equal to the positive difference between the values of t_s and t_r. The playout buffering space is proportional to Δ and allows the packets with early arrivals to be scheduled according to the above rules. Packets arrived too far in advance are discarded because their playout instant is beyond the borderline of the temporal window determined by the buffering space. The proposed scheme adaptively adjusts to the fluctuating network delays of the Internet thanks to the periodic clock synchronization carried out throughout the entire conversation lifetime. Whenever a new synchronization activity is conducted, a new RTT value is computed and depending on its value the clock values, the buffering delay and the buffer dimension are updated. This method guarantees that both the introduced additional playout time and the buffer dimension are always proportioned to the traffic conditions. The reader interested in more technical details and proofs related to the adaptive control mechanism should refer to [22], [1].

B. Securing the Mechanism

B.1 The Handshaking Protocol

As far as the security is concerned, we exploit the existing handshaking protocol in order to exchange among the two authenticated parties fresh session keys, and more precisely a key for each synchronization phase. Such a key will be used to secure the audio conversation and will have a lifetime equal to at most one second, namely the time between two consecutive synchronizations. More precisely, we use such a key as the session key of a stream cipher used to encrypt data. A stream cipher is a symmetric encryption algorithm which usually is faster than any block cipher. While block ciphers operate on large blocks of data, stream ciphers typically operate on smaller units of plaintext, usually bits. A stream cipher generates what is called a keystream (a sequence of bits used as a key) starting from a session key K which is used as a seed for the pseudorandom generation of the keystream. Encryption is accomplished by combining the keystream with the plaintext, usually with the bitwise XOR operation. Examples of well-known stream ciphers are A5/1 (used by about 130 million GSM customers in Europe to protect the over-the-air privacy of their cellular voice and data communication), RC4 (by the RSA's group), and SEAL.

In our protocol, during the generic handshaking phase i the two authenticated parties agree on a 128-bit session key K_i (e.g. exchanged in the *install* packet). We point out that the packets of the handshaking phases, instead of being encrypted by employing the particular stream cipher, are encrypted by using a block cipher (such as RC6 and Blowfish). Whenever the handshaking protocol has a positive outcome, K_i is the new key used to secure the subsequent chunk of audio conversation. Since the handshaking protocol is periodically started during the audio conversation, a sequence of keys $\{K_i\}_{i\in\mathbb{N}}$ is generated.

In order to guarantee the correct behavior of the above mechanism, both the sender and the receiver must come to an agreement. In particular, the sender site has to know whenever the receiver site has received the new RTT and key. Because of this, upon receiving the installation packet, the receiver sends back an *ack* packet. At the reception of this packet, the sender starts to use the new key. An additional information for each audio packet is used as a flag in order to inform the receiver that the key is changed and it is exactly the new key K_i. For instance, following a policy inspired by the alternating bit protocol, if each packet encrypted with the key K_i is transmitted with a flag bit set to 0, then whenever a new synchronization phase is completed, each subsequent packet is transmitted with the bit set to 1. It is worth noting that if either the installation packet or the ack packet do not arrive at their destination, both the sender and the receiver carry on the communication by using the old key. Indeed the sender begins to encrypt with the new key only if it receives the ack packet, and the receiver begins to decrypt with the new key as soon as it receives an audio packet whose flag has been changed with respect to the previous audio packets. The presented policy does not require additional overhead on the original scheme, because it relies on the original handshaking protocol.

As far as the secrecy, authenticity, and integrity conditions of the handshaking protocol are concerned, the following remarks are in order:

- an adversary cannot forge any packet, as he does not know the symmetric key used to encrypt them (e.g. he cannot create or alter a response packet with a given timestamp). He can cheat neither the sender nor the receiver by resending any packet, because of the presence of the timestamp t_s (in the case of the probe and response packets) and also the RTT (in the case of the install and ack packets).
- an adversary can try to drop systematically the messages of the handshaking protocol, so that the lifetime of the old session key is extended from one second to the whole duration of the conversation; in this way, many more data and time are at disposal of a cryptanalysis attempt. A possible solution consists of masquerading the handshaking packets as normal audio packets, by filling the audio sample with rubbish. An additional bit is used to distinguish at the receiving site among the different packets, whereas a third party cannot guess anything because all the data are encrypted. With this assumption in view, an adversary can only try to drop some packets in a random way and, as a consequence, he can break off several consecutive handshaking phases with a negligible probability. In spite of this, an intensive traffic analysis during a full-duplex conversation could significantly restrict the temporal interval in which the two parties are expected to send packets of the handshaking phase. If we want our mechanism to be more robust against this unlikely attack, we can shut down the conversation whenever more than n consecutive handshaking phases are not completed, for some suitable n depending on the strength of the cryptographic algorithm.

In general, the handshaking protocol does not reveal any information flow allowing an adversary to spoof or sniff the conversation. Moreover, the same mechanism is robust to lost

and misordered packets and makes no assumption on the service offered by the network. The described policy is similar to some well-known protocols for radio communications which are spread spectrum frequency open, in the sense that during a conversation the transmission frequency is frequently changed in order to avoid interception and alteration. In the case of our mechanism, the duration of each key is limited to the time space between two consecutive synchronizations (at most 1 second for normal executions), thereby this policy allows for making difficult for a not authenticated party to decode the encrypted data, and practically guarantees to be robust to trivial breaks [24].

B.2 Security Properties

From the security point of view, we aim at proving that our scheme guarantees the properties of secrecy, authentication, and integrity. As far as the secrecy is concerned we show that the robustness of our mechanism depends on both the particular stream cipher we adopt and the adaptiveness of the algorithm. As far as the authenticity is concerned, we show that after a preliminary authentication phase, the two trusted parties are provided with data origin authentication during the conversation lifetime. As far as the integrity is concerned, we show that the receiving trusted party can unambiguously decide that a received packet P_j (timestamped with a value t) is exactly the same packet P_j sent with timestamp t by the sending trusted party.

B.3 Securing the Conversation

The session key exchanged during the handshaking phase is used by the particular stream cipher for the encryption of both the timestamp and the entire audio packet. More precisely, each audio packet belonging to the chunk of conversation i between the two consecutive synchronizations i and $i+1$ is encrypted by resorting to the particular stream cipher and the session key K_i.

In order to guarantee authenticity and integrity of data, we employ this mechanism in conjunction with a message authenticating code (MAC). In particular we can adopt a mechanism similar to the HMAC-MD5 used also in [17] to ensure authenticity and integrity of the audio packets. Alternatively, we can encrypt (by the particular stream cipher) the output of a 1-way hash function applied to the audio packet to ensure authenticity and integrity of the same packet. Examples of well-known hash functions are MD5 and SHA.

In the following algorithm we describe such an approach which guarantees a secure audio conversation. In particular, we denote with $\{P_j\}_{K_i}$ the audio packet P_j encrypted by using the stream cipher starting from the session key K_i and with $MAC(K_i, P_j)$ the message authenticating code for the packet P_j obtained by resorting to the session key K_i.

The algorithm guarantees secrecy, and satisfies the properties of authentication and integrity. More precisely, it guarantees the following condition. *For each audio packet P_j^*, which is created with the above algorithm and received in time for its playout, the receiver can (i) decide its playout instant and (ii) verify its integrity and the authenticity of the sender.*

Algorithm

Sender

1. $P_j = \{t_s, M_j\}$
2. Send $P_j^* = \{\{P_j\}_{K_i}, MAC(K_i, P_j)\}$

Receiver

1. Receive P_j^*
2. Compute t_s and M_j by means of K_i
3. Verify the MAC

B.3.a Secrecy. As far as the secrecy is concerned, our mechanism guarantees that the trusted parties have a high assurance of the privacy of the data transmitted during the conversation lifetime. In fact, we have shown that (i) our protocol does not reveal any information about the secret keys exchanged between the trusted parties and (ii) an adversary as specified in Sect. II cannot guess secret keys. The secrecy is a crucial condition that the recent literature shows to be not met in glaring cases. For instance, let us consider the attack on the A5/1 algorithm (used in GSM systems) proposed in [7], in which a single PC is proved to be able to extract the conversation key in real time from a small amount of generated output. In particular, the authors of [7] claim that a novel attack requires two minutes of data and one second of processing time to decrypt the conversation. Now let us assume that the particular cipher we choose to adopt is as weak as the A5/1 algorithm. In the mechanism we propose, in the absence of a powerful adversary able to identify and drop the handshaking messages, during two minutes of conversation at least 120 different session keys are used, so that the quantity of data that can be analyzed for a single key is not sufficient to perform the attack and to reveal the key and, consequently, the audio conversation. Moreover, in support of the robustness of our approach we point out that in the recent literature the best known attacks of some stream ciphers, proposed in [8], have complexity 2^{59} and require 2^{20} bits of ciphertext and are based on some restrictive assumptions on the characteristics of the stream cipher. In [12], a novel attack has a complexity gain (2^{21}), but it requires 2^{33} bits of ciphertext and it can be used under specified conditions. We can add that guessing somehow a session key may allow an adversary to decipher just one second of conversation with no information about the remaining encrypted data. In general, it is worth noting that the relatively short lifetime of every session key improves the secrecy guarantees for any cryptographic algorithm. Anyway, a study conducted in [1] revealed that too short lifetimes (e.g. less than 0.5 sec) cause a worsening of the speech quality, therefore a massive resort to such an expedient should be carefully analyzed.

B.3.b Authenticity. As far as the authenticity is concerned, we first assume a preliminary authentication phase carried out by the two parties before the conversation (e.g. by resorting to a regular digital signature scheme). After this initial step, only the legitimate parties (i) know the value of the symmetric key agreed during this phase and (ii) can carry out the first packet exchange

of the handshaking protocol by means of the symmetric key. In particular, as we have shown in Sect. III-B.1, an adversary cannot start, carry out, and complete the packet exchange of this synchronization protocol with any of the trusted parties. Later on, during the conversation, each packet is timestamped with the sender clock value at the moment of the audio packet generation, encrypted by means of the session key K_i, and authenticated by means of the MAC, so that each received packet can be played out (i) only once and (ii) only if it arrives in time for being played out according to the adaptive adjustment carried out during the ith handshaking synchronization phase. The receiver is guaranteed that the audio packets encrypted by means of the key K_i and played out according to the piggybacked timestamp have been generated at (and sent by) the sender site. In fact, an adversary cannot behave as a "man in the middle", by creating new packets (as he does not know the session key and he cannot authenticate the packet) or spoofing (as he can resend or delay packets, but the timestamp allows the receiver to discard such packets). Finally, we point out that (i) the key K_{i+1} is agreed by resorting to a packet exchange encrypted by means of a secret key, and (ii) such a negotiation does not reveal any information about the new session key. For these reasons, we deduce that the authentication condition is preserved along the conversation lifetime.

B.3.c Integrity. As far as the integrity is concerned, the following remarks are in order. As a first result, we argue about the correctness of the algorithm, and then we show that an adversary cannot alter the content of the conversation obtained by applying such an algorithm. In a first simplified scenario we assume the system model without malicious parties. We consider a packet P_j^* generated by the sender and arriving at the receiver site in time for its playout. As the trusted parties share the same session key, the receiver can (i) compute the timestamp in order to schedule the playout instant of the packet, (ii) compute M_j in order to playout the audio packet and (iii) check the MAC, in order to verify the integrity of M_j. The effect of this behavior cannot be altered by an adversary and we prove this fact by considering the potential moves of a malicious party. We assume the audio packets generated by the sender and managed by the receiver as seen in the above algorithm, and we show that all the played out packets can be neither generated nor altered by an adversary with the capabilities specified in the threat model. In the case the adversary eavesdrops, captures, drops or delays a packet P_j^*, then the proof is trivial. In fact in these cases the adversary can only prevent the receiver from receiving or playing out P_j^*. The most interesting case arises whenever the adversary tries to alter P_j^*. In particular, he can alter the encrypted timestamp, the plaintext M_j, or the MAC, but in this case the receiver notices the alteration by verifying the MAC and therefore he discards the packet. Note that it is computationally infeasible, given a packet P_j and the message authenticating code $MAC(K_i, P_j)$ to find another packet P_j' such that $MAC(K_i, P_j) = MAC(K_i, P_j')$. On the other hand, the adversary cannot send a new packet P_j to the receiver, because he knows neither the session key nor the playout instant of the audio sample M_j he intends to forge.

IV. Experimental Assessment

A working prototype of the secure audio control scheme illustrated in the previous section, called BoAT, was implemented in 1999 using the C programming language and the development environment provided by both the Linux and the BSD Unix operating systems. In this section we present the results we have derived by analysing such a mechanism under the assumptions of the previous section. After this, we present also the performance of some software tools which offer secure audio communications over the Internet. In particular, we consider the audio tools Nautilus [11], PGPfone [29], and Speak Freely [28]. The above methods adopt block cipher algorithms in order to encrypt each audio packet to be transmitted along the network. More precisely, they employ some well-known cryptographic algorithms such as DES, IDEA, Blowfish, and CAST (see [24] for the technical details of these algorithms). The experiments have been conducted with a 133 MHz Pentium processor, 48 MB RAM, ISA Opti 16 bit audio card, and a 200 MHz MMX Pentium processor, 64 MB RAM, PCI Yamaha 724 audio card. The workstations have been connected by means of two 10/100 Mbit Ethernet network cards. Both Linux (RedHat 6.0) and Windows 98 operating systems, where available, have been used.

In order to provide the reader with an understanding of the reported values, we first specify the scenario in which such results have to be considered.

From a performance standpoint, an efficient coding of the signal is the first factor to consider, in order (i) to work with the available transmission rates over networks, and (ii) to obtain the same speech quality as generated at the sender site. As an example, telephone quality of speech needs 64 Kbits, but in most cases such a bandwidth is not reachable over the Internet. Codecs are used in order to cope with this lack, but as the compression level increases (and the needed bandwidth decreases), the generated speech degrades itself, by turning misunderstandable. In general, a trade-off exists between the quantity of data to be encrypted (specified by the particular codec) and the quality of the transmitted speech. In the case of BoAT, the used codec guarantees high quality at a sending rate of about 850 Bps, corresponding to 25 34-bytes long packets per second of conversation.

As far as the cryptographic algorithms are concerned, the following remarks are in order. The particular stream cipher we have considered in our experiments is the RC4 algorithm [24], whereas the message authenticating code of each packet is computed as the encryption of the output of the MD5 message-digest algorithm [20]. The packets of the handshaking protocol are encrypted by using the block cipher Blowfish [24], and the temporal interval between two consecutive synchronizations is exactly one second.

A. Performance Results

In Table I we report the computing time (expressed in ms) experienced during a second of audio conversation by a sending site that follows the scheme presented in Sect. III, by singling out the different steps of the mechanism:

- encryption of the handshaking packets by means of the block cipher,

- encryption of the audio packets by means of the stream cipher,
- computation of the MAC.

The results of Table I put in evidence the following facts. The overall computing overhead is negligible (equal to few tens of μsec). The extremely low use of the block cipher, which is used for the packets of the handshaking phase only, justifies the almost null computational cost deriving from such an operation. Substantially, we note that the computational overhead is equally divided between the encryption step, performed resorting to the RC4 algorithm (whose performance is about 13.7 MBps), and the authentication step, performed resorting to the MD5 algorithm (whose performance is about 17 MBps). It is worth noting that compatible results on the performance of RC4 and MD5 are also presented in [3], [5], [25], [27]. We point out that we have not considered SEAL-like stream ciphers, because from the performance viewpoint these algorithms do not seem to be appropriate if the key needs to be changed frequently.

Summarizing, the results put in evidence the negligible computational overhead of the implemented security infrastructure, in particular with respect to the overall latency due to the adaptive audio control mechanism, equal to tens of ms, as shown in [1], [16], [18].

TABLE I
COMPUTATIONAL OVERHEAD OF THE SECURING MECHANISM OF BOAT PER SECOND OF CONVERSATION.

	Computing Time (ms)
Block Cipher	0.008
Stream Cipher	0.0591
MAC	0.0474
Total Latency	0.1145

B. Comparison

In this section we report the experimental results for some well-known tools designed for the secure audio transmission over the Internet, namely Nautilus [11], PGPfone [29], and Speak Freely [28]. In particular, the results are obtained by employing the same architecture presented in Sect. IV. As far as the software tools are concerned, the following remarks are in order. Nautilus provides the Unix version only, released in 1996; PGPfone 2.1 and SpeakFreely 7.1 were released in the last two years: the former tool is available for Windows 9x operating systems only, whereas the latter one is available for both Unix and Windows 9x. Each tool employs codecs in order to reduce the quantity of data to be transmitted (as in the case of BoAT), and block ciphers for the encryption/decryption of data.

The codecs implemented in such tools offer different tradeoffs among efficiency of compression, loss of fidelity in the compression process, and the amount of computation required to compress and decompress, and last but not least, they determine the length of data that have to be encrypted and transmitted along the network. We recall that the codec activity is very important in order to establish a suitable QoS. For instance, due to the fact that the Nautilus release is 5 years old, it has been developed considering data rates up to 14400 Bps, so it has been equipped with codecs which have a high compression level in order to cope with low bandwidth, but this choice is paid at the cost of poor quality of the transmitted speech. In this paper we are interested neither in presenting and measuring the performance of the different codecs nor in evaluating the trade-off between QoS and efficiency of such compression mechanisms.

In order to provide the reader with a better understanding of the reported results, we just point out the following remarks on the considered codecs. ADPCM offers toll quality of speech at the cost of a high quantity of data to be encrypted and transmitted; the different versions of the GSM codecs offer high speech quality in spite of a higher compression level (we point out that the performance and quality features of the GSM and BoAT codecs are very close); finally, LPC-10 offers poor speech quality with the maximum level of compression. A summarization of the quantity of data compressed during a second of audio transmission by the different codecs embedded in each tool is reported in Table II; for a comparison, we recall that in the case of BoAT the codec works at about 850 Bps, corresponding to 25 34-bytes long packets per second of conversation.

TABLE II
AUDIO PACKET DIMENSION AND NUMBER OF TRANSMITTED AUDIO PACKETS PER SECOND OF CONVERSATION FOR EACH CODEC.

	Speak Freely 7.1	
	GSM	ADPCM
Bytes per packet	336	496
Packets per sec	5	8.4

	PGPfone 2.1	
	GSM 4.4	ADPCM
Bytes per packet	70	327
Packets per sec	14	13

	Nautilus
	LPC-10
Bytes per packet	56
Packets per sec	5.5

The experimental results are shown in Tables III to V; for each block cipher implemented in the different tools the tables report the computing time experienced during a second of conversation by the encryption step (the reader interested in a survey of these ciphers should refer to [24]). It is worth mentioning that all the results are obtained as mean values of repeated experiments, whose individual duration is 30 sec, and the relative variancy is reported too.

The first interesting point illustrated by our tables is that in all cases the computational overhead of the privacy mechanisms is restricted to few milliseconds (the upper bound is represented by the case of Speak Freely with the block cipher DES and the codec ADPCM in Table III with 20.8 ms). If we observe these results and the ones reported in Sect. IV-A, we can conclude that the securing mechanism of BoAT outperforms the other tools; in particular BoAT turns out to be about 2 orders of magnitude better than the other tools (tens of μsec w.r.t. few milliseconds). This because our mechanism adopts a lightweight ciphering mechanism that is very adequate when integrated with the original handshaking protocol.

TABLE III
SPEAK FREELY 7.1 (WINDOWS 98)

Computing Time (ms)	CODEC			
	GSM		ADPCM	
	Mean	Variancy	Mean	Variancy
Blowfish	2.47	0.01	5.22	0.15
IDEA	3.94	0.01	9.08	0.05
DES	9.77	0.20	20.8	0.16

A comparison among the performance of the different tools strictly depends on the particular codec that is used to compress data. For instance, it may be significant to contrast the performance of PGPfone implementing GSM and BoAT, as the related codecs offer the same QoS and the same quantity of data to be encrypted and transmitted per second of conversation (850 bytes in the case of BoAT and 980 bytes in the case of the PGPfone GSM 4.4). The results (about 0.1 ms for Boat and about $2-7$ ms for PGPfone) confirm once again our claim that BoAT outperforms the other tools. As far as the results obtained with other codecs are concerned, it is worth mentioning that the good performance offered by Nautilus with LPC-10 (see Table V) depends on the fact that such a codec uses a high compression factor (note that the output of the LPC-10 compression algorithm per second of audio conversation is few hundreds of bytes). In particular the speech quality offered by this codec is noticeably poorer than the high quality guaranteed by the mechanisms of the other tools. An interesting remark is in order in the case of the ADPCM codec implemented in PGPfone and Speak Freely (see Table IV and III). Indeed for such a codec we can observe an overhead of the encryption phase of several ms, especially in the case of 3DES, because it offers toll quality of the transmitted speech in spite of a low compression level (thousands of bytes per second of audio conversation).

To conclude this section, we can summarize the obtained results by observing that the nature of BoAT (in particular the handshaking protocol which allows the two parties to share the session keys) seems to be very suitable to extend the original mechanism with security features in a natural and cheap way. Hence, adding security modules to the audio data flow pipeline may be done without compromising the overall end-to-end delay, because the presented approach turns out to be neither a noticeable computational penalty nor a performance bottleneck in real time speech traffic. As far as other well-known tools are concerned, the performance results put in evidence that the computational overhead of the security platform is limited to few milliseconds, and that such a result is about 2 orders of magnitude worse than the performance of BoAT.

TABLE IV
PGPFONE 2.1 (WINDOWS 98)

Computing Time (ms)	CODEC			
	GSM lite 4.4		ADPCM	
	Mean	Variancy	Mean	Variancy
Blowfish	2.09	0.06	4.72	0.02
CAST	2.08	0.002	4.43	0.07
3DES	6.35	0.14	16.8	0.56

TABLE V
NAUTILUS 1.5A (LINUX REDHAT 6.0)

Computing Time (ms)	LPC-10	
	Mean	Variancy
Blowfish	0.32	0.0004
IDEA	0.48	0.004
3DES	0.84	0.0009

V. CONCLUSION

In this paper we have presented a control mechanism for the transmission of real time audio over the Internet which offers:
- a packet audio control algorithm that adaptively adjusts to the fluctuating network conditions in order to maximize the QoS,
- a complete security infrastructure providing authentication of the parties, privacy and integrity of the transmitted data.

On the one hand, the original adaptive playout adjustment scheme offers good performance close to the optimum. In particular, as stressed in recent works ([16], [1]), packet audio control algorithms are optimizable with difficulty because there are intrinsic limits for improving their QoS (e.g. the traffic conditions do not allow us to reduce the playout delay without compromising the human perception of transmitted speech), and at the same time, the QoS they guarantee is borderline, because an higher overall latency cannot be tolerated by the real time constraints typical of such kind of services.

On the other hand, the adaptive playout adjustment scheme has revealed its adequacy to include security services in a simple way and with a negligible overhead. This because we have chosen to exploit the features of the existing adaptive mechanism, instead of e.g. treating the packet audio control mechanisms and the security mechanisms as separate layers in the protocol hierarchy. From the performance standpoint, the adequacy of our algorithm is put in evidence also in the experiments. As an example, an interesting summarization of the results of Sect. IV is reported in Table VI, where we show the computing time experienced by both encryption (at the sending site) and decryption (at the receiving site) during a second of conversation.

TABLE VI
PERFORMANCE COMPARISON

	Computing Time (ms)
BoAT	0.229
Speak Freely	19.54
PGPfone	12.7
Nautilus	1.68

In particular, in such a table we consider the tools BoAT, Speak Freely with the codec GSM and the block cipher DES, PGPfone with the codec GSM lite 4.4 and the block cipher 3DES, and Nautilus with the codec LPC-10 and the block cipher 3DES. The results reveal that (i) in general the provision of security has a computational cost of few milliseconds, and (ii) in particular BoAT performs better than the other tools (tens of μsec w.r.t. few milliseconds).

Acknowledgements This research has been funded by Progetto MURST Cofinanziato TOSCA and by a grant from Microsoft Research Europe.

REFERENCES

[1] A. Aldini, M. Bernardo, R. Gorrieri, M. Roccetti, *"Comparing the QoS of Internet Audio Mechanisms via Formal Methods"*, Tech. Rep. UBLCS-99-04, University of Bologna (Italy), to appear in ACM Trans. on Modeling and Computer Simulation (TOMACS), 2001, `ftp://ftp.cs.uribo.it/pub/techreports/99-04.ps.gz`

[2] J-C. Bolot, A. Vega-Garcia, *"Control mechanisms for packet audio in the Internet"*, IEEE Infocom'96, San Fransisco, CA, 1996

[3] A. Bosselaers, R. Govaerts and J. Vandewalle, *"Fast hashing on the Pentium"*, Advances in Cryptology, Proceedings Crypto'96, LNCS 1109, N. Koblitz, Ed., Springer-Verlag, pp. 298-312, 1996

[4] M. Briceno, I. Goldberg, D. Wagner, *"A pedagogical implementation of A5/1"*, 1999, `http://www.scard.org`

[5] A. Bosselaers, *"Even faster hashing on the Pentium"*, presented at the rump session of Eurocrypt'97, 1997

[6] J. Boyce, R. Gaglianello, *"Packet Loss Effects on MPEG Video Sent Over the Public Internet"*, ACM Multimedia 98, 181-190, Electronic Edition, 1998

[7] A. Biryukov, A. Shamir, D. Wagner, *"Real Time Cryptanalysis of A5/1 on a PC"*, in Fast Software Encryption Workshop, NYC, 2000

[8] A. Canteaut, M. Trabbia, *"Improved Fast Correlation Attacks using Parity-check Equations of weight 4 and 5"*, Eurocrypt'2000, Bruges, 2000

[9] L. Cottrell, W. Matthews, C. Logg, *"Tutorial on Internet Monitoring & PingER at SLAC"*, Stanford Linear Accelerator Center, 2000

[10] D. Dolev, A. C. Yao, *"On the Security of Public Key Protocols"*, IEEE Transactions on Information Theory, 29(2), 1983

[11] B. Dorsey *et al.*, *"Nautilus Documentation"*, 1996, `http://www.lila.com/nautilus/`

[12] E. Filiol, *"Decimation Attack of Stream Ciphers"*, in Cryptology ePrint Archive, Report 2000/040, 2000

[13] V. Hardman, M.A. Sasse, I. Kouvelas, *"Successful Multi-Party Audio Communication over the Internet"*, in Comm. of the ACM 41:74-80, 1998

[14] Internet Engineering Task Force, *"IP Security Protocol"*, in Proc. of the 43 IETF Meeting, 1998, Internet-Drafts available at `http://www.ietf.org`

[15] N. Jayant, *"Effects of Packet Loss on Waveform Coded Speech"*, in Proc. of Fifth Int. Conference on Computer Communications, pp. 275-280, 1980

[16] S.B. Moon, J. Kurose, D. Towsley, *"Packet Audio Playout Delay Adjustment: Performance Bounds and Algorithms"*, in ACM Multimedia Systems 6:17-28, 1998

[17] A. Perrig, R. Canetti, D. Tygar, D. Song, *"Efficient Authentication and Signing of Multicast Streams over Lossy Channels"*, in Proc. of IEEE Security and Privacy Symposium, 2000

[18] R. Ramjee, J. Kurose, D. Towsley, H. Schulzrinne, *"Adaptive Playout mechanisms for Packetized Audio Applications in Wide-Area Networks"*, in Proc. of *INFOCOM '94*, 1994

[19] R. L. Rivest, A. Shamir, L. M. Adleman, *"A method for obtaining digital signatures and public-key cryptosystems"*, Communications of the ACM, 21(2): 120-126, 1978

[20] R. L. Rivest, *"The MD5 Message-Digest Algorithm"*, MIT LCS & RSA Data Security, Inc., 1992

[21] M. Roccetti, *"Secure Real Time Speech Transmission over the Internet: Performance Analysis and Simulation"*, in Proc. of 2000 Summer Computer Simulation Conference (SCSC'2000), (B. Waite, A. Nisanci Eds.), The Society for Computer Simulation International, Vancouver (CA), pp. 939-944, 2000

[22] M. Roccetti, V. Ghini, G. Pau, P. Salomoni, M. E. Bonfigli, *"Design and Experimental Evaluation of an Adaptive Playout Delay Control Mechanism for Packetized Audio for Use over the Internet"*, Tech. Rep. UBLCS-98-04, University of Bologna (Italy), to appear in Multimedia Tools and Applications, an International Journal, Kluwer Academic Publishers, 2001, `ftp://ftp.cs.uribo.it/pub/techreports/98-04.ps.gz`

[23] M. Roccetti, V. Ghini, D. Balzi, M. Quieti, *"BoAT: the Bologna optimal Audio Tool"*, University of Bologna (Italy), 1999, `http://radiolab.csr.uribo.it/BoAT/src`

[24] B. Schneier, *"Applied Cryptography, 2nd Edition"*, John Wiley & Sons, 1996

[25] B. Schneier, D. Whiting, *"Fast Software Encryption: Designing Encryption Algorithms for Optimal Software Speed on the Intel Pentium Processor"*, Fast Software Encryption, Fourth International Workshop Proceedings'97, Springer-Verlag, pp. 242-259, 1997

[26] H. Schulzrinne, *"Voice Communication across the Internet: a Network Voice Terminal"*, Tech. Rep., University of Massachusetts, Amherst (MA), 1992

[27] J. Touch, *"Performance Analysis of MD5"*, in Proc. of Sigcomm '95, Boston MA, 1995

[28] J. Walker, B. C. Wiles, *"Speak Freely"*, 1995, `http://www.fourmilab.ch`

[29] P. Zimmermann, *"PGPfone: Owner's Manual"*, 1996, `http://www.pgpi.com`

Instant Messaging and Presence for SIP Enabled Networked Appliances

Arjun Roychowdhury
Hughes Software Systems
India
archow@hss.hns.com

Stan Moyer
Telcordia Technologies
USA
stanm@research.telcordia.com

Abstract - There has been much interest in enabling networked appliances (NAs) to communicate with and control each other over the Internet. Although there are numerous networking technologies which allow networked appliances to operate within a single home environment or domain, there is currently no support for control of these devices from the Internet, or for interworking the various home networking technologies. The ability to provide such support will radically enhance the ability to provide exciting new services. The new age of interworking will converge both general devices (like PDAs, laptops) and specialized devices (like phones, washing machines) into one date network, enabling services to the end customer that were unthinkable a few years ago. This paper discusses the use of the Session Initiation Protocol (SIP) to provide wide area networking for appliances. In the first half it describes the motivation behind using SIP for such appliances. Following this, we illustrate how services can be provided to the consumer using Instant Messaging and Presence in the networked appliance domain.

A. INTRODUCTION

Network enabling home appliances in not a new field. For a few years now, various companies and bodies have come out with a multitude of technologies which enable networked appliances (NA) to talk to each other. The simplest of the lot is X.10, which allows any user to control his home devices using an X.10 controller which can send electric pulses and put on/put off or change voltage level of certain appliances in and around the house. Then came more complicated and feature rich protocols like Jini [1], UPnP [14], HAVi [3], and other such protocols which added a lot more of capabilities for appliance communication. These existing protocols are mainly designed to work in a single home environment (i.e. local area network) and do not scale for Internet-wide communications and are lacking in the necessary security mechanisms. Home appliance interworking has most been restricted to research labs and limited customer deployment. The primary reasons for the above are:

Interoperability – It is imperative that the underlying protocol used is hidden from the consumer. He should be able to buy an Internet alarm clock from his local store and buy a coffee machine from somewhere else and have the alarm clock start the coffee maker at exactly 8:00 am every day without worrying about whether they both speak the same protocol.

Scalability – The protocol(s) in question must be able to scale across large areas. If Robert goes on a vacation to India and realizes that he has forgotten to program his VCR to record every episode of Friends, he should be able to do so without worrying about whether the underlying protocol can handle the distance. The protocol should be able to traverse multiple hops from the source to the destination, similar to IP routing and in that process not load in-between networks (thus protocols which rely solely on broadcast and multicast would be a bad idea)

Security – This is probably the single most important issue of interworking appliances. The breaches of security that can occur here are of much greater consequence than simple data piracy or data corruption. Human lives are of concern here. By breaking into a home network, it becomes possible to control every appliance that was connected to produce disastrous results. The protocol under consideration not only must be able to provide strong encryption and authentication mechanisms but it must also allow a user to selectively allow or disallow appliance sharing, based on his discretion.

Limited Services – How much bandwidth is available for home appliance communication ? How much more bandwidth is required for wide area appliance communication? In the past, a significant amount of work has gone into trying to make the protocol as efficient as possible to work within strict bandwidth constraints. A lot more concentration has gone into optimization rather than feature provisions. As a result, the services provided were limited in nature and possibly not exciting enough for the consumer to consider very seriously. However, now with broadband technologies coming into the mainstream, this is less of a concern.

Currently, SIP has emerged as a protocol of choice in the VoIP space as an enabler for next-generation services. Based on HTTP, SIP is a lightweight protocol that brings along with it a host of web enabled services

to internet telephony — a space which until now was dominated by heavier and more complicated protocols that made next generation service deployment more challenging. We believe that using SIP for network appliances would enable service provides to quickly and easily introduce more value added services to end consumers.

This paper describes an approach based on SIP for application-layer communications with networked appliances. The paper is a summary of recent Internet Drafts [1][5][7][12] related to this topic (many of which the authors co-wrote). The next section provides a definition or a networked appliance. Then advantages for using SIP

B. WHAT IS A NETWORKED APPLIANCE ?

Before we proceed, let us first describe what we mean by a *networked appliance:*
Networked appliances (NAs) are regarded as networked devices which share the following characteristics:

1 *The ability to interact with the physical environment (either through sensors or actuators);*

Limited (restricted) general computational power, though the devices may possess high-computation power for specific tasks such as image processing, or speech recognition.
In general, NAs will be groups of sensors and/or actuators with a limited computational power, and networking capability. Examples of NAs include: controllable lamps, alarm clocks, SIP phones, washing machines, door bells, cameras, temperature sensors, sprinklers, and garage doors.

C. THE SIP ADVANTAGE

The following attributes of SIP make it an idea choice for use with appliances.
Scalability – SIP is a very scalable protocol. It works well in both LAN and WAN conditions. It does not rely solely on multicast/broadcast technologies to reach a destination endpoint. It has a strong concept of **routing** (similar to HTTP routing) which enables a packet to traverse from source to destination using intermediate existing routes, hopping from one node to another till it reaches its final destination. Further SIP can work on both UDP and TCP which allows SIP based servers to scale well. This coupled with the fact that SIP allows various definitions of servers, ranging from stateless proxies (which simply forward all messages without keeping state) to call stateful proxies (that keep state of all SIP based calls that route through them) enables service providers to select the ideal combination they need based on network load and services to be provided.
Simplicity – An important attribute for any protocol is that it must be simple to provide value added services. The protocol should enable service providers to rapidly deploy new services without getting lost in the mires of implementation. This has a direct relation to how flexible and simple a protocol is. SIP is very lightweight and is text based – implying that both message communication and formulation are simple and non-intensive in nature.
Flexibility – In SIP it is very simple to add extensions to support new features. The protocol is defined in a way that any provider can define extensions easily to the existing grammar set to add features which may not exist in the core SIP specification yet. Most importantly, the protocol also specifies mechanisms which can ensure that any new extension does not break an existing SIP aware node that may be in route and which does not understand this new extension (For example, proxies are supposed to transparently pass bodies across. Further, if it does not understand any header, it should pass it along to the next node)
Registration – In SIP is it not necessary that a calling device needs to know exactly where to locate the called device. A device can REGISTER its current location with its registrar. Bob simply specifies that he wants to talk to Peter. The exact present location of Peter will be resolved by the proxy in conjunction with the registrar to ensure that the message reaches Peter wherever he is now. Location independence is important for appliances too.
Personal Mobility – Related to the point above, SIP provides the concept of personal mobility at no extra cost. Again, this is a key service enable for appliances too. Consider an example of Bob trying to remotely control his VCR to record a program at 7pm. He does not know that his wife has moved the VCR from his den to their bedroom that very day in order to re-organize the home décor. He simply sends the request to `vcr@home.bob.net` and the residential proxy server in his house knows that `vcr@home.bob.net` is now an alias to point to `vcr@bedroom.home.bob.net`. This happened when the VCR was shifted – it automatically registered with the new location with its local registrar.
Security – SIP provides both authentication and encryption using schemes such as basic, digest and PGP to provide end-to-end security. Security is an important aspect of appliance interworking for both safety and privacy reasons.

Transport Independence – SIP does not rely on any fixed transport protocol. That essentially means that SIP messages can traverse through various heterogeneous networks. So if Bob wanted to control his home VCR from office, he did not have to bother about the fact that this SIP message might have actually traversed various intermediate networks based on TCP, UDP, SCTP and other such transport protocols. This is important since NAs may be not have much CPU power and little memory and therefore only implement UDP (and not TCP).

Event Notification – SIP has recently been extended to introduce SUBSCRIBE and NOTIFY messages which enable elements to "subscribe" to certain events and be notified when they occur. For example, Bob could also tell his office laundry washing machine "Start tumble washing , *and let me know when you finish* so that I can come on down and pick up my clothes". Event notification forms a very important part of Instant Messaging and Presence, which we discuss later.

Addressing – SIP uses URIs for addressing. This may be SIP URLs, HTTP URLs, Tel URL or other schemes that map into the generic URI grammar. This means that we can introduce any new addressing scheme like SLP or others by mapping it either to an existing scheme of URLs (see [7] for mapping SLP URLs to SIP URLs). We believe that the URI scheme is an extremely generic and flexible addressing mechanism that can encompass a very wide range of addressing requirements, that may well be the case for appliances.

Integration With Existing SIP service mechanisms – using SIP as a protocol enables appliances to exploit the same service rich infrastructure that SIP provides. For example, Peter could send a request saying "*Could any printer print my document please ?"* to his office proxy which could then fork this request to the various printers in the office and then allot the job to any available printer.

Session based and Non-session based communication – If Bob wants to remotely ask his close circuit TV to start transmitting the video capture in room 28 of the supermarket, he essentially needs to set up a regular receive-only session with the circuit TV which then starts to stream video to his browser QuickTime plug-in. This is no different from the regular SIP INVITE session establishment. On the other hand, Dilbert may want to simply send a command to his internet alarm clock to set a time for wakeup and forget about it. No session is required for this – this is no different from the Instant Messaging concepts in SIP. SIP allows us to communicate using both session based and non-session based transactions, which is ideal for appliances.

Business Models – Finally, since SIP and the HTTP are so similar in nature, using SIP would introduce the same possibilities of different service models that the Browser-Server technology have had success in introducing to the consumer market – that of different business models for products that were not possible without the Internet – the ASP and pay per use models. In fact, these model are already being experimented in home appliances with Electrolux introducing their "pay-per-use" washing machines (See [16])

D. INSTANT MESSAGING AND PRESENCE

Now that we have established a foundation of why SIP seems to be an appropriate protocol for appliances, we shall proceed to illustrate how one can use concepts of IM to enable services for NAs.

The authors assume that the reader is aware of the Instant Messaging extensions for SIP and their model of operation (for details, please see [8]) The applicable extensions are a new SIP message type, MESSAGE [9], and the use of the previously proposed new message types SUBSCRIBE and NOTIFY [10][11].

Basic Communication

Most often, the act of asking a device to perform a certain operation consists of sending it a message with the command embedded in the body of the message. The requisites for such an operation are: a) Sending the command to the destination using any available route, and b) Receiving an acknowledgement that the destination has received the message.

Hence, simple communication essentially consists of a 2 way transaction exchange. For the purpose of such communication, we have defined an extension to SIP methods called DO (see [5]) which allows the sender to encode a message in its body using an XML like format. This message is proxies across intermediate hops and finally reaches the destination. Using SIP processing rules, the response too traces back the same path and reaches the source. *Note,* The operation of the DO command is strikingly similar to the MESSAGE command defined in [9]. The reason for the separation of commands was mainly due to concerns within the SIP community of overloading the MESSAGE command too far. A disagreement exists asa to whether instant messaging a person is equivalent in semantics to instant messaging a device. Since creating another message type does not add additional complexity for SIP Proxy or User Agent implementations, it was agreed to use a

different message type. This is still a debatable issue – however the intent of DO remains the same as that of MESSAGE and using MESSAGE instead of DO would not change any of the concepts discussed in this paper.

An example:
Bob, sitting in his office wants to start the self diagnostic check in his new internet enabled car. In case of any damage detected by the car microprocessor, he wants to be informed so that he can set up an appointment with the repairman.

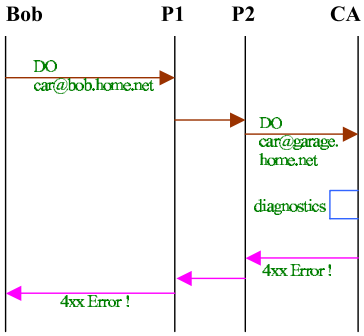


Figure 1: Example of IM

The illustration above shows a very simple case of how IM forms the basic block for communication to appliances. Here, Bob sends a DO message to his car parked in the garage. The office proxy (P1) recognizes the domain to be `bob.home.net` and forwards it to the proxy in his house. The home proxy (P2) now looks up the registration database and sees that the car is now inside the garage. It proxies this request to the correct location. The internet enable microprocessor in the car, on receiving this message proceeds to run the self diagnostics and sees that its wheel alignment needs to be corrected. It now reports this diagnostic error in a 4xx message back to Bob who can then proceed to take corrective action. (Note: in case the diagnostic process takes time, it can also return a 100 trying response which is not shown in the illustration).

Instant Messaging forms the very basic block for appliance communication. Other examples of IM for devices include querying your thermostat for the current temperature, sending a command to your microwave to start thawing the steak or asking your VCR at home to start recording you favorite program while you are sitting in the office trying to finish off your presentation for that important meeting the next day. This allows both control of devices (e.g., "turn on") and querying (e.g., "what is the temperature in the living room?").

In all of the above, what we really need is a simple two way transaction that can traverse multiple hops and routes to where you want it to and have the results routed back to you. Isn't IM then, just what we need ?

IM combined with Presence
We now show some powerful examples of how IM and presence combined can provide enhanced services.

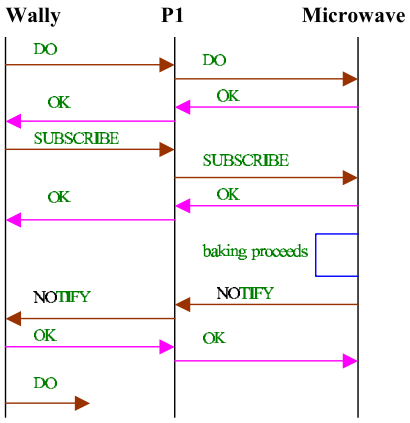


Figure 2: IM and presence

Here, Wally sends an instant message from his PC browser to his microwave asking it to start grilling the chicken at power level 5 until it is well done. He encapsulates this in a DO command body. The microwave responds by saying that the process has begun. This is a time consuming operation. Wally wants to be notified when the microwave finishes this process. So, he sends a SUBSCRIBE message to the microwave, asking it to notify him when the grilling gets done. In this case, Wally has subscribed to the microwave state, asking it to be notified when the grilling gets done. In IM terminology, Wally is the SUBSCRIBER and the microwave is the PRESENTITY.
The microwave responds to the request by sending a 200 OK. Now, Wally does not need to poll the microwave anymore. He proceeds to carry on with his usual work, while the microwave grills away. Some time later, the microwave completes the process and looks up the subscribers who have subscribed to this change of state. It realizes that `wally@home.net` was a subscriber and proceeds to send a notification message to this URI

about this state change. This is an asynchronous message that Wally receives on his browser thorough a popup which says "*Microwave has finished grilling. time taken was 2 minutes".* Wally now proceeds to send the next set of commands to the microwave.

This is a very powerful form of communication and very useful for communications with certain types of devices. For example, notification of events that occur in the home such as a security alarm being triggered or a doorbell being pressed. Presence forms a very important part of appliance communication. It provides a mechanism for different appliances to subscribe to state information of other appliances and be notified asynchronously when any event occurs.

E. EXTENSIONS REQUIRED

As mentioned previously, SIP as specified in RFC 2543 does not support Instant Messaging for networked appliances. Some extensions/modifications are required and are described in this section.

A new method, DO [5] is required to support simple, datagram-style messaging and the SUBSCRIBE and NOTIFY extension [10] are required for asynchronous event notifications.

To better meet the naming and addressing requirements for NAs, a modified SIP URL addressing scheme is required. A structured device naming scheme (e.g. Service Location Protocol (SLP) URL) is encoded to the left of "@" sign in the To: field. This may then be encrypted to ensure privacy. For example:
[d=lamp,r=bedroom,u=stsang]`@home.net`
where the information in the square brackets would be BASE-64 encoded (to be compliant with addressing conventions) and (optionally) encrypted for privacy.

In addition, a new payload type, specific to devices, is required. A new MIME type, called Device Messaging Protocol (DMP) [12], is proposed to carry the information required to excite NAs and which can carry responses back to the originator. There is no reason that other payloads (e.g., SOAP [13]) could not be carried too (either as another MIME type or as part of the DMP).

F. HOW DOES SIP SOLVE THE PROBLEM?

In the introduction, we listed four problems hindering the adoption of networked appliances — these problems are:

- Interoperability
- Security
- Scalability
- Limited Services (i.e., Functionality)

In this section we describe how SIP (and SIP Instant Messaging) address each of these four issues.

Interoperability

The SIP for networked appliances solution enables communication with devices in the local area without concern for the type of local device communication protocol being used. For example, given an architecture like that pictured in Figure 3, there are several types of local device communication protocols depicted in use — UPnP, HAVi, X.10, and SIP. This architecture assumes that an appliance controller exists to provide the necessary protocol translation/interworking.

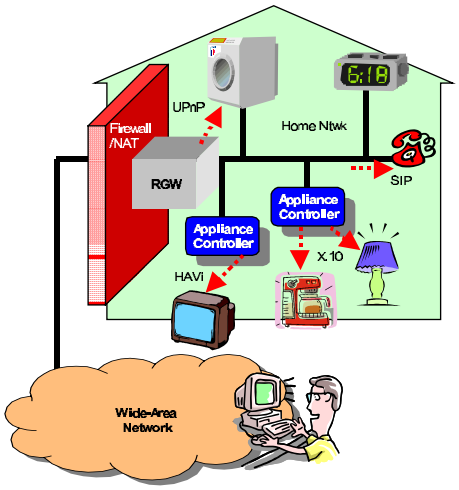


Figure 3. Example Home Networked Appliance Architecture

SIP for networked appliances uses SIP to communicate from the wide area network into the local home domain where the SIP message is then translated to the appropriate local device communication protocol (if necessary). An example of this is shown in Figure 4. The proposed Device Messaging Protocol payload is XML-based and defines a superset of all NA communication protocols — therefore it can be translated by the SIP UA associated with the NA(s) from DMP to the specific protocol for the NA.

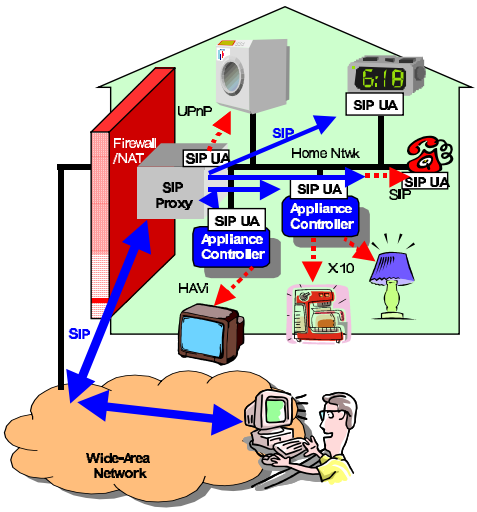


Figure 4. Use of SIP for Networked Appliances

Security
SIP for networked appliances utilizes the inherent capability of SIP to provide the security required for communication with NAs from the wide area. SIP provides an authentication mechanism (PGP is specified in the standard and Internet Drafts exist for supporting CHAP and Radius servers). If all SIP messages entering the local home domain are authenticated than an authorization check can be performed before the requested operation is executed. In addition, SIP provides a means for encrypting most portions of the message (including the payload) for privacy. However, the address is not encrypted, therefore, if the appliance name and/or description is included in the address, then that portion of the address must be encrypted (as previously described).

Scalability
As described in section C, the SIP protocol is very scalable. This property is achieved by its use of routing proxies in the network and the fact that no (or little) state is kept in the network.

Limited Services
SIP is an extensible protocol — it allows new method/message types, new types/forms of addresses, and any kind of MIME body type. Through these capabilities, extensions were designed to enable the SIP for networked appliances solution to communicate with NAs in four different ways:

Control (e.g., "turn on the coffee maker")
Query (e.g., "how much milk is left in the refrigerator?")
Event Notification (e.g., "tell me when my fire alarm goes off")
Multimedia Session (e.g., "View the babysitter cam.")
Currently those are the only four types of communication modes identified for NAs and SIP supports them all. If other modes are required in the future, SIP's extensibility should accommodate the necessary modifications.

G. FUTURE WORK

In order to produce an inter-operable and open framework, detailed designs and specifications of the SIP message payloads for device control and device registration will be produced. The relevance of, and relationship to, other protocols – such as SOAP [11] and UPnP [14] – are being assessed and will be included into the framework described in this paper if appropriate. The possibility of interworking the SIP based service portability framework with the Open Services Gateway Initiative (OSGi [15]) framework is under investigation and the roles and capabilities that can be provided by the home gateway will also be evaluated.

H. SUMMARY

We have shown how SIP with Instant Messaging extensions supports application-layer communication with networked appliances. Many reasons for why SIP is a good solution for this problem were described. However, the real advantage to using SIP is that since SIP phones are essentially networked appliances too, SIP Instant Messaging for NAs can leverage the same SIP infrastructure deployed for Internet Telephony. Leveraging the same infrastructure greatly eases the operational (e.g., deployment, administration, and management) burden for networked appliance-based services.

Acknowledgements

The authors would like to thank S.Tsang, D. Marples, J. Katz, P. Gurung, T. Cheng, A. Dutta, S. Khurana and H. Schulzrinne for their important contributions in defining the usage of SIP with networked appliances.

References

[1] JINI, www.jini.org.
[2] UPnP, www.upnp.org.
[3] HAVi, www.havi.org.
[4] Tsang, Moyer, Marples, Schulzrinne, and Chowdhury, "Requirements for Networked Appliances: Wide- Area Access, Control, and Interworking ", Internet Draft draft-tsang-appliances-reqs-01.txt, September 2000
[5] Tsang, Moyer, Marples, Schulzrinne, and Chowdhury, "SIP Extensions for Communicating with Networked Appliances," Internet Draft draft-tsang-sip-appliances-do-00.txt, November, 2000
[6] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: session initiation protocol," Request for Comments (Proposed Standard) 2543, Internet Engineering Task Force, March 1999.
[7] Moyer, et al, "Framework Draft for Networked Appliances using the Session Initiation Protocol", Internet Draft draft-moyer-sip-appliances-framework-01.txt, November, 2000
[8] Mark Day, J. Rosenberg "A Model for Presence and Instant Messaging", Internet Draft draft-ietf-impp-model-01.txt
[9] J. Rosenberg, et al, "SIP Extensions for Instant Messaging", Internet Draft draft-rosenberg-impp-im-00.txt, June, 2000
[10] A. Roach, "Event Notification in SIP," Internet Draft draft-roach-sip-subscribe-notify-02.txt, November, 2000.
[11] J. Rosenberg, et al, "SIP Extensions for Presence", Internet Draft draft-rosenberg-impp-presence-00.txt, June, 2000
[12] Khurana, Gurung, and Dutta, "Device Message Protocol (DMP): An XML based format for Wide Area Communication with Networked Appliances," Internet Draft, draft-khurana-dmp-appliances-00.txt, November 2000.
[13] N. Deason, "SIP and SOAP", Internet Draft draft-deason-sip-soap-00.txt, June 2000.
[14] Universal Plug and Play, http://www.upnp.org
[15] Open Services Gateway Initiative, http://www.osgi.org
[16] http://interactive.wsj.com/public/current/articles/SB948905716900470386.html

Quality of Service II

ADAPTIVE *Delay aware* ERROR CONTROL FOR INTERNET TELEPHONY

Catherine Boutremans, Jean-Yves Le Boudec
Institute for Computer Communications and Applications
Swiss Federal Institute of Technology at Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
{Catherine.Boutremans,Jean-Yves.LeBoudec}@epfl.ch

Abstract— **Forward Error Correction copes with packet losses, but at the expense of an increase of the end-to-end delay. By failing to take this into account, existing error control schemes for audio often lead to end-to-end delays larger than 150 ms, which has an impact on the perceived audio quality. In this paper, we develop an adaptive error control scheme for audio which is delay aware, i.e. which incorporates the impact on the end-to-end delay in the choice of FEC.**

To this end, we model the perceived audio quality as a function of the end-to-end delay and of the encoding rate at destination. We develop a joint rate/error/delay control algorithm which optimizes this measure of quality and is TCP-Friendly. We show that our scheme increases utility in a single class best effort network. We evaluate the benefit for audio sources to use the Alternative Best Effort service, which offers applications the choice between lower end-to-end delay and more overall throughput.

Keywords— **IP telephony, FEC, end-to-end delay, TCP-Friendly**

I. INTRODUCTION

Real-time, interactive audio over IP networks often suffers from packet loss. Forward Error Correction (FEC) can be used [1] to mitigate the impact of packet losses. However, FEC has the drawback of increasing the end-to-end delay. Indeed, in current systems, FEC requires the destination to wait for several packets to be received in order to repair packet losses.

When used over the standard IP best effort service, an audio source should also control its rate in order to react to network congestion and share bandwidth fairly, in some sense [2], [3]. Recently, Bolot [1] proposed an adaptive rate/error control which optimizes a subjective measure of quality and incorporates the constraint of rate control. This scheme proved to be efficient but it does not try to optimize the overall end-to-end delay; in particular, it does not manage the additional delay due to FEC.

Now, it is recognized that above a certain threshold (around 150ms) the end-to-end delay starts to be annoying [4]. In some cases, it may be that a rate/error control such as [1] causes the delay to be larger than the threshold, whereas it would have been possible to avoid this, at the expense of an increased distortion. If the resulting distortion is still acceptable, then a reduced delay would be preferable. This is our main motivation to propose a delay aware scheme for controlling rate and FEC. Our simulation examples in Section VI-A tend to indicate that our method does avoid that a source waste delay on the FEC when it is not necessary.

A secondary motivation is the emergence of a number of proposals for internet differentiated services which propose to combine performance objectives related to delay and to throughput or packet loss [5], [6], [7], [8]. The proposal in [8] is based on per flow queuing and assumes that sources employ packet pair probing. In this paper, we focus on another proposal, Alternative Best Effort (ABE) [9], which does not require per flow queuing. It offers an application the choice of either a lower end-to-end delay or more overall throughput. In today's internet, where explicit congestion notification (ECN) [10] is not yet widespread, the low delay class is likely to receive more packet loss. This asks the question whether there is a real benefit for an audio application to use the low delay class, since it may be forced to compensate the additional loss by more FEC, and thus more end-to-end delay. We show some simulation results in Section VI-B with audio sources implementing our delay aware control scheme; they tend to indicate that the answer is positive.

In this paper, we propose an adaptive error control scheme for real time audio over best effort networks which is *delay aware*, namely, which chooses the FEC according to its impact on the end-to-end delay. We start by considering the perceived audio quality as a function of the audio encoding rate received at the destination *and* of the end-to-end delay. Then we write our control problem as an optimization problem, and solve it numerically and theoretically. We also incorporate a TCP-friendly module, in order to ensure that our rate conforms with current practice for the Internet. This gives us the basis for a rate/error/delay control algorithm which (1) optimizes our delay-aware measure of audio quality and (2) is TCP-Friendly.

This paper constitutes an extension of the work reported in [1] where a rate/error control problem was first formulated as an optimization problem. However, our work brings two new contributions. First and foremost, we consider the end-to-end delay in the optimization problem. Second, we introduce a TCP-Friendly module to control the rate of the audio stream. We also provide details about the resolution of the optimization problem ([1] did not explain in details how the optimization was solved).

The validation of a particular measure of perceived audio quality is outside the scope of this paper. Therefore, we increase robustness by plugging into our method several different quality measures.

The paper is organized as follows. Section II reviews the state of the art on error control for audio applications and TCP friendly rate control. Section III describes our utility function approach. Section IV presents our main result, namely, the delay aware error and rate control. An auxiliary component for implementing TCP-friendliness is described in Section V. Results of simulations implemented in ns2 [11] are given in Section VI-A (today's internet) and Section VI-B (ABE network). Details

about the optimization problem and its solution are in appendix.

II. Related work

A. Error recovery

As best-effort networks do not provide any guarantee in terms of Quality of Service, the end-points in an audio transmission must be able to recover from packet losses. To this end, an audio transport protocol may cope with packet losses by [12]: (1) retransmitting dropped packets, (2) applying Forward Error Correction (FEC) to reconstruct the missing packet, or (3) using error-concealment algorithms to correct the losses.

Retransmission algorithms based on Automatic Repeat reQuest (ARQ) have been successful in protocols like TCP, but they are typically not acceptable for real-time audio applications since they dramatically increase the end-to-end delay. FEC, on the other hand, is an attractive alternative to ARQ since it provides relatively low-delay performance. The principle of FEC is to send redundant information, along with the original information, so that lost data can be recovered, at least partially, from this redundant information.

Originally, there was much interest in the provision of *media independent* FEC using block codes (*e.g.* based on Reed-Solomon [13] or on parity codes [14], [15]) to provide redundant information. Unfortunately, these techniques have the disadvantage of introducing important additional delays (because a source must wait for the entire block of packets before computing and transmitting the redundancy packet). These schemes can therefore be used for one-way, near real-time audio transmission but are not suitable for interactive audio communications.

One way to avoid this coding delay is instead of sending redundant information rely upon *error-concealment* algorithms at the receiver to correct the effect of the missing packets [16]. The most simple error-concealment techniques replace the missing audio unit with silence, white noise or a repeated segment. As such, these techniques work well for relatively small loss rates ($\leq 10\%$) and for small packets (4-40ms of audio) but they break down when the loss length approaches the length of a phoneme (5-100ms). More sophisticated concealment techniques, such as Adaptive Packetization and Concealment (AP/C)[17], exploit the network loss characteristics and the property of long-term correlation within a speech signal together, to mitigate the impact of packet losses. This is accomplished by an adaptive choice of the packetization interval of the voice stream at the sender. Besides, modern speech codecs, such as G.729 [18], use concealment algorithms that are defined as part of their specification. These algorithms interpolate the missing codec parameters based on surrounding and previous values. These techniques lead to a good quality of the reconstructed speech if the number of consecutive lost frames is small and if the loss does not occur at an unvoiced/voiced transition [19]. Hence, error-concealment schemes should not be regarded as substitutes for FEC, but rather a combination to the latter.

Most audio conferencing tools use a *media specific* FEC scheme that combines error-concealment and FEC. The principle of the *media specific* FEC is to transmit each segment of audio, encoded with different quality coders, in multiple packets. When a packet is lost, another packet containing the same segment (maybe encoded differently) can be able to cover the loss. This approach has been advocated by Hardman *et al* [16] and Bolot *et al* [20] for use on the Mbone, and extensively simulated by Podolsky *et al*. [2].

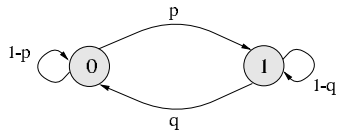


Fig. 1. *The Gilbert model*

The first transmitted copy of the audio segment is referred as primary encoding and subsequent transmissions as secondary encodings. In the *media specific* FEC scheme, redundant audio segments are piggy-backed onto a later packet which is preferable to the transmission of additional packets, as this decreases the amount of packet overhead and routing decisions. For example, in the case of a single redundant segment, packet n contains, in addition to its encoded samples, a redundant version of packet $n-1$. This redundant information is usually obtained using a lower-bit-rate, lower-quality encoding than the primary information. This simple scheme only recovers from isolated losses but can be modified (as proposed in [3]) to recover from consecutive losses as well by carrying in packet n redundant versions of packets $n-1$ and $n-2$, or of packets $n-1$, $n-2$ and $n-3$ or of packets $n-1$ and $n-3$ etc.

Obviously, the more redundant information is added at the source, the more lost packets can be reconstructed. However, sending more redundant copies implies increasing the bandwidth requirement at the source (and henceforth the packet loss rate) and increasing the end-to-end delay (since the receiver has to wait longer for the redundant information). Moreover, it would make little sense to add much redundant information when the network load is low and the packet losses are rare.

Therefore, a robust FEC scheme should be adaptive and choose the FEC according to the network characteristics (such as packet loss process, available bandwidth,...) at any given time and depending upon its impact on the end-to-end delay. In the following sections we describe the packet loss process of audio packets in the Internet and we review the rate control schemes that have been proposed in the literature.

B. Loss process of audio packets

The efficiency of the FEC depends on the characteristics of the loss process of audio packets. Typically, FEC is more efficient when the consecutive number of lost packets is small.

There has been much research efforts in the measurement and modeling of end-to-end Internet characteristics [21],[22],[23]. The main result is that the correlation structure of the loss process of audio packets can be modeled with low order Markov chains. In particular, a two-state Gilbert model was found to be an accurate model in many studies. Moreover, it was found that the distribution of the number of packets lost in a loss period is approximately geometric [20],[23]. These results confirmed that FEC schemes are well suited for interactive audio applications in the Internet. In the rest of the paper, we will use the

Gilbert model to characterize the loss process of audio packets. The Gilbert model (depicted in Figure 1) is a two-state model in which state 1 represents a packet loss and state 0 represents a packet reaching the destination. The parameters p and q denote respectively the probabilities of passing from state 0 (no loss) to state 1 (loss) and from state 1 to state 0. In absence of redundant information, one can easily derive the packet loss rate $PLR = \frac{p}{p+q}$. This model also allows to compute the packet loss rates after reconstruction when FEC is used. For example, if we consider the case where packet n only includes redundant information about packet $n-1$, the packet loss rate after reconstruction is equal to $\frac{p(1-q)}{p+q}$.

C. Rate control

As we mentioned earlier, the addition of FEC repair data to a media stream is an effective means by which that stream may be protected against packet loss. However, the addition of large amounts of repair data when loss is detected will increase network congestion and hence packet loss, leading to a worsening of the problem which the use of FEC was intended to solve. Therefore, the rate of audio sources should be controlled in order to avoid congestion collapse in the network.

In addition, it has been suggested that audio applications share resources fairly with each other and with current TCP-based applications. One way to ensure this is to implement some form of congestion control that adapts the transmission rate in a way that *fairly* shares congested bandwidth with TCP applications. One definition of *fair* is that of TCP *friendliness* [24] - if a non-TCP connection shares a bottleneck link with TCP connections, traveling over the same network path, the non-TCP connection should receive the same share of bandwidth as a TCP connection.

There has been significant previous research on TCP-Friendly control mechanisms and many control schemes were proposed in the literature. We can distinguish three main classes of control mechanisms: (1) the window-based control mechanisms, (2) the mechanisms based on additive increase, multiplicative decrease (AIMD), and (3) the equation-based mechanisms.

The control mechanisms closest to TCP are the window-based mechanisms [25], [26]. They maintain a *congestion window* which is used to control the transmission of the packets. Although window-based schemes exhibit a behavior very close to the one of TCP, their main disadvantage is their lack of flexibility. Since these protocols strictly adhere to TCP window dynamics, it would be hard to modify them to take into account timeliness requirements of real-time streams.

Another class of control mechanisms uses additive increase, multiplicative decrease (AIMD) in some form, but do not apply AIMD to a congestion window. The RAP protocol (Rate Adaptation Protocol) [27] employs an AIMD rate control algorithm based on regular acknowledgments sent by the receiver. Another AIMD protocol has been proposed in [28]. This scheme relies on regular RTP/RTCP [29] reports sent between sender and receiver to estimate the loss rates and round-trip times. While AIMD schemes exhibit good response to transient changes in congestion, real-time streams find decreasing the sending rate in multiplicative order in response to a single loss to be unnecessarily severe (as it can noticeably decrease the user-perceived quality [30]). For this reason, equation-based control mechanism seem to be leading candidates for mechanisms to provide relatively smooth congestion control for real-time traffic.

In Equation-based congestion control [24] schemes, the sender uses an equation characterizing the allowed sending rate of a TCP connection as a function of the RTT and packet loss rate, and adjusts its sending rate according to those measured parameters. A key issue, when using these schemes, is of course to choose a reliable characterization of the TCP throughput. The TCP-Friendly protocols proposed in [30], [31] are based on the TCP response function first reported in [24] and later formalized in [32]. Since this characterization does not take the *timeouts* into account, it has been reported in [32] that this model is not accurate for loss rates higher than 5%. Another formulation of the TCP response function was derived in [33]. It states that the average throughput (in bytes/sec) of a TCP connection (r_{TCP}) is given by:

$$r_{TCP} = \frac{s}{t_{RTT}\sqrt{\frac{2l}{3}} + t_{RTO}(3\sqrt{\frac{3l}{8}})l(1+32l^2)} \quad (1)$$

where s is the packet size, t_{RTT} is the round trip time, t_{RTO} is the TCP re-transmission timeout and l is the frequency of loss indications per packet sent. Note that l is not exactly equal to the packet loss rate but the latter provides an upper bound on the value of l and can be used as an approximation. Based on this model, Padhye and al. [33] proposed a scheme in which the receiver acknowledges each packet. At fixed time intervals, the sender estimates the packet loss rate experienced during the previous interval and updates the sending rate using equation (1). Since this scheme updates the sending rate at fixed time intervals, it is suitable for use with multimedia applications. But it has the disadvantage to have a poor transient response at small time-scales. Besides, Floyd and al. recently proposed the TFRC protocol [34]. In TFRC, the sender explicitly adjusts its sending rate as a function of the measured rate of *loss events*, where a loss event consists of one or more packets dropped within a single RTT. Their algorithm for calculating the loss event rate is based on the method of Average Loss Interval. It offers a very good tradeoff between responsiveness to changes in congestion and avoidance of unnecessary abrupt shifts in the sending rate. Unfortunately, TFRC cannot be used, as such, with audio streams for two main reasons: first, because it requires the receiver to send feedback to the source at least once every RTT, which is not recommended when using RTCP for feedback collection; second, because the method of Average Loss Interval is not appropriate in the case where sources adjust their sending rate by changing the packet size while keeping the interval between packets constant. Our rate control scheme, which will be described in section V, was adapted from the work of Padhye ([33]) and adjusts the rate by keeping the interval between packets constant.

III. A UTILITY FUNCTION APPROACH WHICH ACCOUNTS FOR DELAY

As mentioned above, the purpose of our work is to design an error control algorithm which is *delay aware*, namely, which incorporates the impact on the end-to-end delay in the choice of the FEC. To this end, we take a utility function approach.

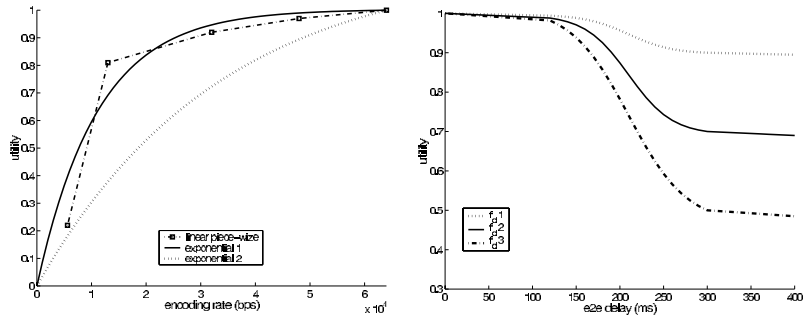


Fig. 2. *Utility as a function of (a) the rate only (for delay=0), (b) the end-to-end delay only (for encoding rate = 64Kbit/s)*

For users employing our audio application, we define the utility not only in terms of sound quality at the destination but also in terms of *interactivity*. In other words, we consider the utility (quality) as being a function of the encoding rate of audio received at the destination *and of the end-to-end delay*.

This utility function $f : \Re^+ \times \Re^+ \rightarrow [0,1]$ was obtained as follows. We characterized separately:

- **the utility as a function of the encoding rate**: $f_r : \Re^+ \rightarrow [0,1]$. There exists objective and subjective methods to assess the quality of an audio source as a function of the encoding rate. Objective methods use specific signal metrics to assess the quality, such signal-to-noise ration (SNR) [35]. These metrics, while easy to obtain, are only approximations for two main reasons. First, because they are sensitive to the characteristics of the signal, and hence to the words being pronounced. Second, because they often fail to correlate with perception properties of the human hearing system [36]. Despite these limitations, a SNR model can still give insight into the audio quality. Moreover, operational SNR curves (on a linear scale from 0 to 1) can be modeled using negative exponentials, the quality increasing rapidly at low rates, and more moderately at higher rates. Therefore, we considered two utility functions of the form: $f_r(x) = c_1 + c_2 e^{-\alpha x}$, where c_1 and c_2 are constants, selected so that $f_r(64Kbit/s) = 1$ and $f_r(0) = 0$. Figure 2(a) depicts two of these exponentials, corresponding respectively to $\alpha = 0.0001$ and $\alpha = 0.00003$. Quality assessment models based on subjective measurements provide more accurate results but are more difficult to obtain. A widely used model is MOS (Mean Opinion Scores) where the perceived quality is usually rated on a 1 to 5 scale. Utility functions for adaptive flows can be obtained using score tables (scaled down between 0 and 1), and interpolating between values to obtain piece-wize linear utility functions. We present such a function in Figure 2(a). Besides, exponentially-decay functions were introduced in [37] to describe utility curves for adaptive application. These curves account for the fact that the quality increases slowly at very low rates (below the minimum rate required by the application), then rapidly at intermediate rates and again slowly at higher rates. In this work, we restricted ourselves to consider strictly exponential curves, but more complex functions could be used for future work.

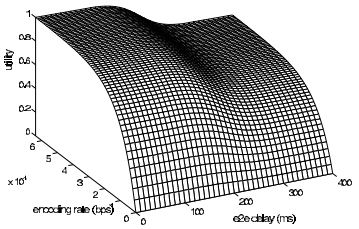


Fig. 3. *Utility as a function of the rate and the end-to-end delay*

- **the utility as a function of the end-to-end delay**: $f_d : \Re^+ \rightarrow [0,1]$. Even though an objective and unique function can not be set to describe the quality as a function of the end-to-end delay, various studies concluded that, for natural hearing, the end-to-end delay should be approximately 150ms [4], [38]. While a lower delay can not really be appreciated, delays above this threshold will be noticed by the users and will become a hindrance to interactivity. Moreover, it is also recognized that telephony users find delay of greater than about 300ms more like half duplex connection than a conversation. These considerations lead us to consider utility curves like those represented on figure 2(b). These utility functions present the following behaviour: for delays below the critical threshold of 150ms, the quality decreases very slowly as the users do not benefit from getting a lower end-to-end delay. Then, above this threshold, the quality drops steeply as any increase of the end-to-end delay hurts the interactivity. Then, above 300ms, since the connection is considered as a half duplex conversation anyway, any further increase of the delay only slightly affects the quality (which is already low). Even though it agrees with common intuition, the nature of the exact behaviour of this function remains out of the scope of this study. Our goal is not to validate a particular utility function but rather to show that the use of this kind of function allows to incorporate the impact on the end-to-end delay in the choice of FEC. To this end, we considered a set of utility func-

tions of the form:

$$f_d(x) = \begin{cases} 1-\gamma_1 x & \text{if } x \leq 150 \\ b_1 \tanh(\beta(x-b_2)) + b_3 & \text{if } 150 < x < 300 \\ 1-\delta-\gamma_2 x & \text{if } x \geq 300 \end{cases}$$

where x is the end-to-end delay (in ms), γ_1, β and γ_2 are parameters representing the steepness of the decrease in each of the 3 regions and δ determines the difference between the full and the half-duplex quality. b_1, b_2 and b_3 are constants selected to ensure the continuity of f_d. The utility functions depicted on figure 2(b) were obtained by tuning these parameters. A user will opt for either one or the other depending on the importance she attaches to the end-to-end delay.
Then, the global utility is defined as the product of f_d and f_r:

$$f(x,y) = f_r(x)\, f_d(y)$$

where x and y represent respectively the reconstructed rate at the destination and the end-to-end delay. Such a function is depicted on figure III.

Note that the packet loss rate after reconstruction also impacts the quality. Many codecs employ various concealment techniques which are able to mask such losses as long as $PLR_{after\,reconstr} \leq PLR_{max}$ where the threshold PLR_{max} depends on the codec. We could have incorporated this in our utility function approach. However, for tractability, we have chose to express this as a constraint, namely, we accept only encodings such that $PLR_{afterreconstr} \leq PLR_{max}$. Whether there is value in incorporating $PLR_{afterreconstr}$ in the utility function, and how to do it, is the object of future research.

In the simulation we have performed, we have used various values for the parameters of the utility functions.

IV. Our joint rate/errror/delay control

Once we have determined the influence of the end-to-end delay on the quality (utility) of the call, our goal is to **find the best redundant information that will maximize the perceived quality at the receiver.** This is the purpose of this section.

Consider a source with the flexibility to encode its samples at a rate $x \in [0, X_{max}]$ (we suppose X_{max} to be 64 Kbits/s). The quality of the voice call is characterized by a function $f : \Re^+ \times \Re^+ \rightarrow [0,1]$ of (1) the reconstructed rate at the destination and (2) the end-to-end delay.

The source transmits voice packets to a destination over an unreliable network characterized by:

- **a packet loss process:** Y_i which we suppose to be a Gilbert process where $Y_i \in \{0,1\}$ (see section II-B). If the ith packet is received at the destination, then $Y_i = 0$, otherwise, $Y_i = 1$. The parameters p and q of the Gilbert model are estimated on-line at the receiver using the maximum likelihood estimator.
- **a delay distribution.** We don't make any assumption about the distribution of delays in the network. Rather do we consider the 99 percentile of the delays experienced by the voice packets, which we call d_{net}. d_{net} actually represents the playout delay of the voice packets and is estimated at the destination as described in [39].

The parameters p, q and d_{net} (which are all estimated on-line at the receiver) are sent back to the source via the application specific part (APP) of the RTCP receiver reports.

Let R_{max} be the rate available for the audio flow. R_{max} is the result of our TCP-Friendly rate control scheme (which is described in the next section) and is updated upon reception of an RTCP receiver report.

Then, consider that we use the *media specific* FEC scheme described above to recover from packet losses. Let $K-1$ denote the maximum number of redundant pieces of information sent along with the primary information. Thus, packet n carries information about at most (i.e. a subset of) packets $n-1,\ldots,n-K+1$. Therefore the total number of copies (encoded at different rates, including 0) of a given packet sent by the source is equal to K. The optimal value of K is a priori unknown and is supposed to be in $[1, K_{max}]$. In practice, the larger K, the longer the destination has to wait to receive the redundant information, and thus, the longer the end-to-end delay. Let τ_K represent the delay introduced by the use of FEC. τ_K is the delay between sending the first and the last copy of a given packet and can thus be written as follows: $\tau_K = (K-1)\,pcktInt$, where $pcktInt$ is the time interval between two consecutive audio packets.

Further, define the random variable Δ to be $\Delta = \{i | Y_i = 0, i = 1,\ldots,K\}$, namely the set of copies of a given packet that are received at the destination.

Then, our problem can be stated as follows: Given that we can send at most K_{max} copies of each voice packets, find the optimal number of copies to send, and the optimal encoding rate for each copy, so as to maximize the quality of the voice call subject to the rate constraint. Mathematically, it gives the following optimization problem (which we call P1):

$$\underset{\substack{1\leq K\leq K_{max} \\ x_i\,(i=1,\ldots,K)}}{\text{maximize}} \quad \sum_{\Delta\subseteq\{1,\ldots,K\}} P(\Delta) \max_{i\in\Delta} f(x_i, d_{net}+\tau_K)$$

$$\text{subject to} \quad \begin{cases} \sum_{i=1}^{K} x_i + R_{overhead} \leq R_{max} \\ x_i \geq 0, i = 1,\ldots,K \\ PLR_{after\,reconstr} \leq PLR_{max} \end{cases}$$

where x_i is the encoding rate of the copy placed in ith position in the stream ($i = 1$ corresponds to the primary information); $P(\Delta)$ is the probability to receive the set Δ; $R_{overhead}$ is the bandwidth overhead of the IP/UDP/RTP headers, $PLR_{after\,reconstr}$ is the packet loss rate after reconstruction and PLR_{max} is the maximum acceptable value for $PLR_{after\,reconstr}$. The choice of a value for PLR_{max} mainly depends on the efficiency of the error resilience scheme used at the receiver. Typical values of PLR_{max} range between 5 and 10% [40].

The objective function above represents the average quality measured at the destination. This model assumes that different copies of a given packet can not be combined to produce a better quality copy of the original packet. We rather assume that the receiver will send to its audio driver the *best* copy (i.e. leading to the highest quality) it has received of a given packet. The formulation of the objective function could be different if we used layered or multiple description coding [41] schemes for audio. But this is kept for future work.

The problem above appears to be, in general, difficult to solve but a careful analysis of the objective function allowed us to derive solutions for K_{max} up to 5. The methodology remains the same for greater values of K_{max} but the main burden is that the number of terms in the sum grows exponentially with K, and there is no generic formulation to express the probabilities associated to each term as a function of K. For further details about the resolution of the optimization problem, one can refer to appendix A. In the following, we just describe the general characteristics of the results:

- $x_1 \geq x_i, \forall i = 1, ..., K$: the primary information should always be encoded using the best quality encoding among those used to encode the different copies.
- **if** $p + q < 1$, $x_1 \geq x_K \geq x_i, \forall i = 1, \ldots, K$: it pays to use good quality coders to encode the end packets. Furthermore, for a given number of copies to send, the larger K, the better the audio quality at the destination, but also the larger the end-to-end delay. In this case, our algorithm allows to find the good tradeoff between quality of the reconstructed signal and delay.
- **if** $p + q > 1$, $x_1 \geq x_2 \geq \ldots \geq x_K, \forall i = 1, \ldots, K$: the redundant copies should closely follow the primary packet and the quality of the encodings should decrease as the copies are moved away from the primary packet.

V. The TCP-Friendly rate control module

Our TCP-Friendly rate control relies on the Real-Time Transport Protocol (RTP) [29] and its control part, RTCP. In our scheme, the sender performs equation-based congestion control based on feedback information contained in RTCP reports. In details, it works as follows: about every 5s (as advised in [29]), the receiver issues an RTCP report containing the packet loss rate experienced since sending the last report and various timestamps. After receiving the nth RTCP receiver report, the sender estimates the bandwidth share it should be using as follows:

- It computes the round-trip-time, t_{RTT_n} using the timestamps contained in this report and updates the TCP timeout value (t_{RTO_n}) accordingly.
- To avoid reactions to sudden loss peaks in the network, it maintains an Exponentially Weighted Moving Average of the packet loss rate:

$$PLR_n = (1 - \alpha)\, PLR_{n-1} + \alpha PLR \qquad (2)$$

where PLR is the packet loss rate reported in the RTCP packet and α is the smoothing factor set here to 0.3. This value was also suggested in [42].

- Then, it estimates the TCP-Friendly rate r_{TCP_n} (in bytes/sec) using equation (1) with the parameters t_{RTT_n}, t_{RTO_n} and PLR_n. The value of s is set to the packet size of a competing TCP connection. Typically, s can be set to 576 Bytes (MSS) or 1500 Bytes (MTU).
- And finally, it updates the sending rate (r_n) as follows:

```
if    (r_TCP_n < r_n-1)
  decrease sending rate to r_TCP_n
else
  r_AI = r_n-1 + sT / t_RTT_n^2
  increase sending rate to min[r_TCP_n, r_AI]
```

where $T \sim 5s$ is the time elapsed since receiving the last RTCP report and r_{AI} is the rate that a TCP connection would have reached by increasing its congestion window by one packet every round trip time. This last condition was introduced to make sure that our audio flow does not increase its bandwidth share faster than a TCP connection sharing the same link.

The audio source then adjusts its sending rate by **changing the packet size**, the time interval between packets remaining constant. Usually, audio packets are generated every 20, 40 or 80ms. In this work, we fixed this interval to 20ms.

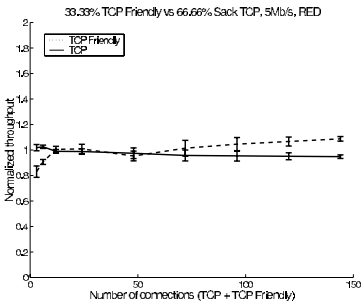


Fig. 4. *Our rate control protocol competing with TCP.*

We have tested our rate control protocol in the *ns* [11] simulator. Figure 4 illustrates the fairness of our protocol when competing with TCP Sack traffic in a RED queue. In these simulations, n TCP Friendly and $2n$ TCP flows share a common bottleneck. Graphs show the mean throughput of each type of flow, averaged over the last 300s of simulation and over all the connections, and normalized so that a value of 1 corresponds to a fair share of the bandwidth. Each point on this graph represents the results averaged over 5 simulation runs and the corresponding confidence intervals. This figure shows that our rate control protocol co-exists fairly with TCP under a wide range of network conditions.

VI. Simulation examples

We implemented and tested our *delay aware* error control scheme in the ns2 simulator [11]. This section presents a summary of our results. Two types of IP networks were considered: (1) Single Class Best-Effort networks (that we call 'Flat' networks) and (2) Alternative Best-Effort networks.

A. Flat Best-Effort Networks

This section investigates the behaviour of our scheme under a wide range of loss conditions. We consider a simple scenario where n audio (TCP-Friendly) flows and $3n$ Sack TCP flows share a single bottleneck link. The packet loss rate experienced by the connections is varied artificially by changing the number of connections sharing the link. Figures 5(a) and (b) represent respectively the packet loss rate experienced by audio flows and the corresponding TCP-Friendly rate constraint as a function of the number of connections sharing the link. The bottleneck bandwidth is 5Mbits/s and the one-way propagation delay (without queuing) is the same for all connections and is fixed at 100ms. The graphs show the mean values averaged over the last 300s of simulation and over all connections. Each point on

this graph represents the results averaged over 5 simulation runs and the corresponding 99% confidence intervals. For our experiments, the parameter PLR_{max} was set to 5%.

As explained in Section III, various values can be used for the parameters of the utility function $f = f_r f_d$. Figure 5(c) shows the mean utility obtained by the sources for three different parameter settings of f_r, for a fixed $f_d = f_{d2}$ (see Figure 2(b)). The results obtained with the piecewize linear f_r (MOS) and with the exponential of parameter $\alpha = 0.0001$ (exponential 1 on the figure) were extremely close to each other and the results obtained with the exponential of parameter $\alpha = 0.00003$ differed slightly in the way they spread the rate among the different copies. In all cases, the end-to-end delay (including FEC) was the same. In the rest of the paper, all the results shown were obtained using the exponential of parameter $\alpha = 0.0001$ for f_r. We now consider three different utility functions, which are defined as follows: Utility i $= f_r f_{d_i}$, for $i \in \{1, 2, 3\}$ where the functions f_{d_i} are the one depicted in Figure 2(b). Utility 1 corresponds to a source that does not attach much importance to the delay. Utility 3 characterizes a source for which delay is an important issue and Utility 2 represents a tradeoff between delay and audio distortion issues.

Figures 5 (d), (e) and (f) compare the mean utility obtained when using the delay aware FEC to the one obtained when using the FEC scheme proposed in [31] (which we will refer to as *classical FEC*) for the three utility functions of interest. For clarity, we do not show confidence intervals but we just mention that they were small ($< 2\%$). On these figures, each curve corresponds to a particular parameter setting (i.e. value of K) of the classical FEC scheme. As one can see, there is no optimal setting of the classical FEC which maximizes the utility in all loss conditions. The **delay aware scheme** (bold curve on the figures), in return, **always chooses the optimal amount of FEC (i.e. yielding the maximal utility) depending on network conditions.** Figures 5 (g), (h) and (i) show the corresponding end-to-end delays (including the FEC-induced delay) obtained with each scheme. One can observe that, in the delay aware scheme, the end-to-end delay is adapted, depending on loss conditions. The delay is kept small when the losses are moderate and is increased when the losses become more significant. Moreover, when comparing figures 5 (g), (h) and (j), one can see that, depending on the importance the user attaches to the end-to-end delay, the amount of redundancy used is increased more or less rapidly as the loss rate increases. The same simulations were performed with smaller values of the propagation delay and showed that, in the case where this delay is very small (i.e. 50 to 80ms), the delay aware scheme leads to performances similar to the classical FEC with parameter $K = 3$ or $K = 4$, which could have been expected since, in this case, the delay induced by the FEC has no consequence on the perceived quality (because we stay below the critical threshold of 150ms).

In the light of these results, we can conclude that the delay aware scheme increases the utility by avoiding a source wasting delay using FEC when it is not necessary (and when it could hurt the perceived quality).

B. Alternative Best-Effort Networks

As stated above, a secondary motivation to develop a delay aware scheme is the existence of a number of proposals for internet differentiated services which propose a tradeoff between delay and throughput [8], [7]. In the case of TCP-Friendly audio sources whose sending rate is controlled via packet losses, this tradeoff is not simple. Indeed, in today's Internet, a source that chooses a low delay class will be likely to experience more losses and hence may be forced to compensate this additional loss by more FEC, and thus more end-to-end delay. This raises the following question: **does an audio source benefit from trading delay for throughput?**

In this paper, we chose to look into the proposal for ABE service [9], which is closer to today's Internet context. We leave the analysis of [8] for further study. With ABE, every best-effort packet is marked either *green* or *blue*. Green packets are guaranteed a low bounded delay at every router, but, in return, are more likely to be dropped during periods of congestion than blue ones. As a consequence, they will typically receive less throughput than blue ones. In this framework, the question above can be reformulated as follows: is it worth being green? In the sequel, we make use of our delay aware scheme to provide an answer to this question.

We considered a topology consisting of 2 consecutive bottleneck links. n Long flows traverse both bottlenecks, n small flows traverse only the first link and n small flows traverse only the second link. The adaptive audio applications represent one third of the connections of each type, the other two thirds are Sack TCP connections. Among the adaptive applications, half are green (i.e. use only green packets) and half are blue. Sack TCP connections use blue packets. Other flow repartitions, which are not covered here, were simulated and lead us to similar conclusions to the one presented here.

Let us first consider the case of small flows. Figures 6 (a) and (b) show the packet loss rates and the corresponding TCP-Friendly rate constraint for small green and blue flows. The bottleneck bandwidth is 5Mbits/s and the propagation delay is 50ms. Figure 6 (c) depicts the 99% percentile of the end-to-end delay (non including the FEC) experienced by green and blue packets. The tradeoff of delay and loss (or equivalently throughput) appears clearly on these three figures. Figures 6 (d), (e) and (f) present the utility received by the flows for the three utility functions of interest: Utility 1, 2 and 3 (see description above) respectively. From these graphs, one can see that, when the delays are small, the difference between the utilities of blue and green flows is minor. It is even more visible in the case where Utility 2 (which represents a tradeoff between delay and distortion of audio at destination) is used.

Secondly, we consider large flows. The propagation delay is now 90ms. Figures 6 (g), (h) and (i) show respectively the packet loss rate, TCP-Friendly rate constraint and the 99% percentile of the delays in the network for green and blue flows. The utility values obtained in these cases are depicted in Figures 6 (j), (k) and (l), for the three utility functions of interest. In this case, one can notice that the difference between the utilities received by green and blue flows is much higher. Furthermore, figures 6 (k) and (l) clearly show that, a **user who attaches at least a some small importance to the end-to-end delay will**

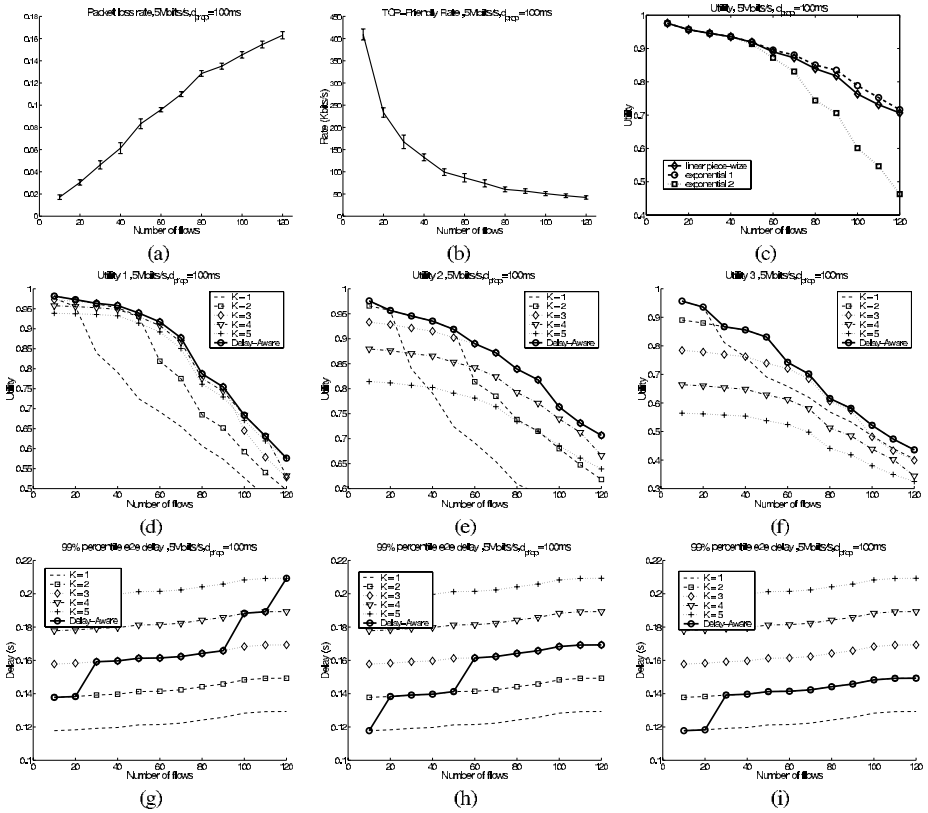


Fig. 5. *Packet loss rate (a). TCP-Friendly rate (b). Utility for different f_r (c). Utility of delay aware FEC vs classical FEC for (d) $f_d = f_{d1}$, (e) $f_d = f_{d2}$, (f) $f_d = f_{d3}$. e2e delay (including FEC) of delay aware FEC vs classical FEC for (g) $f_d = f_{d1}$, (h) $f_d = f_{d2}$, (i) $f_d = f_{d3}$.*

benefit from being green, up to a certain level of network congestion where the available rate becomes an impediment and the source could choose to switch to being blue in order to increase its throughput. It is up to the source to determine when it is better to switch from one color to another. This result tends to indicate that there is a need for color choosing algorithm for audio sources using ABE. On the other hand, a user who does not care about delay (see figure 6(j)), will probably choose the blue service in all cases.

From these results, we conclude that (time sensitive) audio applications benefit, when using ABE, from being green except when the network is very badly congested or in trivial cases when the delay is extremely small anyway.

VII. Conclusions

In this paper, we presented an adaptive error control scheme for audio which is *delay aware*, namely, which incorporates the impact on the end-to-end delay in the choice of the FEC. To this end, we took a utility function approach and we defined the quality as being a function of the encoding rate received at destination *and* of the end-to-end delay. We formalized our control problem as an optimization problem and solved it numerically and theoretically.

We showed by simulation that the delay aware scheme does avoid that a source waste delay using FEC when it is not necessary. Moreover, using our scheme in the framework of Alternative Best Effort networks, we showed that there is a real benefit, for audio applications, to use the low delay class of the service.

We are pursuing this work in several directions. One is to implement the delay aware scheme in a real audio software (such as the Robust Audio Tool [43]). Another is to develop Color Choosing algorithms for audio sources using the ABE service. Yet another one is to use the same utility function approach with other audio coding schemes such as multiple description coding [41].

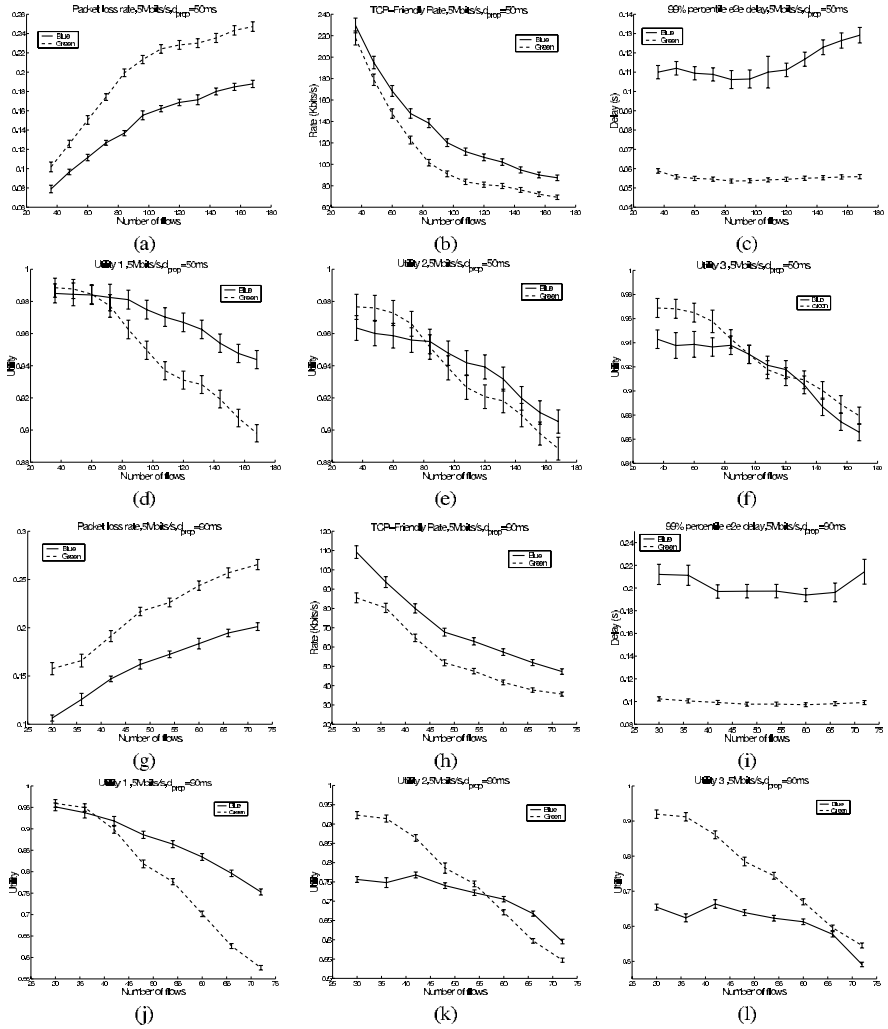


Fig. 6. *Small flows: Packet loss rate (a). TCP-Friendly rate (b). 99% percentile of e2e delay (not including FEC) (c). Utility for (d) $f_d = f_{d1}$, (e) $f_d = f_{d2}$, (f) $f_d = f_{d3}$. Long flows: Packet loss rate (g). TCP-Friendly rate (h). 99% percentile of e2e delay (without FEC) (i). Utility for (j) $f_d = f_{d1}$, (k) $f_d = f_{d2}$, (l) $f_d = f_{d3}$.*

APPENDIX

I. THE OPTIMIZATION PROBLEM

In this section, we describe the method used to solve the problem P1, which we can rewrite as follows:

$$\underset{1 \le K \le K_{max}}{\text{maximize}} \quad F_K(K, \underline{x}^K)$$
$$\underline{x}^K = (x_1, .., x_K) \in [0, X_{max}]^K$$

$$\text{subject to} \quad \begin{cases} \sum_{i=1}^{K} x_i + R_{overhead} \le R_{max} \\ PLR_{after\ reconstr} \le PLR_{max} \end{cases}$$

with $F_K(K, \underline{x}^K) = \sum_{\Delta \subseteq \{1,..,K\}} P(\Delta) \max_{i \in \Delta} f(x_i, d_{net} + \tau_K)$. For clarity, we use the notations $F_K(K, \underline{x}^K) = F_K$ and $f(x_i, d_{net} + \tau_K) = f_{i,K}$ when the context permits. In the following, we give

the details for $K_{max} = 5$. We have:

$$F_1 = \frac{q}{p+q} f_{1,1}$$

$$F_2 = \frac{q}{p+q}(1-p) \max_{i\in\{1,2\}} f_{i,2} + \frac{pq}{p+q} f_{1,2} + \frac{pq}{p+q} f_{2,2}$$

$$F_3 = \frac{q}{p+q}(1-p)^2 \max_{i\in\{1,2,3\}} f_{i,3} + \frac{pq}{p+q}(1-p) \max_{i\in\{1,2\}} f_{i,3} + \frac{pq^2}{p+q} \max_{i\in\{1,3\}} f_{i,3} + \frac{pq}{p+q}(1-p) \max_{i\in\{2,3\}} f_{i,3} + \frac{pq}{p+q}(1-q) f_{1,3} + \frac{p^2 q}{p+q} f_{2,3} + \frac{pq}{p+q}(1-q) f_{3,3}$$

$$F_4 = \frac{1}{p+q} * \{q(1-p)^3 \max_{i\in\{1,2,3,4\}} f_{i,4} + pq(1-p)^2 \max_{i\in\{1,2,3\}} f_{i,4} + pq^2(1-p) \max_{i\in\{1,2,4\}} f_{i,4} + pq^2(1-p) \max_{i\in\{1,3,4\}} f_{i,4} + pq(1-p)^2 \max_{i\in\{2,3,4\}} f_{i,4} + pq(1-p)(1-q) \max_{i\in\{1,2\}} f_{i,4} + p^2 q^2 \max_{i\in\{1,3\}} f_{i,4} + pq^2(1-q) \max_{i\in\{1,4\}} f_{i,4} + p^2 q(1-p) \max_{i\in\{2,3\}} f_{i,4} + p^2 q^2 \max_{i\in\{2,4\}} f_{i,4} + pq(1-p)(1-q) \max_{i\in\{3,4\}} f_{i,4} + pq(1-q)^2 f_{1,4} + p^2 q(1-q) f_{2,4} + p^2 q(1-q) f_{3,4} + pq(1-q)^2 f_{4,4}\}$$

$$F_5 = \frac{1}{p+q} * \{q(1-p)^4 \max_{i\in\{1,2,3,4,5\}} f_{i,5} + pq(1-p)^3 \max_{i\in\{1,2,3,4\}} f_{i,5} + pq^2(1-p)^2 \max_{i\in\{1,2,3,5\}} f_{i,5} + pq^2(1-p)^2 \max_{i\in\{1,2,4,5\}} f_{i,5} + pq^2(1-p)^2 \max_{i\in\{1,3,4,5\}} f_{i,5} + pq(1-p)^3 \max_{i\in\{2,3,4,5\}} f_{i,5} + pq(1-p)^2(1-q) \max_{i\in\{1,2,3\}} f_{i,5} + p^2 q^2(1-p) \max_{i\in\{1,2,4\}} f_{i,5} + pq^2(1-p)(1-q) \max_{i\in\{1,2,5\}} f_{i,5} + p^2 q^2(1-p) \max_{i\in\{1,3,4\}} f_{i,5} + p^2 q^3 \max_{i\in\{1,3,5\}} f_{i,5} + pq^2(1-p)(1-q) \max_{i\in\{1,4,5\}} f_{i,5} + p^2 q(1-p)^2 \max_{i\in\{2,3,4\}} f_{i,5} + p^2 q^2(1-p) \max_{i\in\{2,4,5\}} f_{i,5} + p^2 q^2(1-p) \max_{i\in\{2,3,5\}} f_{i,5} + pq(1-p)^2(1-q) \max_{i\in\{3,4,5\}} f_{i,5} + pq(1-p)(1-q)^2 \max_{i\in\{1,2\}} f_{i,5} + p^2 q^2(1-q) \max_{i\in\{1,3\}} f_{i,5} + p^2 q^2(1-q) \max_{i\in\{1,4\}} f_{i,5} + pq^2(1-q)^2 \max_{i\in\{1,5\}} f_{i,5} + p^2 q(1-p)(1-q) \max_{i\in\{2,3\}} f_{i,5} + p^3 q^2 \max_{i\in\{2,4\}} f_{i,5} + p^2 q^2(1-q) \max_{i\in\{2,5\}} f_{i,5} + p^2 q(1-p)(1-q) \max_{i\in\{3,4\}} f_{i,5} + p^2 q^2(1-q) \max_{i\in\{3,5\}} f_{i,5} + pq(1-p)(1-q)^2 \max_{i\in\{4,5\}} f_{i,5} + pq(1-q)^3 f_{1,5} + p^2 q(1-q)^2 f_{2,5} + p^2 q(1-q)^2 f_{3,5} + p^2 q(1-q)^2 f_{4,5} + pq(1-q)^3 f_{5,5}\}$$

The original maximization problem can therefore be divided into the sub-problems of finding the constrained maxima of $F_K(K, \underline{x}^K)$, $K = 1, \ldots, K_{max}$, where $\underline{x}^K$ is the variable and K is fixed.

Still, the maximization of F_K is made difficult by the presence of the *max* functions. To get around this, we partitioned $\Re^K$ into 2^{K-1} sub-spaces σ_i characterized by $\{x_j < x_k < \ldots < x_l\}$ where $\{j, k, \ldots, l\}$ are all the permutations of the set $\{1, \ldots, K\}$. Considering F_K over each of these sub-spaces allowed to remove the *max* functions. Moreover, we could identify the subspaces in which the maxima of F_K occur. These sub-spaces depend on the values of p and q. And we could finally rewrite the optimization problem P1 into the problem P2:

$$\begin{array}{l} \text{maximize} \\ 1 \le K \le K_{max} \end{array} \quad \max_{\underline{x}^K \in [0, X_{max}]^K} F_K(K, \underline{x}^K)$$

subject to the same constraints as P1. Where the functions F_K are defined as follows:

$$F_1 = \frac{q}{p+q} f_{1,1}$$

$$F_2 = \frac{q}{p+q} f_{1,2} + \frac{pq}{p+q} f_{2,2}$$

K	Redundancy	$PLR_{after\ reconstr}$
1	none	$p/(p+q)$
2	-1	$p(1-q)/(p+q)$
3	-2	$(p^2q + p(1-q)^2)/(p+q)$
	-1-2	$(p(1-q)^2)/(p+q)$
4	-3	$(p(3pq - p^2q - 2q^2p + (1-q)^3))/(p+q)$
	-1-3	$(p(1-q)(pq + (1-q)^2))/(p+q)$
	-1-2-3	$p(1-q)^3/(p+q)$
5	-4	$(p^2q((1-p)^2 + 2(1-p)(1-q) + 3(1-q)^2) + p^3q^2 + p(1-q)^4)/(p+q)$
	-2-4	$(p^3q^2 + 2p^2q(1-q)^2 + p(1-q)^4)/(p+q)$
	-1-2-4	$(p^2q(1-q)^2 + p(1-q)^4)/(p+q)$
	-1-2-3-4	$(p(1-q)^4)/(p+q)$

TABLE I

Loss rates after reconstruction. Note: in the column Redundancy, -1-2 means that packets n-1 and n-2 were sent in packet n.

if $p + q < 1$

$$F_3 = \frac{1}{p+q} * \{q f_{1,3} + p^2 q f_{2,3} + pq(2-p-q) f_{3,3}\}$$

$$F_4 = \frac{1}{p+q} * \{q f_{1,4} + p^2 q(2-p-q) f_{2,4} + p^2 q(1-q) f_{3,4} + pq(3-3p+p^2+2pq-3q+q^2) f_{4,4}\}$$

$$F_5 = \frac{1}{p+q} * \{q f_{1,5} + p^2 q(pq + (1-q)^2) f_{2,5} + p^2 q(2(1-p)(1-q) + (1-q)^2 + (1-p)^2) f_{3,5} + p^2 q(1-q)^2 f_{4,5} + pq((1-p)(1-q)^2 + (1-p)^2(1-q) + (1-q)^3 + 2pq(2-p-q) + (1-p)^3) f_{5,5}\}$$

if $p + q > 1$

$$F_3 = \frac{1}{p+q} * \{q f_{1,3} + pq f_{2,3} + pq(1-q) f_{3,3}\}$$

$$F_4 = \frac{1}{p+q} * \{q f_{1,4} + pq f_{2,4} + pq(1-q) f_{3,4} + pq(1-q)^2 f_{4,4}\}$$

$$F_5 = \frac{1}{p+q} * \{q f_{1,5} + pq f_{2,5} + pq(1-q) f_{3,5} + pq(1-q)^2 f_{4,5} + pq(1-q)^3 f_{5,5}\}$$

The constraint on the packet loss rate after reconstruction can be formulated as a set of constraints on the values of $\underline{x}^K = (x_1, \ldots, x_K)$. Actually, Table 1 shows $PLR_{after\ reconstr}$ for a given amount of redundancy. Hence, a maximum value for $PLR_{after\ reconstr}$ amounts to imposing a minimum amount of redundancy. If r_0 denotes the minimum rate used to encode audio samples, $PLR_{after\ reconstr} < PLR_{max}$ is equivalent to $x_i \ge r_0$, for all i in the minimal set of copies which yield a packet loss rate smaller than PLR_{max}.

Once the formulation of the original problem is simplified, the maximization of the objective functions $F_K, K = 1, \ldots, K_{max}$ can be carried out using classical methods, the choice of method dependant on the utility function $f(x, y)$. If $f(x, y)$ is differentiable with respect to x and strictly concave, the maximizing values $\underline{x}^K$ can be found by the Lagrangian method. If $f(x, y)$ is a non-linear concave function of x, numerical methods such as SQP (Sequential Quadratic Programming) can be used. In addition, if $f(x, y)$ is a piecewise linear function of x, linear programming methods provide a solution to the maximization problem.

REFERENCES

[1] J-C. Bolot, S. Fosse Parisis, and D. Towsley, "Adaptive FEC-based error control for internet telephony," in *Infocom'99*, March 1999.

[2] M. Podolsky, C. Romer, and S. McCanne, "Simulation of fec-based error control for packet audio in the internet," in *IEEE Infocom'98*, April 1998.

[3] J-C. Bolot and A. Vega Garcia, "Control mechanisms for packet audio in the internet," in *IEEE Infocom'96*, March 1996.

[4] T. J. Kostas, M. S. Borella, I. Sidhu, G. M. Schuster, J. Grabiec, and J. Mahler, "Real-time voice over packet-switched networks," *IEEE Network*, pp. 18–27, January/February 1998.

[5] C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional differentiated services: Delay differentiation and packet scheduling," in *Proc. SIGCOMM'99*, September 1999, pp. 109–122.

[6] Y. Moret and S. Fdida, "A proportional queue control mechanism to provide differentiated services," in *International Symposium on Computer System*, Belek, Turkey, 1998 October.

[7] P. Hurley, J.-Y. Le Boudec, and P. Thiran, "The asymmetric best-effort service," in *IEEE Globecom 1999*, Rio de Janeiro, Brazil, Dec. 1999.

[8] S. Keshav, *Congestion Control in Computer Networks*, Ph.D. thesis, UC Berkeley TR-654, September 1991.

[9] P. Hurley, M. Kara, J.-Y. Le Boudec, and P. Thiran, "Abe: Providing a low-delay service within best-effort," Tech. Rep. DSC/2001/011, EPFL-DI-ICA, February 2001.

[10] S Floyd, "TCP and explicit congestion notification," *ACM Computer Communications Review*, vol. 21 n0 5, pp. 8–23, 1994.

[11] "Network simulator," Available from http:www-mash.cs.berkeley.edu.ns.

[12] C. S. Perkins, O. Hodson, and V. Hardman, "A survey of packet-loss recovery techniques for streaming audio," September/October 1998.

[13] R. Blahut, *Theory and Practice of Error control codes*, Addison-Wesley, 1993.

[14] N. Shacham and P. Mc Kenney, "Packet recovery in high-speed networks using coding and buffer management," in *Proc. IEEE Infocom'1990*, June 1990, vol. 1, pp. 124–131.

[15] D. R. Figueredo and E. de Souza e Silva, "Efficient mechanisms for recovering voice packets in the internet," in *Proc. IEEE Globecom'99*, 1999, pp. 1830–1836.

[16] V. Hardman, A. Sasse, M. Handley, and A. Watson, "Reliable audio for use over the internet," in *Proceedings of INET'95*, June 1995.

[17] H. Sanneck, "Concealment of lost speech packets using adaptive packetization," in *Proceedings IEEE Multimedia Systems 1998*, Austin, TX, June 1998.

[18] International Telecommunications Union, "Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear-prediction (cs-acelp)," Tech. Rep., ITU-T Recommendation G.729, March 1996.

[19] H. Sanneck and N. Le, "Speech property-based FEC for Internet Telephony applications," in *Proceedings of the SPIE/ACM SIGMM Multimedia Computing and Networking Conference (MMCN)*, San Jose, CA, January 2000, pp. 38–51, ftp://ftp.fokus.gmd.de/pub/glone/papers/Sann0001:Speech-FEC.ps.gz.

[20] J-C. Bolot and A. Vega Garcia, "The case for FEC-based error control for packet audio in the internet," in *to appear in ACM Multimedia Systems*.

[21] V. Paxson, "End-to-end internet packet dynamics," in *Proc. ACM Sigcomm'97*, september 1997, Cannes, France.

[22] J.-C. Bolot, "End-to-end packet delay and loss behavior in the internet," in *Proc. ACM Sigcomm'93*, August 1993, pp. 189–199, San Francisco, CA.

[23] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modelling of temporal dependence in packet loss," in *Proc. IEEE Infocom'99*, March 1999, New York, NY.

[24] J. Mahdavi and S. Floyd, "TCP-Friendly unicast rate-based flow control," in *Draft posted on end2end mailing list*, January 1997, http://www.psc.edu/networking/papers/tcp_friendly.html.

[25] S. Jacobs and A. Eleftheriadis, "Providing video services over networks without quality of service guarantees," in *RTMW'96*, October 1996, Sophia Antipolis, France.

[26] I. Rhee, V. Ozdemir, and Y. Yi, "TEAR: TCP emulation at receivers - flow control for multimedia streaming," Technical report, Department of Computer Science, NCSU, April 2000.

[27] R. Rejaie, M. Handley, and D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet," in *Proc. IEEE Infocom'99*, March 1999, New York.

[28] D. Sisalem and H. Schulzrinne, "The loss-delay adjustment algorithm: A TCP-Friendly adaptation scheme," in *NOSSDAV'98*, July 1998, Cambridge,UK.

[29] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," RFC 1889, 1996.

[30] W. Tan and A. Zakhor, "Error resilient packet video for the internet," in *ICIP'98*, October 1998, vol. 3, pp. 458–462.

[31] J-C. Bolot and T. Turletti, "Experience with rate control mechanisms for packet video in the internet," *ACM Computer Communication Review*, vol. 28, no. 21, January 1998.

[32] M. Mathis, J. Semke, J. Madhavi, and T. Ott, "Macroscopic behavior of TCP congestion avoidance algorithm," *Computer Communication Review*, vol. 27, no. 3, July 1997.

[33] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in *Proc. SIGCOMM'98*, 1998.

[34] Sally Floyd, Mark Handley, Jitendra Padhye, and Joerg Widmer, "Equation-based congestion control for unicast applications," in *SIGCOMM'2000*, August 2000.

[35] T. Cover and J. Thomas, *Elements of Information theory*, John Wiley and Sons, 1991.

[36] J. Tribolet, P. Noll, B. McDermott, and R. Crochiere, "A study of complexity and quality of speech waveform coders," April 1978, pp. 586–590, Tulsa.

[37] L. Breslau and S. Shenker, "Best-effort versus reservations: A simple comparative analysis," *Proc. of ACM SIGCOMM'98*, August 1998, Vancouver, Canada.

[38] R. V. Cox and P. Kroon, "Low bit-rate speech coders for multimedia communication," *IEEE Communications Magazine*, pp. 34–41, December 1996.

[39] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks," in *Proc. IEEE Infocom'94*, 1994.

[40] N. Jayant, "Effects of packet loss on waveform coded speech," in *Proc. 5th Int. Conference on Computer Communications*, October 1980, pp. 275–280.

[41] V. A. Vaishampayan, "Design of multiple description scalar quantizers," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 821–834, May 1993.

[42] D. Sisalem, F. Emmanuel, and H. Schulzrinne, "The direct adjustment algorithm: A TCP-Friendly adaptation scheme," Technical report, GMD-FOKUS, August 1997.

[43] "Robust audio tool," http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/.

Performance Analysis of Measurement-Based Call Admission Control on Voice Gateways

Feng Cao Hanlin Fang Mary Conlon
Packet Telephony Division, Cisco Systems, INC.
170 West Tasman Drive, San Jose, CA 95134, U.S.A.
Email: {fcao, hfang, meconlon} @cisco.com

Keywords: Call Admission Control, Quality of Service, Voice over IP

Abstract

Quality of Service (QoS) is critical for the success of real time applications over IP, such as Voice over IP (VoIP). On voice gateways, Call Admission Control (CAC) plays an important role for guaranteed QoS, for it makes decisions on whether and how to deliver the traffic based on different kinds of resources. In our study, we propose the measurement-based CAC model based on the various system resources on the local voice gateways. The end-to-end network congestion is also considered along with the configurable busy out on the voice gateways. The performance data is provided to show the improvement on a set of parameters for better Quality of Service and better serviceability of voice gateways.

1 Introduction

The traditional circuit switch infrastructure for telephony services will be augmented by packet switch infrastructure in the near future. Transport of voice and data across Internet has been integrated by both enterprises and service providers. There are many benefits for choosing Voice over IP. For example, the low cost of IP can save the customers more than the expensive circuit switches. More service can be easily created and delivered in IP than the current telephony infrastructure.

Voice gateways play an important role for carrying voice over IP. The voice comes into the ingress voice gateways through T1, E1, or POTS and is streamed into Voice over IP (VoIP) packets that is routed to the egress voice gateways.

VoIP applications are different from data services. As they are real-time and interactive, there are strict requirements on the delay and the jitter for end-to-end delivery. Therefore, Quality of Service (QoS) is critical for providing the expected behaviors of VoIP applications. In most of cases, it is impossible to imagine that voice calls or fax calls can be delivered without reasonable delay and loss for customers.

In this paper, we study the QoS issues on voice gateways through Call Admission Control (CAC). To guarantee QoS, CAC must be provided on voice gateways, allowing the voice traffic to be accepted when and only when expected performance can be assured before the voice traffic enters the voice gateways. Many factors may be considered in CAC, such as interface bandwidth, system resources of gateways, the network conditions, and configured policy control.

In the following sections, we show how system resources on voice gateways help to guarantee the QoS of voice traffic. System resource module is shown for call admission control and traffic engineering. Network conditions are important for real-time streams. We provide the end-to-end probing module to detect connectivity and congestion for delivering the traffic. All the modules discussed here can be used as a part of CAC to guarantee QoS, some procedures are recommended for integration based on the experience on H.323 VoIP calls in this paper.

2 System Resource Availability

System resources refer to the common resources on VoIP gateways in this paper. To be more specific, CPU, memory and call volumes are discussed in this study for providing better QoS for VoIP applications.

Voice gateways may be overloaded in some extreme cases. CPU utilization may go over 99% if huge bursty traffic is coming into voice gateways simultaneously, such as a large number of fax calls or Interactive Voice Response (IVR) calls. The memory consumption may be quite high if multiple IVR calls or fax calls are in process. Similarly, only a certain number of calls can be handled by voice gateways, otherwise the performance of voice gateways will be decreased.

In order to prevent the extreme cases from affecting the performance of all other processes on voice gateways, we provide the module for system resource measurement, and measure-based call admission control with per-call treatment and voice gateway busyout. The performance result is presented to demonstrate better serviceability and availability on voice gateways.

2.1 Two-threshold model

The two-threshold model is proposed here to catch abnormal cases. Namely, it has low and high thresholds. Whenever the current value is over the high one, the model remains in the unavailable state until the current value drops below the low one. For example, if CPU utilization is configured as [70%, 90%], and the current value is 92%, which is over 90%, that means that CPU is in unavailable state until the current value drops below 70%.

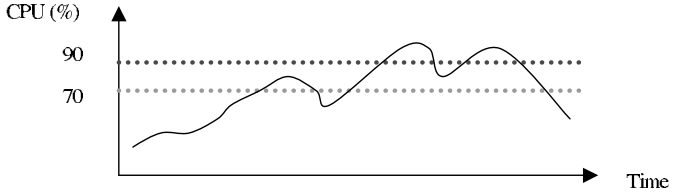


Figure 1: Two-threshold example

There are many advantages for this model. First, it is generic for modeling different resources. For call volume, the thresholds can be used as the number of calls. For memory, the thresholds can be either percentage or the absolute byte numbers. Another advantage of using two-threshold model is to avoid the spiking condition of some system resources. Moreover, the two-threshold model is a super set of one-threshold model. For a example, if CPU utilization is configure as [90%, 90%], it is the same as one-threshold model with the threshold defined as 90%.

On the other hand, it needs careful configuration for different gateways if resource-based call admission control is enable in the rest of this section. The proper values for low threshold and high threshold should depend on the administrator's requirement and expectation. They may be different among configured resources on different gateways, and have respective impact on the performance of voice gateways.

2.2 More features on call volume

Besides the option for extreme cases, call volume may also be used for traffic engineering. For example, the users can specify the different two thresholds for call volumes for the multiple access Voice gateways with the busyout enabled on each of them. This will help the switch to send the calls to the available access gateways

instead of continuously delivery of calls to the unavailable ones. Better load-balancing can be achieved if multiple gateways connected to the same switch have CAC enabled.

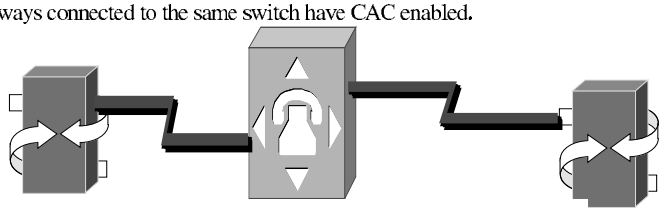


Figure 2: load-balancing example by call volume control

2.2 Resource-based CAC

Whenever the high thresholds are crossed on the configured system resources, including CPU average utilization, memory consumption and call volume, this will trigger the admission control module. We provide two options:

- *per-call treatment*: the new calls will not be accepted and be treated as the configured behavior, such as playing message saying "Please try another number ...." or playing different tones.
- *system denial*: the PSTN interfaces of the ingress gateways will be busied out to inform the PBXs or the switches to not send new calls to them.

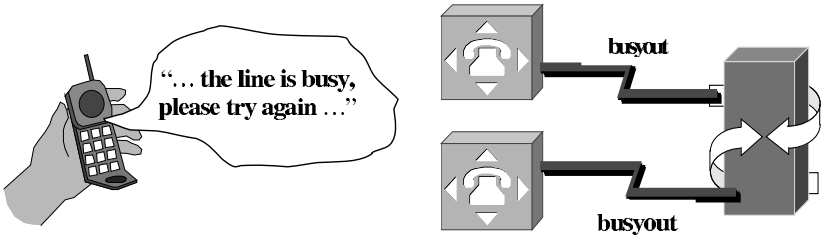


Figure 3: per-call treatment vs. system denial

By adding these options, system resource availability becomes a part of call admission control to guarantee the performance of VoIP applications. By per-call treatment or system denial, PBXs or the switches may reroute that call through the paths with good quality.

2.3 Performance Analysis

In our performance analysis, we chose system denial as the method for busying out the calls from coming into the voice gateways. Theoretically, given that the two-threshold model is used and the switches are blocked from sending new calls to worsen the unavailable situation, this should improve the call success ratio after the calls are connected.

2.3.1 Test topology

Figure 4 and 14 show the test topology on system resources concentrate on the originating voice gateways.

2.3.2 Overall CAC overhead

With the measurement turned on, the load for providing the measurement data and configured actions for busying out the trunks must be considered to ensure the introduced CAC load doesn't worsen the performance of voice gateways. The data in Figure 5 shows the load of CAC overhead can be ignored.

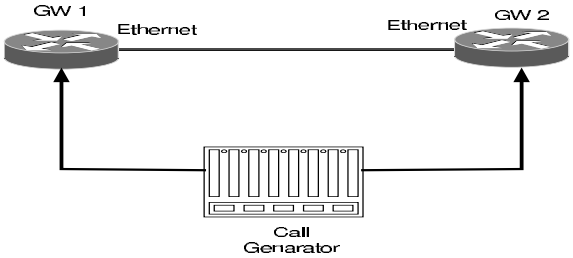


Figure 4: Overall CAC overhead Topology

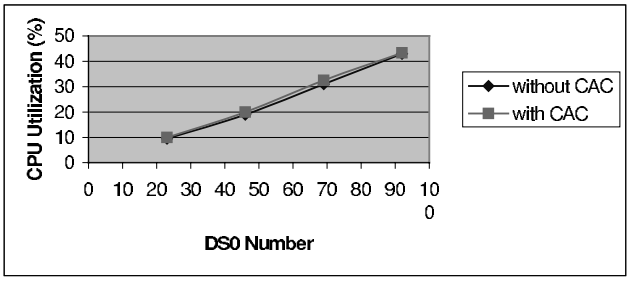


Figure 5: the CPU overhead for CAC

2.3.3 Call Success Ratio, delay, …

There are parameters that demonstrate that CAC based on system resources can provide the better serviceability and the availability. One of them is Call Success Ratio (CSR), which is the ratio of the final successful calls to the calls with successful setup.

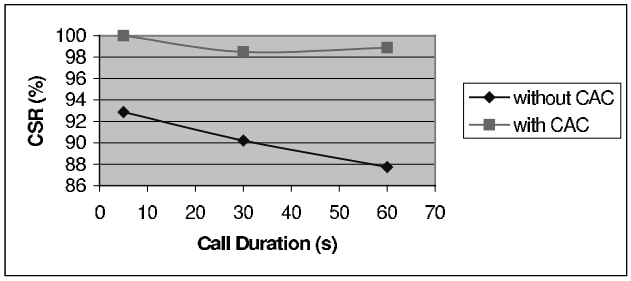


Figure 6: CSR comparison with different call durations

The tests here focus on the parameters on the originating gateways with the bursty data. Figure 6 demonstrates CSR is consistent and acceptable with CAC enabled. Without CAC enabled, more calls fail after successful call setups and CSR is dropping to unsatisfactory level.

Round Trip Delay (RTD) in this paper is measured by the call generator on the delay for voice path confirmation. Figure 7 shows the improvement of RTD is much better when CAC is enabled.

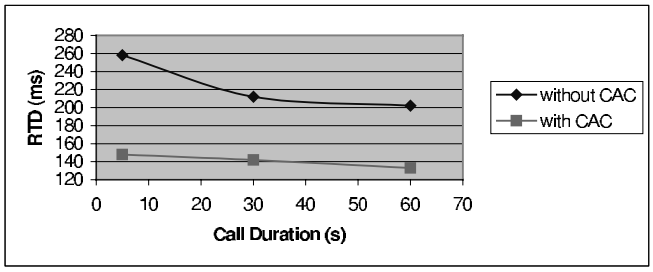


Figure 7: RTD comparison with different call durations

3 Network congestion measurement

Given the concern about the scalability of RSVP and some existing old Internet infrastructure, RSVP hasn't been deployed in all the routers in the current Internet. That implies that VoIP cannot fully rely on RSVP in many scenarios, at least at the current time.

QoS could be based on measurements for VoIP. Two of important factors in choosing the VoIP routes for QoS are network connectivity and availability.

If the network is down along the paths from the source to the destination, it's better to stop all the incoming VoIP calls and reroute them through traditional PSTN if there is no IP route. The same strategy applies the congested network. If the congested network cannot guarantee the QoS of VoIP calls, it's better stop new incoming VoIP calls and wait for the recovery from network congestion.

The most used parameters in determining QoS for VoIP applications are loss, delay, jitter and ICPIF (ICPIF short for Calculated Planning Impairment Factor, see ITU-T G.113). Many service providers need this kind of measurement as a part of CAC for VoIP applications.

3.1 Probe-based measurement

There are many ways for network measurement and management. One of them recently provided by Cisco is specific for VoIP applications through most of Cisco gateways. Parameters such as jitter, delay, and ICPIF are obtained through Response Timer Response (RTR) probes.

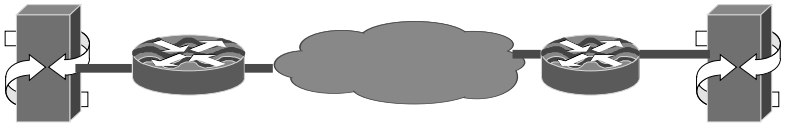


Figure 8: end-to-end network condition

RTR is the enhanced module for point-to-point probes. In addition to the traditional ICMP echo probes, RTR provides more features. One of them is to use configured IP ToS fields for ICMP echo probes. Another is to support new probes for UDP and TCP response time measurement. For the later, the far end must enable a RTR responder to listen to a UDP port (port 1976) for RTR control message authentication.

To make the probes more accurate, RTR probes allow the users to define the packet size, the number of packets and the interval between consecutive packets. First, VoIP packets are RTP/UDP, which can use RTR's

UDP response time. Second, based on the VoIP codec, we can use the proper packet size to simulate the actual voice packets. As loss, delay and ICPIF may rely on the codec, RTR probes provide a better way to discover QoS and the voice quality.

Based on the RTR probes that simulate the VoIP packets, there are usually two scenarios for service providers. One is the probes show the network is disconnected from the source to the destination. The reasonable action in the VoIP gateways is to busy out its proper PSTN interfaces, which prohibits the switches to send more VoIP calls to the same destination. As soon as the RTR probes indicate the network is connected again, the PSTN interfaces should be brought up again.

The other is the probes show loss rate, delay or ICPIF is too bad to provide the expected QoS for new traffic. An option is provided for the users to drop the new incoming VoIP calls to that destination whenever the configured thresholds are crossed for loss, delay or ICPIF.

There are some drawbacks for RTR probes. One is that they are asynchronous probes, which may not reflect the dynamic changes of network performance. Aother is that the probing packets are not the voice packets, which may not reflect the real treatment of voice packets from the source to the destination. Probe packets add extra load on the network, especially when the network is already congested. Therefore, this approach makes more sense for detecting network connectivity and then triggers the system denial on some voice interfaces.

3.2 RTR-based CAC

Whenever the probing results to the desired destinations are below the configured expections (such as delay, loss, or icpif), this will trigger the admission control module. We provide two options:

- *per-call rejection*: the new calls will not be accepted and be denied with configured cause code, such as no QoS available.

- *Selected system denial*: the users can select certain PSTN interfaces of the ingress gateways to be busied out to inform the PBXs or the switches to not send new calls to them until the probing results turn good.

.

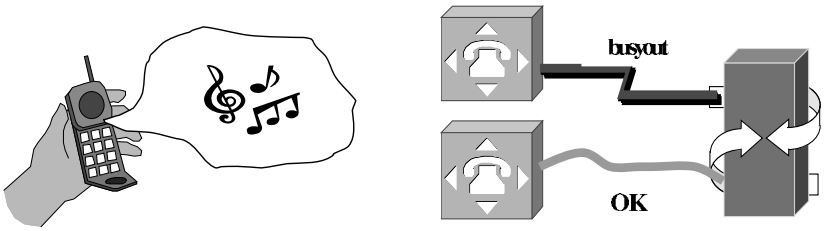


Figure 9: per-call rejection vs. configurable trunk busyout

Note that the selected system denial provides the flexibility of sharing the gateways for different ISPs or different users. For example, user A owns ISDN interface One and delivers calls to New York, and user B owns ISDN interface Two and delivers calls to Los Angels. A can configure to busy out ISDN interface One preventing calls from the switch if the network condition to New York is below expectation.

By adding these options, system resource availability becomes a part of call admission control to guarantee the performance of VoIP applications. By per-call rejection or selected system denial, the PBX or the switch may reroute that call through the paths with good quality.

3.3 Performance Analysis

In our performance analysis, we choose selected system denial as the method for busying out the calls from coming into the voice gateways.

3.3.1 Test topology

The tests on probing the network condition concentrate on the originating voice gateways.

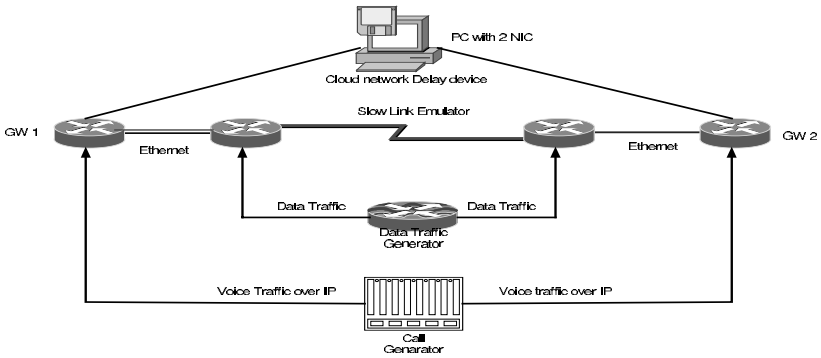


Figure10: Congested Network Topology

3.3.2 Overall CAC overhead

With the measurement is turned on, the load for providing the probes and configured actions for busying out the trunks must be considered to ensure the introduced CAC load doesn't worsen the performance of voice gateways.

It's not surprising that CPU utilization and memory consumption are actually lower with CAC enabled. As the system denial is functioning when the network condition is unsatisfactory, the switches are blocked from sending new voice calls into the gateways. Therefore the gateways use less systems resources.

The concern for the probing approach is the extra probing traffic introduced. This could worsen the congested networks or could not scale if the gateways want to deliver calls to a large number of destinations belonging to different domains. The extra load also depends how large the probing packet is. For example, if the probes, in G729, are updated by 10 packets every 10 seconds, then the bandwidth required is (20+12+8+20)*8*10/10 = 480 bps.

3.3.3 Call Success Ratio, delay, …

In this subsection, we demonstrate the RTR probes can provide the better serviceability and the availability through preventing voice traffic from entering congested networks.

In Figure 12, we compare the CSR in different traffic patterns in the tests. The percentage mentioned below is about the bandwidth of the slow link in the test topology. In Traffic pattern 1, the network traffic through the common link is 400 kbs voice and 400 kbs data. In Traffic pattern 2, 800 kbs voice and 800 kbs data. In Traffic pattern 3, 1100 kbs of voice and 1400 kbs of data. In Traffic pattern 4, 1100 kbs of voice and 1800 kps data. Note that voice traffic means the traffic the switches want to send to the voice gateways . With the CAC enabled, the channels are busied out and the voice traffic cannot enter the voice gateways.

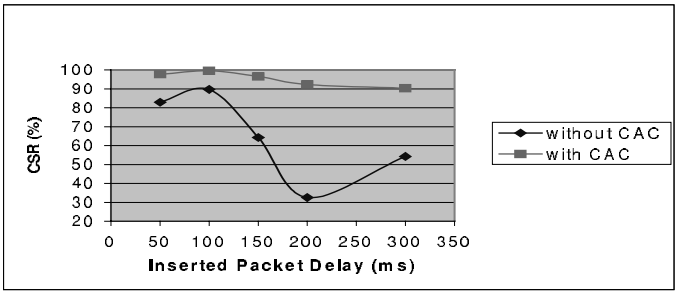


Figure 11: CSR comparison with packet delays

Figure 12 demonstrates that the CSR is quite good and consistent under different traffic patterns with CAC enabled. The reason for that is that voice gateways inform the switches of backing off whenever the network is congested. On the other hand, without CAC enabled, the voice calls keep entering the IP world even if the network is congested, which will introduce a lot of call failures due to packets drop such as no voice path failures.

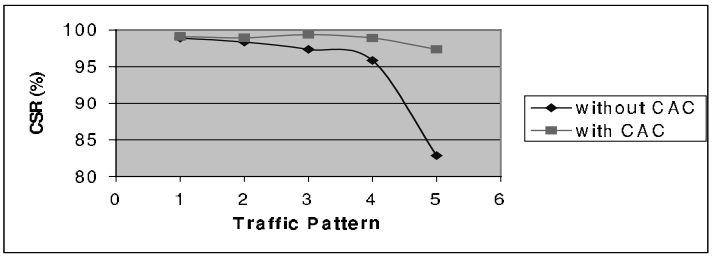
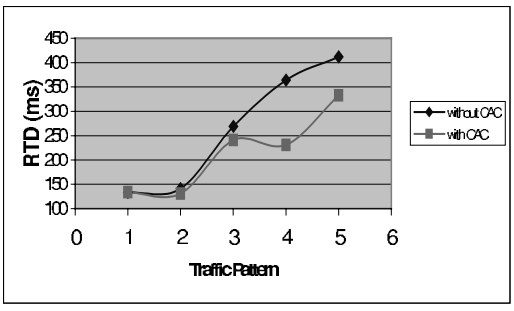


Figure 12: CSR comparson with different traffic patterns

Figure 11 shows the similar results when the certain number of delay is introduced in the network. Without CAC, the call failure rate is unacceptable even if the calls are connected. With CAC, the call success ratio is quite better.



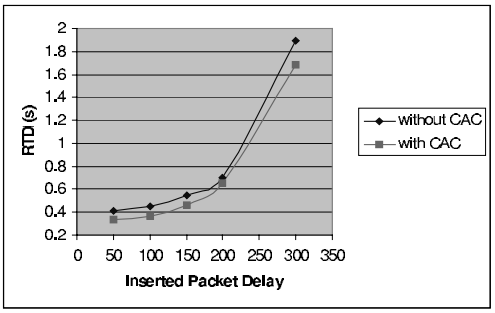


Figure 13: Round Trip Delay comparison with different traffic patterns

Figure 13 provides the same information about the improvement of RTD with CAC enabled. The reason is that calls are blocked when the network is congested. So RTD is better for calls when the network condition gets better.

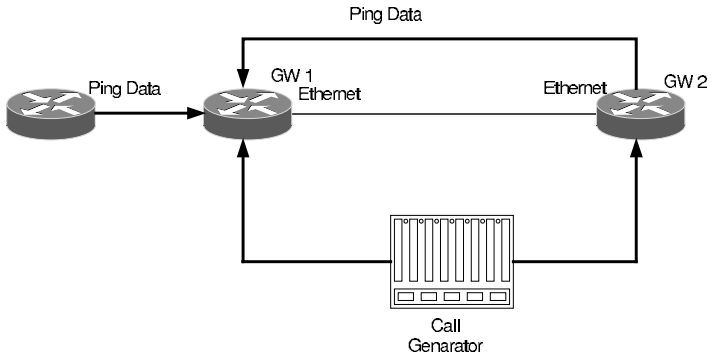


Figure14: Mixed Traffic Topology

4 Integration for CAC

In this paper, we demonstrate some modules for CAC on Voice gateways. With the help of these module, improved QoS will be provided for end-to-end VoIP applications. Each module is independent of each other, and can be used in different places. Based on our experience on VoIP H.323 calls, one procedure we recommend is

when the call is from telephony side,

1. The system resource should be checked first to decide if enough resource is available on the gateways. If yes, continue. If no, do call treatment if configured.
2. The outgoing interface resource should be checked by LCAC module for bandwidth and call volume. If yes, continue. If no, reject the call.
3. If RSVP isn't enabled, use end-to-end network congestion measurement module. If yes, goto 5.
4. If RSVP is enabled, use RSVP synchronized module to reserve the bandwidth for both directions.
5. allow the call to continue.

Similarly, when the call is from IP side,

1. the incoming interface resource should be checked by LCAC module for bandwidth and call volume. If yes, continue. If no, reject the call.
2. system resource should be checked to decide if enough resource is available on the gateways. If yes, continue. If no, do call treatment.
3. If RSVP is enabled, use RSVP synchronized module to reserve the bandwidth for both directions.
4. allow the call to continue.

This procedure shows how these modules work together to provide better QoS through call admission control. Other procedures may be used based on the signaling protocol and vendors' favors, and more modules can be added for call admission control.

5 Conclusion and Future Work

QoS is critical for the success of real time applications over IP, such as Voice over IP. On voice gateways, call admission control plays an important role for guaranteed QoS, for it makes decisions on whether and how to deliver the traffic based on different kinds of resources.

In this paper, we study the QoS issues on VoIP gateways through measurement-based CAC. To guarantee QoS, CAC must be provided on VoIP gateways, allowing the voice traffic to be delivered when and only when expected performance can be assured at the time voice traffic enters the VoIP gateways. Many factors may get involved in CAC, such as interface bandwidth, gateway system resources, the network conditions, ...

We show how system resources on voice gateways help to guarantee the QoS of voice traffic. System resource module is shown for call admission control and traffic engineering. Network conditions are considered by several approaches to detecting connectivity and congestion. All the modules discussed here can be used as a part of CAC to guaranteed QoS, some procedures are recommended for integration in our paper.

As the rapid progress is being made in providing QoS, there are a lot of issues for better Call admission control. For example, with RSVP enhancement for aggregation, CAC should be provided based on the aggregation policy instead of per call requirement. How to integrate some of the above CAC modules with DiffServ and MPLS is another interesting topic.

References

1. F. Cao, H. Salama, and D. Shah, "Approaches to Providing Guaranteed Quality of Service for VoIP through Call Admission Control on Voice Gateways", Proceedings of "International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet" (SSGRR 2000), July 31 – August 6, L'Aquila, Italy, 2000

2. S.Charttjerjee and J. Strosnider, A Generalized Admission Control Strategy for Heterogeneous, Distributed Multimedia Systems, Proceedings of the Third ACM International Multimedia Conferences, San Fransisco, 1995

3. J. Huang, Y. Wang, and F. Cao, On Developing Distributed Middleware Services for QoS- and Criticality-Based Resource Negotiation and Adaption, Journal of Time-Critical Computing Systems, 16, pp 187-221, 1999

4. J. Huang, Y. Wang, S. Vaidy, and F. Cao, GRMS: A Global Resource Management System for Distributed QoS and Criticality Support, Proceedings of the IEEE International Conference on Multimedia Computing and Systems (MMCS'97), Ottawa, Canada, 1997

5. D. Hutchison, G. Coulson, A. Campell, and G.S. Blair, Quality of Service Management in Distributed Systems, Distributed Systems Management, Ed. Morris Sloman, Imperial College London, 1995.

6. H. Kaneko, J.A. Stankovic, S. Sen, and K. Ramamritham, Intergrated Scheduling of Multimedia and hard-real-time tasks, Proceedings of the IEEE Real-time Systems Symposium, December, 1996.

7. A.M. van Tilborg and G. Koob, Foundations of Real-time computing-Scheduling and Resource Management, Kluwer Academic, 1991.

A SIMULATION ANALYSIS OF AGGREGATION STRATEGIES IN A WF^{2}Q+ SCHEDULERS NETWORK

R. G. Garroppo, S. Giordano, S. Niccolini, F. Russo
{r.garroppo, s.giordano, s.niccolini, f.russo}@iet.unipi.it

Department of Information Engineering
University of Pisa
Via Diotisalvi 2
56126 Pisa - Italy
Tel. +39 050 568511, Fax +39 050 568522

Abstract **– The paper presents an analysis in a DiffServ network scenario of the achievable QoS (Quality of Service) performance when different aggregation strategies between video and voice traffic flows are considered. Each network node of the analyzed DiffServ scenario is represented by a Worst-Case Fair Weighted Fair Queueing scheduler with a shaper obtained using a Shaped Starting Potential Fair Queueing. The system of each node, referred in literature as WF^{2}Q+ scheduler, permits to guarantee the isolation among the different PHB (Per Hop Behavior) service classes, maintaining the multiplexing gain. The parameters setting of the WF^{2}Q+ scheduler is also discussed. In particular, the necessary network resources, estimated by the WF^{2}Q+ parameters setting obtained considering the aggregation of traffic sources belonging to the same service class, are compared with those estimated on a per flow basis. The higher gain achievable using the first approach with respect to the second one, is also qualitatively highlighted. The simulation results, presented in the paper, evidence the possible problems that can be raised when voice traffic is merged with video service traffic. As a consequence, the paper results suggest to consider in different service class queues the two kinds of traffic.**

Keywords **–** WF^{2}Q+ scheduler, LBAP traffic characterization, DiffServ architecture, voice model, QoS parameters

1. Introduction

A key challenge of the current telecommunication age is represented by the developing of new architecture models for IP networks in order to satisfy the recent QoS requirements of innovative IP-based services (e.g. IP Telephony and videoconferencing).

At present, the ISP (Internet Service Provider) often provide the same service level independently from the traffic generated by their clients. Taking into account the transformation of Internet to a commercial infrastructure, it is possible to understand the need to provide differentiated services to users with widely different service requirements.

In this framework, the DiffServ approach is the most promising for implementing scalable service differentiation in IP networks. The scalability is achieved by considering the aggregate traffic flows and conditioning the ingoing traffic at the edge of the network. Aggregation obviously decreases the complexity of traffic control in the core network, but it produces some unwelcome effects, such as "lock-out" or "full-queues" phenomena, which contribute to increase end-to-end delay and jitter of the traffic flow of a single service. These two phenomena take place respectively when few flows monopolize queue space preventing other connections from getting in the queue and when it is not possible to maintain the queues non-full. Hence, the effects of the aggregation mechanisms on the QoS parameters of the different aggregated flows need to be further analyzed. The first works in this field have highlighted relevant concepts to support traffic aggregation [1], however a still open issue is what kind of aggregation strategies is better to carry out. To this aim we investigate traffic aggregation strategies because there is still no clear position on what is the better configuration (standardization organisms say anything regarding this matter). On the other hand recent publication [2] suggests to divide network traffic in only two service classes (e.g. real-time and non real-time) but it seems, from our point of view, a little bit restrictive with respect to different traffic features. In the paper, we analyze the impact of the aggregation of real-time video and voice traffic in the same service class (hence, in the same queue in a per-service queueing system) on the experimented QoS parameters of the different flows. The QoS concept used in this work is to be identified with the whole set of properties which characterize network traffic (e.g. in terms of resource availability, end-to-end delay, delay jitter, throughput and loss probability). The results obtained in this scenario are then compared with those obtained considering real time video and voice as separate traffic flows.

The DiffServ architecture [3] is a good starting point but it is useless if there is no teletraffic engineering background able to provide the needed differentiation. Therefore we should take into account also scheduling disciplines and their dimensioning, in order to understand if they may affect results (changing scheduling discipline change the way the flows are treated). Hence, we have firstly chosen to use one of the best work-conserving

scheduling algorithms (WF2Q+) instead of a non work-conserving one used in other analysis [4]. The choice derives from the assumption that the best the scheduling algorithm is (keeping acceptable its complexity) the better treatment a flow receives in terms of low end-to-end delay and jitter. Moreover, the parameters setting of the considered scheduling discipline has been analyzed as described is in Section 4.

In the analysis, we use as traffic characterization approach, the LBAP (Linear Bounded Arrival Processes) theory [5]. Furthermore, we investigate the effects on LBAP characterization of the multiplexing of the video traffic, instead of taking into account the simple sum of the traffic descriptors obtained with the single source. By means of simulation analysis we evaluate if the multiplexing gain derived from characterization of aggregated traffic does not affect the QoS parameters.

The rest of the paper is organized as follows. In Section 2 we present the simulation scenario while in Section 3 we describe the voice source model, the video and data traffic taken into account in the simulation analysis. In Section 5, the results are discussed while Section 6 summarizes the main results presented in the paper.

2. Simulation Scenario

Our simulation scenario mainly reflects the topology of a DiffServ domain of an IP network. The simulation scenario is implemented using the OPNET Modeler vers. 6.0.L, a powerful CAMAD (Computer Aided Modeling and Design) tool used in modeling communication systems and in analyzing network performance. The considered scenario is shown in Fig. 2.1.

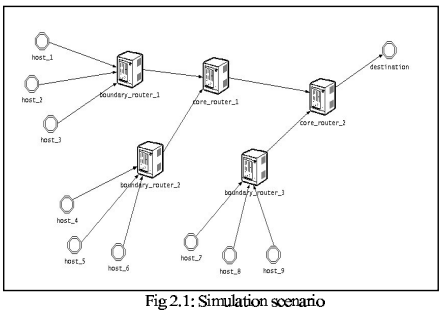


Fig 2.1: Simulation scenario

The network model is represented by edge and core routers, each one having a work conserving scheduler that permit to realize the isolation of the entering flows, based on performance guarantees. The scheduling discipline is a Worst-Case Fair Weighted Fair Queuing with the addition of a Shaper, obtained using a Shaped Starting Potential Fair Queuing (SSPFQ), denoted as WF^2Q+ [6]. The WF^2Q+ is a GPS (Generalized Processor Sharing) approximating service discipline with high fairness properties and relatively low implementation complexity. Moreover, in order to simulate a single *Diffserv* domain, we implement at the edge router the classifier and the marker necessary to associate each packet to the selected PHB. Based on this classification and marking, each packet receive the suitable forwarding treatment by the core routers. The traffic sources taken into account in the simulation scenario, are the most heterogeneous possible because we want to analyze the performance of a real network; it must integrate the carrying of video, voice and data traffic. Hence, describing the scenario shown in Fig. 2.1 in more details, every block named as host 1, 4 and 7 contains 15 voice sources, while every block named as host 3, 6 and 9 contains a video source. The remaining hosts contain "data" module, which simulate best-effort traffic.

The statistics we have collected concern the most significant QoS parameters of real-time services, i.e. end-to-end delay and jitter delay, which are evaluated considering the connection among the different sources and the destination node shown in Fig. 2.1.

3. Source models

In the simulations, we adopt a model only for the voice sources, while for the other kinds of traffic we consider actual traffic data.

The model used for the voice sources consists in an On-Off model, suggested by the typical behavior of a voice source with VAD (Voice Activity Detection): it is active or inactive depending on the talker is speaking or silent. Assuming that no compression is applied to voice signal, during active periods the source transmits at the constant bit rate of ν=64 Kbps (this corresponds to a standard PCM codec with VAD). In-depth analyses of this traffic source, shown in literature, have emphasized that the distribution of active and inactive periods lengths can be approximated by an exponential function [7], with mean values respectively equal to T_{on}=350 msec and T_{off}=650 msec. The packet size is 64 bytes, and considering the bit rate and the header overhead (40 bytes taking into account the RTP/UDP/IP header) the source generates one packet every 3 msec.

Video flow	Mean_rate (Mbps)	Peak_rate (Mbps)
GOLDFINGER	0.584	5.87
ASTERIX	0.537	3.54
SIMPSONS	0.446	5.77

Table 3.1 – Statistical parameters of considered video sources

The traffic data used for the video sources are described in [8], where also their statistical analysis is presented. They have been obtained collecting the output of an MPEG-1 encoder loaded by different sequences of movies half an hour long. Some relevant statistical parameters of the traffic data used, named Goldfinger,

Asterix and Simpsons, are summarized in Table 3.1.

The video packets are produced at application level, dividing the number of bytes produced by the encoder in the frame period, T=1/24 sec, in consecutive packets of size equal to 1500 bytes (in this case an MTU, Maximum Transfer Unit, of 1500 is supposed). Moreover, in the simulation we consider the 40 byte of overhead, related to the UDP/RTP/IP header, assuming that every packet transports 1460 byte of the traffic data registered at the output of the encoder.

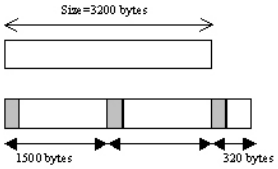


Fig 3.1 - An example of packet fragmentation at application level

The time interval between the generation of the consecutive packet in each frame period, has been considered deterministic and equal to T/N, where N is the number of packets needed to transport all the bytes produced by the encoder during a frame period. As an example, Fig. 3.1 presents a case where 3200 bytes are necessary for the encoding of a frame; at application level we divide the frame in three packets, which are sent with time interval equal to T/3 sec.

The Best Effort sources have been obtained considering the traffic data acquired at the Faculty of Engineering of the University of Pisa. In particular, we consider the traffic exchanges by the Faculty of Engineering with the external world (essentially other University sites and Internet) by means of an ATM network at 155 Mbps [9]. The peak rate of the considered traffic is equal to 11 Mbps, while a mean rate of only 400 Kbps has been observed; the high peak-to-mean ratio is a clear evidence of the high burstiness of the data traffic. During the data acquisition both the arrival time and the size of each packet have been registered. Hence, in this case the packet generation process to use in the simulations is directly obtained from the traffic data.

4. Parameters Setting

In order to set the scheduler parameters, we first characterize the traffic sources by mean of the LBAP approach. The traffic characterization of the single source is obtained by the parameters (b, ρ) where b is indicated as bucket size and ρ as token rate. The physical interpretation of the LBAP parameters can be understood, considering that the number of bytes produced by a single source in a time interval of $(0,\tau)$, $A(\tau)$, is upper bounded by

$$A(\tau) \le b + \rho\tau \qquad \forall\, \tau>0$$

The results presented in [10] permits to have an upper bound for the end-to-end delay experimented by the traffic when it traverses through a Latency Rate scheduler network, as that considered in our simulation scenario (i.e. WF^{2}Q+); estimation of WF^{2}Q+ latency term is described in [11]. In particular, the delay introduced by a single node to a packet belonging to the *i-th* flow, characterized by the LBAP parameters (b_i, ρ_i) is upper bounded by (in the following we suppose that for the *i-th* flow a service rate equal to ρ_i is allocated)

$$D \le \frac{b_i}{\rho_i} + \Theta_i$$

where Θ_i represents the latency of the scheduler, defined as $\Theta_i = \frac{L_{i,\max}}{\rho_i} + \frac{L_{\max}}{C}$. $L_{i,\max}$ and $L_{\max}$ respectively represent the maximum packet size of the *i-th* flow and of the global traffic arriving to the scheduler, while ρ_i and C, are the service rate allocated to the *i-th* flow and the global output service rate respectively. Extending the analysis to a network of K WF^{2}Q+ schedulers, the end-to-end delay experimented by a single packet belonging to the *i-th* flow is upper bounded by

$$D_i \le \frac{b_i}{\rho_i} + \sum_{j=1}^{K} \Theta_i^j ,$$

where Θ_i^j indicates the latency of the *j-th* node evaluated for the *i-th* flow.

The end-to-end delay bound has been obtained considering the worst-case analysis, which is more conservative with respect to experimented end-to-end delay.

Furthermore, as shown in [12], also the LBAP traffic characterization is conservative with respect to the statistical modeling approaches. These two considerations leads us to assume that the maximum end-to-end delay of the *i-th* flow can be upper bounded simply by

$$D_i \le \frac{b_i}{\rho_i} .$$

This hypothesis permits to establish the buffer size and the guaranteed rate to set in the scheduler, simply evaluating the LBAP curve of the *i-th* flow.

The simulation results that will be presented in Section 5 point out the goodness of our assumption, showing the very conservative nature of the worst-case analysis.

Considering the above hypothesis, the procedure used to set the scheduler parameters consists in evaluating the LBAP curve and in finding the point where this curve intersect the straight line $b_i = \rho_i D_i$, where D_i represents the maximum delay fixed for the considered source.

Figure 4.1 shows the presented approach in the case of the video sources. In particular, in the figure we can observe the LBAP curves for three different video sources, and the curve related to the traffic obtained aggregating these sources. Moreover, assuming a

maximum end-to-end delay of 200 msec, we can observe the relate straight line and the intersection points with each LBAP curves that, as described above, give the couple (ρ_i, b_i) to consider in the setting of scheduler parameters.

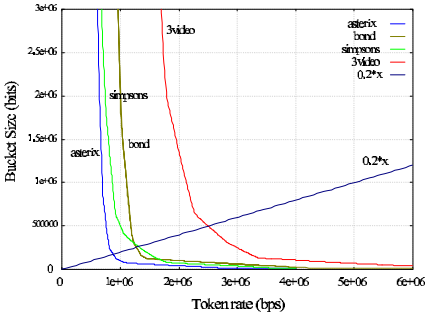


Fig. 4.1 - Video characterization

The characterization results obtained considering the considered three video flows (named Goldfinger, Asterix and Simpson), the aggregate of 15 voice sources (corresponding to a host in the simulation scenario) and the data traffic are reported in Table 4.1. In the table, the column Dmax indicated the maximum end-to-end delay analytically obtained from the estimation of scheduler parameters.

Traffic flow	**Rate** (ρ) (Mbps)	**Buffer** (b) (Kbit)	**Dmax** (ρ/b) (msec)
GOLDFINGER	1.25	250	200
ASTERIX	0.83	160	193
SIMPSONS	1.27	260	205
15 VOICE sources	1.10	30	27
Data	0.40	400	1000

Table 4.1 - Traffic characterization of considered sources

In the setting of the scheduler parameters, we can choice to set a service rate equal to the sum of ρ obtained for the three sources, and a buffer size equal to the sum of b obtained for each source. Considering this approach and the results obtained with the procedure presented above, the total service rate to allocate for the video services and the related buffer size are equal to 3.35 Mbps and 670 Kbits respectively. The other approach consists in the estimation of the parameters directly from the LBAP curve of the multiplexed traffic. In this case, it is expected to obtain a multiplexing gain in the setting of the resources to guarantee to the video service. In particular, considering the same upper bound of the end-to-end delay, we need to allocate a service rate of 2.5 Mbps and a buffer size of 500 Kbits. Then, in terms of buffer size we observe a gain of 170 Kbits (corresponding to a reduction of about 25%), while in terms of service rate a gain of 850 Kbps (25%) is achieved. Considering the service rate evaluated for each kind of traffic source, it is possible to set the scheduler parameter assuming a utilization factor of the link equal to 0.9. In more details, we evaluate the sum of ρ_i, ρ_{tot}, and fix the rate of the output link equal to $\rho_{tot}/0.9$. Hence, for boundary and core routers, the parameters are set as summarized in Table 4.2.

	Output Service Rate (Mbps)	Buffer Size, B (Kbit)
Boundary Routers	3.06	B_{voice}=30
		B_{data}=500
		B_{video}=250
Core router 1	6.12	B_{voice}=60
		B_{data}=1.000
		B_{video}=500
Core router 2	9.18	B_{voice}=90
		B_{data}=1.500
		B_{video}=750

Table 4.2 – Parameters of the routers

The buffer size is given for each traffic class, i.e. voice, video and data. When considering the scenario related to the aggregation of voice and video flows, the buffer size for this aggregated class of traffic has been set equal to $B_{voice}+B_{video}$.

5. Simulation results

The simulation analysis is mainly focused in the evaluation of the impact of different aggregation strategies on the QOS parameters. The low scalabity of the IntServ network architecture, suggests for the IP Telephony scenario to study a DiffServ core network architecture. Hence, it is expected that in each node of an IP Telephony core network, a per-class queueing is implemented and an appropriate scheduling algorithm guarantees the QoS requested by the real time sources.

Two relevant problems arise in this framework. The first concerns the choice of an appropriate scheduling algorithm and of a procedure for the setting of their parameters. The second issue is related to the aggregation strategies to adopt. Hence, in this Section we present the results that give insights on these problems.

We consider two different aggregation strategies: in a first one we have carried video and voice together in a premium class and data in a best effort class (in the figures the related curves are indicated with label containing the word "1 link"); in a second one we have supposed to carry video traffic in a separate queue from voice traffic (curves labeled with the noun containing the word "2 link").

Fig 5.1 presents the complementary probability of the end-to-end delay experimented by the voice packets when the two different strategies are considered and only one video source is activated (in this case the setting of simulation parameters have been changed according to the dimensioning procedure presented in the previous paragraph, in order to take into account the inactivity of the others video sources). Fig. 5.2 presents the same curves obtained when all three video sources are active.

As first analysis of the simulation results, it is possible to note that the adopted scheduling algorithm, i.e. WF^{2}Q+, and the procedure for its parameters setting permit to guarantee the target QoS for the voice sources if the video and voice traffic flows aren't merged in a single queue. In particular, the curves labeled as "1 Link" either in Fig. 5.1 and in Fig. 5.2 clearly show that the maximum delay observed during the simulation is under 10 msec, which is lower than the fixed delay of 27 msec, considered in the setting of the scheduler parameters.

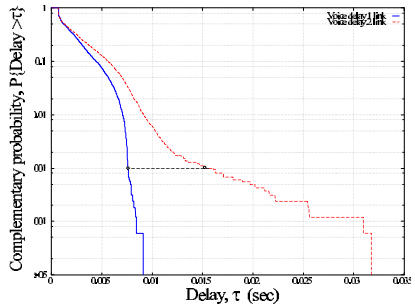


Fig 5.1 - Complementary Probability of Voice Delay: P{delay > τ} - One Active Video Source

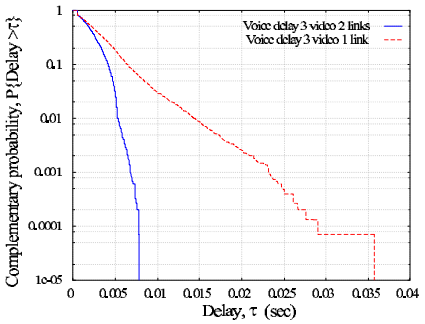


Fig 5.2 - Complementary Probability of Voice Delay: P{delay > τ} - Three Active Video Source

Both Fig. 5.1 and Fig. 5.2 show the degradation in terms of end-to-end delays of voice traffic when the video flows is merged in the same queue with the voice traffic. Indeed, in the Fig. 5.1, we can note that in correspondence of a probability P=0.001 a delay of 7 msec is observed in the first case (curve "1 Link"), which is lower than the 15 msec registered in the second case.

Furthermore, this degradation is amplified when the number of video sources is increased. Indeed, in Fig. 5.2 the maximum end-to-end delay registered for voice traffic is unvaried with respect to the previous case, i.e. about 7 msec, while it is increased to 23 msec, when all real-time flows are aggregated in the same queue.

Hence, in this second case the worsening of the delay parameter of voice service is due to the increase of the number of bursty traffic multiplexed with the voice sources. Hence, we can suppose that if there is more requested bandwidth the need for network resources increase in a non-linear way when considering a wrong strategy of aggregation, in this case the real time application may be damaged seriously.

On the other hand, the video performance take benefits from the aggregation, showing a little delay improvement that can be related to the multiplexing with the voice (the related figures are not reported for sake of simplicity).

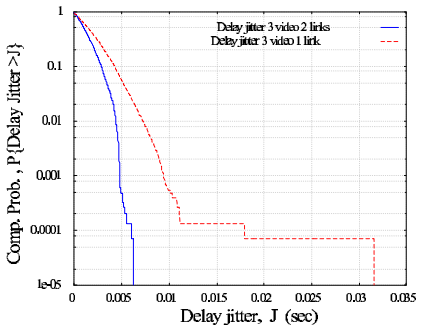


Fig 5.3 - Complementary Probability of Voice Jitter Delay: P{ delay jitter > τ} - Three Active Video Source

The different performance observed with the two considered aggregation strategies, can be related to "lock-out" phenomenon, which plays a decisive role in deteriorating voice performance. Indeed, when the video sources are merged with the voice traffic, the first monopolize the queue space obstructing the second from receiving the desired service level. The "lock-out" phenomenon can be avoided using different queues for video and voice traffic, while, at the same time, the choice of appropriate scheduling disciplines can guarantees an adequate multiplexing gain.

Finally, we observed that the best effort traffic (used as background traffic) is not affected by the fusion of the two service classes because the total bandwidth share (video + voice) is unchanged.

The same worsening of performance can be observed when we consider the jitter parameter, as shown in Fig. 5.3, which plots the complementary probability estimated for this statistic (the results are related to the multiplexing of three video sources).

6. Conclusion

The main goal of the paper is the evaluation of different traffic aggregation strategies for voice and video services in a *Diffserv* environment. In this framework, the simulation analysis presented in the paper highlights that the wrong aggregation of traffic flows with different

statistical features, such as video and voice traffic, may lead to performance worsening, which should be avoided especially in providing IP-based business services, such as IP Telephony.

On the other hand, the simulation results emphasize that with an adequate isolation between video and voice traffic flows and an appropriate dimensioning of network resources, it is possible to provide real-time services. In particular, the considered WF^2Q+ schedulers network and the proposed procedure for setting the related parameters, permit to achieve the target QoS. Furthermore, analyzing the proposed procedure for the setting of schedulers parameters, based on the LBAP traffic characterization, it has been possible to highlight the multiplexing gain obtainable considering the LBAP characterization of aggregated traffic.

Finally, the simulation results have evidenced that although we have neglect the latency terms in the expression of the end-to-end delay reported in literature, the maximum delay experimented is lower than that analytically estimated (see the fourth column in Table 4.1). This result is a further evidence of the very conservative nature of the worst-case analysis.

Acknowledgment

This work was partially carried out with the support of the EU in the framework of the Project MOICANE, IST-2000-7.1.3

References

[1] K. Dolzer, W. Payer, M. Eberspacher "A Simulation study on Traffic Aggregation in Multi-Service Networks", Conference on High Performance Switching & Routing (joint IEEE ATM Workshop 2000), Heidelberg, Germany, 26-29 June, 2000

[2] K. Dolzer, W. Payer "On aggregation strategies for multimedia traffic" Proceeding of the first Polish-German Teletraffic Symposium, Dresden, September 2000

[3] S. Blake, D. Blake, M. Carlson, E. Davies, Z. Wang, W. Weiss "An architecture for Differentiated Services", Internet RFC 2475, December, 1998

[4] H. Naser, A. Garcia, O. Aboul-Magd "Voice over Differentiated Services", Internet Draft, Diffserv Working Group, December, 1998

[5] S. Keshav "An Engineering Approach to Computer Networking", Addison-Wesley, January, 1998

[6] J. Bennet, H. Zhang "WF2Q: Worst-case Fair Weighted Fair Queueing", Proc. Of IEEE Infocom '96, March, 1996

[7] J. N. Daigle, J. D. Langford "Models for Analysis of packet Voice Communications Systems", IEEE JSAC, Vol. 6, pp 847-855, 1986

[8] O.Rose "Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems", Un.of Wuerzburg - Inst. Of Computer Science Research Report Series. Report N. 101. February 1995

[9] R.G. Garroppo, S. Giordano, M. Pagano, G. Procissi, "On the Relevance of Correlation Dependencies in On/Off Characterization of Broadband Traffic", Proc. of IEEE ICC 2000, New Orleans, Louisiana, USA, 18-22 June, 2000

[10] D. Stiliadis, A. Varma, "Latency-Rate Servers: A general model for analysis of traffic scheduling algorithms", Tech. Rep. UCSC-CRL-95-38, July 1995

[11] A. Charny, F. Baker et al. "EF PHB Redefined" Internet Draft, http://search.ietf.org/internet-drafts/draft-charny-ef-definition-01.txt, November, 2000

[12] R. Bruno, R.G. Garroppo, S. Giordano "Estimation of token bucket parameters of VoIP traffic", Proc. of IEEE ATM Workshop 2000, Heidelberg, Germany, 26-29 June, 2000

Speech Quality

Conversational Speech Quality - The Dominating Parameters in VoIP Systems

H. W. Gierlich; F. Kettler
HEAD acoustics GmbH
Ebertstraße 30 a, 52134 Herzogenrath, Germany

Abstract

Speech quality, especially in voice over IP systems is influenced by many parameters:

Delay has significant impact on the conversational quality mainly for two reasons: On the one hand the conversation between both subscribers and their interaction is more difficult. On the other hand the delay has a strong influence on the echo perception, longer transmission delay leads to an increased sensitivity in the users echo perception. It is important to evaluate the echo performance of a system in all conversational situations since typically additional artifacts like voice switching and background noise modulation may occur due to echo cancellation processes.

Voice switching may be introduced at many stages in a VoIP scenario. Voice switching may lead to speech quality degradation perceived as missing syllables or missing words. Since again the effect is subjectively perceived different in single talk situations as compared to double talk situations both situations have to be taken into account.

The transmission of background noise is a very critical parameter for the naturalness of a conversation. Due to clipping caused e.g. by VAD or the use of background noise reduction algorithms, background noise modulation - in combination with comfort noise injection - may degrade the perceived quality significantly. Packet loss may lead to speech quality degradation during single talk and double talk periods.

The article describes objective methods which allow the assessment of various speech quality parameters relevant for the conversational quality.

A. INTRODUCTION

In modern IP systems the communication using speech can no longer be regarded as it used to be in traditional PSTN networks. The differentiation between terminals and network is no longer possible. Typically the same signal processing algorithms are used in networks and in terminals. Basically three different configurations have to be taken into account:

IP gateway to IP gateway , IP terminal to PSTN terminal and IP terminal to IP terminal including the acoustical interfaces like hands-free phones, headsets or handset (see figure 1).

Since the interaction of the signal processing between terminal and network highly influences the speech quality, typically a complete configuration has to be considered for all testing including network and traffic load simulations.

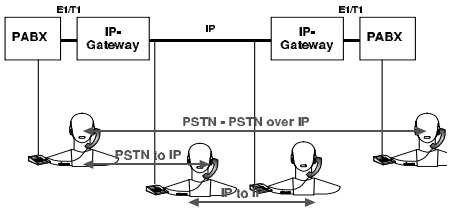


Fig. 1: The typical scenarios in VoIP networks

B. SIGNAL PROCESSING IN IP CONFIGURATIONS

Figure 2 introduces a typical block diagram for the signal processing from a speech transmission quality point of view.

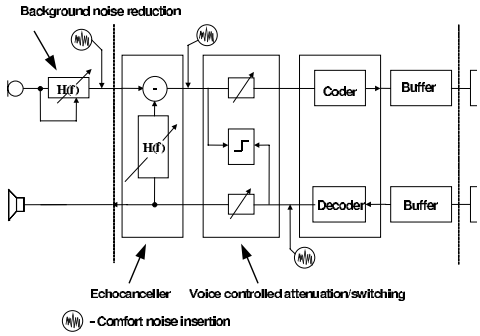


Fig. 2: Block diagram of typical components in voice over IP systems

The acoustical access is realized by acoustical components such as hands-free units, headsets or handset terminals. In any case sufficient echo control has to be provided in order to guarantee a sufficient echo loss even under worst case conditions. Due the influence of delay on echo perception (see[1]) these worst case conditions have to take into account the maximum expected delay due to the propagation delay, the delay variation ('jitter') in the network, packetization delay and buffer size in the equipment. The echo control can either be achieved by good acoustical terminal design or by using a speech echo canceller or any sort of voice activated switching (echo suppressor) which is shown in

the block diagram. Since in many cases the amount of echo cancellation is not a sufficient, speech echo cancellers typically are backed up by voice controlled amplification/attenuation systems in sending and receiving direction. Occasionally cancellation and switching is not realized in full bands but subband echo cancellers and attenuations are used. Sometimes additional signal processing components such as companding devices (AGC) etc. are introduced in order to enhance speech quality under special conditions e.g. noisy environments or a wide range of speech signal levels in the network. Typically voice activity detectors (VAD) are used in order to measure the voice activity and avoid the transmission of speech packets in case no voice activity is detected. In order to hide those silence intervals, comfort noise is inserted (typically on the far end side of a connection to save transmission bandwidth). In any case speech is coded using various sorts of a speech coding algorithms such as G.711, G.723, G.729. Buffers of variable size are used in order to guarantee the transmission of the speech with a minimum of packet loss under typical network and traffic conditions.

C. THE MOST DOMINATING PARAMETERS DETERMINING THE SPEECH QUALITY IN VOIP SYSTEMS

One of the dominating parameters influencing speech quality is delay. Delay may strongly impact the conversational quality but delay as well influences the delectability and annoyance caused by echo. Echo may occur in single talk as well as in double talk situations. In general the echo attenuation required to avoid customer complains is a function of delay and as soon as one way delay exceeds 250 ms, the echo attenuation has to be at least 46 dB. More information can be found in [1]. Due to various kinds of signal processing echo attenuation depends on robustness and adaptation speed of echo cancellers, the reliability of double talk detection, the sensitivity against background noise and other kinds of disturbances. Consequently the requirements on echo attenuation have to be checked under all these conditions. Specifically the double talk situation has to be taken into account. In order to assess this situation subjectively and find limits for the double talk situation auditory tests were conducted recently [2]. Specific double talk tests developed for this specific scenario were used. The test subjects had to rate the annoyance caused by echoes during double talk using MOS (Mean Opinion Scores) MOS 1 corresponds to a highly annoying echo, MOS 5 corresponds to "echo not perceptible". In Fig. 3 the difference in echo loss -always compared to the single talk situation- is shown in relation to the MOS rating.

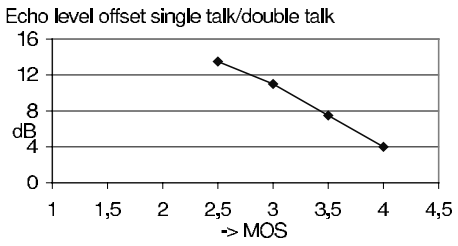


Fig. 3: Differences in echo loss requirement between the single and double talk situation as a function of perceived echo annoyance during single talk

From this results it is obvious that the requirements for echo attenuation during double talk are high and depend on the quality achieved in single talk situation. This means that in case the echo attenuation in single talk situations is poor the requirement for double talk may be low as well. From Figure 3 the echo loss requirement during double talk depending on the annoyance of echo perceived subjectively under single talk conditions can be derived .

Switching itself is a critical parameter under both single and double talk conditions. Basic requirements for switching in the single talk mode are known already since many years (e.g. ITU-T Recommendation P.340 [3]). The annoyance caused by switching, specifically front-end clipping was widely investigated by Sotscheck [4]. The most important result of these investigations is that front-end clipping of more than 15 ms should be avoided in any case. Frontend clipping is certainly not the only relevant parameter for VoIP scenarios since packet loss causes more statistically distributed temporal clipping. Without any kind of error correction (packet loss concealment), the speech gaps resulting from packet loss cause a high degradation of speech sound quality and may even reduce the intelligibility of speech significantly.

More critical however is switching during double talk. Since during double talk typically echo cancellation is less efficient again switching is introduced by additional nonlinear processes in conjunction with the echo canceller function (e.g. loss insertion to guarantee the necessary overall echo attenuation). If a loudness variation between the single and the double talk situation is more than 3 dB, the speech quality is degraded. Figure 4 gives the relationship between the annoyance caused by loudness variations as a function of attenuation range. Again specific double talk tests have been used for the tests. MOS 1 corresponds to highly annoying switching

whereas MOS 5 corresponds to switching not perceptible. For more information again see [2].

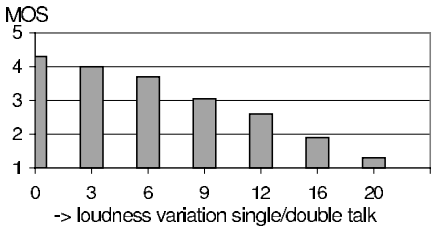


Fig. 4: Annoyance caused by loudness variations between single talk and double talk

When regarding the speech sound quality the speech quality may be highly influenced by packet loss and jitter. Packet loss in combination with various codecs may degrade the speech quality significantly. In order to evaluate this degradations methodologies based on the human perceptions of speech quality have been developed. Such methods are PSQM [5], TOSQA [6] and the new PESQ [7]. A typical example of the correlation between the quality perceived subjectively and the predicted speech quality is shown in figure 5. The results are expressed in mean opinion scores where MOS 1 represents unacceptable quality and MOS 5 represents a very high quality.

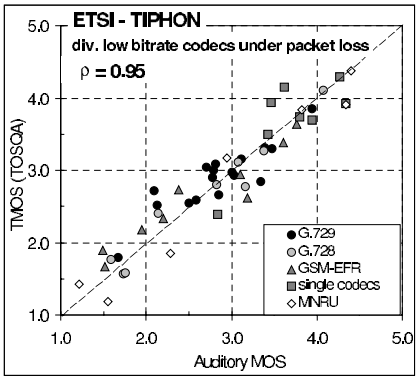


Fig. 5: Comparison of MOS values derived from auditory tests and MOS values predicted by TOSQA (TMOS) (see [6])

Those methods can be used for known codecs and for known impairments in order to evaluate speech sound quality. All those methods are perceptional based, but allow speech quality evaluations only in single talk conditions and unfortunately no method has been validated yet in order to evaluate the speech quality of systems including terminals.

Those methods can be used for known codecs and for known impairments in order to evaluate speech sound quality. All those methods are perceptional based, but allow speech quality evaluations only in single talk conditions and unfortunately no method has been validated yet in order to evaluate the speech quality of systems including terminals.

The transmission quality of background noise is - from the subjective point of view - one of the most important parameters. Since background noise is also transmitted during phases where no speech is present, background noise must be regarded as a signal. The transmitted background noise contains important information for the conversational partner about the environmental conditions. Echo cancellers, VAD's, DTX, and other algorithms are used in order to eliminate speech pauses/background noise in order to reduce bandwidth for transmission. Such operations may strongly degrade the quality of background noise transmission since pauses typically are filled by comfort noise which may interact with the transmitted background noise and lead to modulation of background noise.

D. Evaluation Methods and Examples

The most important evaluation methods for the dominating parameters influencing the speech quality in its various aspects are given using selected examples.

1.. Background noise transmission quality

Since the transmission of background noise subjectively is one of the most important parameters and in addition it is highly affected by the signal processing in IP connections the objective measurement of this parameter is of high importance as well. When analyzing the background noise transmission mostly the influence of the transmission system on the background noise is of importance, any structure inherent to the background noise itself should not influence the result of the analysis, such it is useful for analysis purposes to start with the evaluation of more or less constant background noises (e.g. office-type noises, street-type noises, interior vehicle noises for mobile terminals). The analysis methods used for that purpose is the "relative approach" [8]. The algorithm does not use any reference signal, but is working directly on the transmitted background noise signal. It takes into account the sensitivity of the human

ear on a signal fluctuation in the time domain as well as on dominant spectral structures. It is recognized that slow variations of a signal in time and/or frequency are typically not disturbing which is taken into account by the algorithm. The algorithm is based on forward estimation using the signal history. The new signal examples are predicted and compared to the actual acquired signal. The basis for the procedure is the hearing adequate spectral representation of the time and frequency domain. The basis therefore is a hearing model according to [9]. The nonlinear relationship between sound pressure level and loudness perceived subjectively is taken into account by time/frequency warping in a Bark filter bank and proper integration of the individual outputs. The filter bank is realized in the time domain. The output signals of the filter bank are rectified and integrated, thus the envelope is generated. The three-dimensional output of the hearing model is the basis for the relative approach. In each critical band long term level (integration time: 2-4s) is compared to the short term level (2ms).

An overall value can be derived for example by applying the following equation (see [8]):

$$Q = f(N,S) + f\left(\sum_{i=1}^{24}\left[\left|F_G(i-1) - F_G(i)\right| \cdot w_1(i,F_G(i)) + \sum_{n=1}^{T}\left|F_G(i,n) - F_G(i,n+1)\right| \cdot w_2(i,F_G(i))\right]\right)$$

where $F_G(i)$ is a mean value of the critical band level over a period T of 2 to 4 seconds, $F_G(0) = F_G(1)$, $F_G(i, n)$ is a mean value of the critical band level over a much shorter period (approx. 2 msec), n is the current (time-dependent) value. The weighting factors $w_1(i,F_G(i))$, $w_2(i,F_G(i))$ depend on the critical band level $F_G(i)$. In addition the overall value is influenced by the function f (N,S) which describes an auditory factor, dependent on loudness N and sharpness S.

When just displaying the result in the time/frequency domain a typical result looks like the picture below. The difference is color-coded and represented as a type of spectrography: high color represent big differences between estimation and actual signal, dark colors indicate low differences.

Fig. 6 shows the analysis result of a background noise reduction algorithm used in a hands-free terminal. The upper part of the picture shows the time response (light color) when switching on the hands-free terminal in the presence of background noise (the dark color indicates the time signal of the background noise signal).

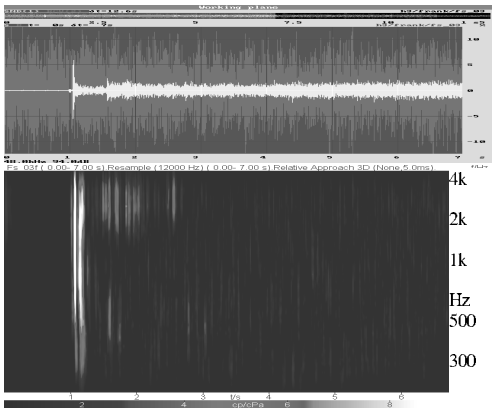


Fig. 6: Relative Approach Analysis,

upper: corresponding time signal
(light: transmitted background noise signal,
dark: background noise signal)
lower: Relative Approach analysis
(light: annoying signal components)

From the time signal it is not very obvious whether the algorithms works properly or not. When however analyzing the result of the relative approach it can be seen that the high peak in the beginning of the adaptation process is auditory annoying, the energy is spread over the whole frequency range for a small period of time. Within the first two seconds additional structures can be found between 1.6 and 3.4 kHz which are obvious in the relative approach and are obvious as well during the auditory judgement of the background noise transmission. Similar results could be achieved for other sorts of background noises and algorithms. Research work is ongoing in order to quantify the result of the relative approach analysis.

2.. Duplex performance

During double talk speech quality may be affected mainly by echo and/or switching. Special test signals an analysis techniques were developed in order to simulate the double talk behavior in the most realistic way but being able to analyze reliably and repeatable this situation. One test signal configuration used is shown in Fig. 7.

The signals are based on a CS-signal [10] simulating voiced/unvoiced sounds of speech, typical power density, modulation and distribution of speech using deterministic signals: voiced sound in the beginning followed by a pn-sequence and a pause. The double talk signal is constructed similar but uncorrelated to the test

signal. During the pauses of the double talk signal -these are the time intervals where for real conversations echo and switching is audible- evaluations on echo attenuation and switching can be made.

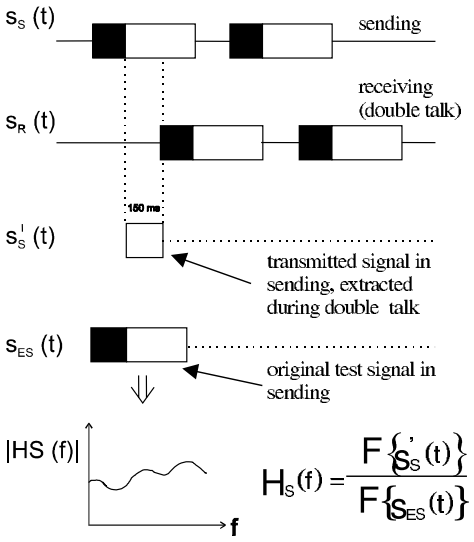


Fig. 7: Test signal for double talk evaluations

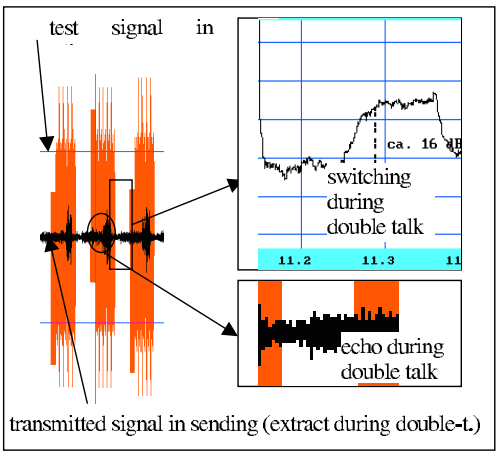


Fig.8: Example of extracting signal components for switching and echo evaluation during double talk in sending direction

Fig. 8 gives one example how to extract the information about switching and echo during double talk based on the test introduced with Fig. 7. Switching is evaluated during periods where only the signal in sending direction is present. The echo is determined during periods, where only the double talk signal is present and no signal in sending direction should be expected. In general all these investigations have to be conducted under various network conditions in order to get a range of performance requirements.

E. SUMMARY

The present article discusses impairments which may occur in modern VoIP systems during speech communication. Starting from functional units typically found in connections a short overview about signal processing is given. Instrumental procedures which may be used for the investigation of the dominating speech quality parameters in VoIP systems are introduced. Methods and performance requirements are based on auditory tests and cover the conversational situation including the transmission of background noise. Especially for the double talk situation and the background noise transmission the evaluation procedures and the underlying principles namely the auditory test results are given.

F. ACKNOWLEDGEMENTS

Part of the work was conducted in project with T-Nova Deutsche Telekom Berkom.

G. REFERENCES

[1] ITU-T Recommendation G.131
[2] F. Ketler; Gierlich, H.W.; Diedrich, E. Echo and Speech Level Variations During Double Talk Influencing Handsfree Telephones Transmission Quality, IWAENC 99, 27- 30.9.1999, Pocono Manor, USA
[3] ITU-T Recommendation P.340
[4] Sotscheck, J.: Über die Wahrnehmbarkeit von Clipping-Erscheinungen am Wortanfang, DAGA 90, pp. 1119-1122
[5] ITU-T Recommendation P.861
[6] Berger, J.: Instrumentelle Verfahren zur Qualitätsschätzung, Ph.D. Thesis, 1998, Shaker Verlag, ISBN 3-8265-4091-3
[7] Draft ITU-T Recommendation P.862
[8] Genuit, K. Objective Evaluation of Acoustic Quality Based on a Relative Approach, Internoise '96, Liverpool, UK
[9] Sottek, R.: Modelle zur Signalverarbeitung im menschlichen Gehör, PHD thesis RWTH Aachen, 1993
[10] Gierlich, H. W.: A Measurement Technique to Determine the Characteristics of Hands-Free Telephones, Signal Processing, Vol. 27, Issue 3, 1992

Impact of Packet Loss Location on Perceived Speech Quality

L. F. Sun, G. Wade, B. M. Lines, E. C. Ifeachor
Department of Communication and Electronic Engineering,
University of Plymouth,
Drake Circus, Plymouth PL4 8AA, United Kingdom,
{ L.F.Sun@jack.see.plym.ac.uk, j.wade@plymouth.ac.uk, B.Lines@plymouth.ac.uk, E.Ifeachor@plymouth.ac.uk}

Abstract – In VoIP applications, packet loss can have a major impact on perceived speech quality. The impact is affected by factors such as packet loss size, loss pattern and loss locations. In this paper, we report an investigation into the impact of loss location on perceived speech quality and the relationships between convergence time and loss location for three different codecs (G.729, G.723.1 and AMR) using perceptual-based objective measurement methods (PSQM+, MNB and EMBSD). Our results show that loss location has a severe effect on perceived speech quality. The loss at unvoiced speech segments has little impact on perceived speech quality for all codecs. However, the loss at the beginning of voiced segments has the most severe impact on perceived speech quality. The convergence time depends on the speech content (voiced/unvoiced). For unvoiced segments, the convergence time is stable whereas for voiced segments it varies but has an upper bound at the end of the segment. Our method allows a more accurate measurement of the exact effect of packet loss on perceived speech quality. This could help in the development of a perceptually relevant packet loss metric, which could be valuable in non-intrusive VoIP measurements.

Keywords – Voice over IP, Packet loss, Speech quality, Objective perceptual measurement, Codecs, Concealment performance

I. INTRODUCTION

Packet loss is a major source of speech impairment in voice over IP (VoIP) applications. Such a loss could be caused by discarding packets in the IP networks due to congestion or by dropping packets at the gateway/terminal due to late arrival. The impact of packet loss on perceived speech quality depends on several factors, including loss pattern, codec type, and packet loss size [1][2]. It may also depend on the location of loss within the speech.

In modern codecs (e.g. G.729, G.723.1 and Adaptive Multi-Rate, AMR codec), internal concealment algorithms are used to alleviate the effects of packet loss on perceived speech quality [3][4][5]. When a loss occurs the decoder derives the parameters for the lost frame from the parameters of previous frames to conceal the loss. The loss also affects subsequent frames because the decoder takes a finite time (the convergence time) to resynchronise its state to that of the encoder. Recent research has shown that for some codecs (e.g. G.729) concealment works well for a single frame loss, but not for consecutive or burst losses [1], and that the convergence times are dependent on speech content. Further, the effectiveness of a concealment algorithm is affected by which part of speech is lost (e.g. voiced or unvoiced). For example, it has been shown that concealment for G.729 works well for unvoiced frames, but for voiced frames it only works well after the decoder has obtained sufficient information [6]. Further, the decoder fails to conceal the loss of voiced frames at an unvoiced/voiced transition. Thus, the location of packet loss in relation to different parts of speech is important.

In most studies [1][6], the analysis of concealment performance and convergence times is based on the mean square error (MSE) and signal-to-noise ratio (SNR) criteria (with subjective or perceptual-based objective methods only used to assess overall quality under stochastic loss simulations). The perceptual impact of concealment algorithms or convergence times for different loss locations is still unknown. It is important to understand the effects of loss location and loss pattern on perceived speech quality, for different types of codec, to allow a more accurate measurement of voice quality. This requires the use of perceptual-based objective methods in the analysis. This could be helpful in setting up more efficient speech recovery system and for the development of perceptually relevant packet loss metrics which could be valuable in non-intrusive VoIP measurement.

The IETF has recently proposed a set of new metrics for packet loss [2]. This includes loss constraint distance (i.e. distance threshold between two losses) and "noticeable" loss rate (i.e. percentage of lost packets with loss distances smaller than loss constraint distance). For the same loss rate, different loss patterns may have different effects on perceived speech. In VoIP applications, the loss constraint is related to the convergence times of the decoder. However, it is still unclear how to determine the loss constraint threshold and whether (or how) the threshold is related to codec type, burst size or speech.

The aims of the study reported in this paper are two fold: (1) to investigate the impact of loss location on perceived speech quality and hence the concealment performance of codecs, and (2) to investigate the relationships between convergence times and loss locations/speech content, codec type or loss size.

The work reported here is based on three codecs – two existing codecs (G.729B [13] and G.723.1) and a new codec (AMR [7][14]) for VoIP. Three major perceptual distance measurement algorithms (PSQM/PSQM+ [8][9], MNB [10][11] and EMBSD [12]) are used for perceptual performance analysis for different loss location. Each

algorithm quantifies perceptual quality, but has a different range of perceptual distance.

The results show that the loss location has a severe effect on perceived speech quality. The loss at unvoiced speech segments has little impact on perceived speech quality for all three codecs. However, the loss at the beginning of voiced segments has the most severe impact on perceived speech quality. The extent of the impact depends on the size of the burst loss and codec type. The convergence time depends on the speech content. For unvoiced segments the convergence time is stable whereas for voiced segments it varies but constrained by the duration of the segment.

The remaining sections of the paper are structured as follows: Section II presents a brief overview of the codecs used and their concealment algorithms. The perceptual distance measurement algorithms (PSQM/PSQM+, EMBSD and MNB) are summarised briefly in Section III. The simulation system is described in Section IV, the experiments, results and their analysis are given in Section V. Section VI concludes the paper.

II. CODECS AND THEIR INTERNAL CONCEALMENT

A. Codec types - G.729, G.723.1 and AMR

The G.729 CS-ACELP (Conjugate Structure Algebraic Codebook Excited Linear Prediction, 8 Kbps) and G.723.1 (MP-MLQ/ACELP: Multipulse excitation with a maximum-likelihood-quantizer/Algebraic Codebook Excited Linear Prediction, Dual rate: 5.3/6.3 Kbps) are both standardized by the ITU and have been used in VoIP applications. The AMR (Adaptive Multi-Rate, ACELP) speech codec was developed by ETSI and has been standardized for GSM. It has been chosen by 3GPP as the mandatory codec. The AMR is a multi-mode codec with 8 narrow band modes with bit rates between 4.75 to 12.2 Kb/s. Mode switching can occur at any time (frame-based). AMR speech codec represents a new generation of coding algorithms which are developed to work with inaccurate transport channels. The flexibility on bandwidth requirements and the tolerance in bit errors of AMR codecs are not only beneficial for wireless links , but are also desirable for VoIP applications.

The three codec types belong to CELP analysis-by-synthesis hybrid codec. At each speech analysis frame, the speech signal is analysed to extract the parameters of the CELP model (Linear Prediction, or LP filter coefficients, adaptive and fixed codebooks' indices and gains). For stability and efficiency, LP filter coefficients are transformed into Line Spectral Frequencies, or LSF's for transmission. These parameters are then encoded and transmitted. At the decoder, the parameters are decoded and speech is synthesized by filtering the reconstructed excitation signal throuth the LP synthesis filter.

The major differences between the three codecs lie in the excitation signals, the partitioning of the excitation space (the algebraic codebook), delay and the way in which the coefficients of the filter are represented. For example, the G.729 uses two stage codebook structures for LSP parameters and gets the name "conjugate structure".

The frame sizes for the three codecs are 10 ms (80 samples at 8 kHz sampling) for G.729, 20 ms (160 samples) for AMR and 30 ms (240 samples) for G.723.1. They all have voice activity detection and silence suppression processing. The frames are classified as normal speech frame, SID (Silence Insertion Description) frame and null frame (non-transmitted frame).

B. Codec Internal Concealment

All three codecs have built-in concealment algorithms, which can interpolate the parameters for the loss frames from the parameters of the previous frames. For example, for the G.729 the concealment algorithm works in accordance to the following steps:

- The line spectral pair coefficients of the last good frame are repeated
- The adaptive and fixed codebook gain are taken from the previous frame but are damped to gradually reduce their impact.
- If the last reconstructed frame was classified as voiced, the fixed codebook contribution is set to zero. The pitch delay is taken from the previous frame and is repeated for each following frame. If the last reconstructed frame was classified as unvoiced, the adaptive codebook contribution is set to zero and the fixed codebook vector is randomly chosen.

III. PERCEPTUAL SPEECH QUALITY MEASURE – PERCEPTUAL DISTANCE

Perceptual distance is used to measure the perceptual difference between a reference speech signal and a degraded speech signal. It normally includes a perceptual model and a cognition model to mimic the process in the human's hearing perceptual process. Various perceptual speech quality measurement algorithms exist with different perceptual or cognition models.

PSQM (Perceptual Speech Quality Measurement) developed by KPN has been adopted as ITU-T Recommendation P.861 for assessing the speech quality for codecs [8]. PSQM+ was proposed by KPN to improve the performance of PSQM for loud distortions and temporal clipping [9]. PSQM/PSQM+ can generate a perceptual distortion value for each frame (32 ms for 8 kHz sampling, with 50% overlapping) and the overall PSQM/PSQM+ value is calculated for the whole test sentence via different weighting factors for silence or non-silence frames. As PSQM+ provides a more accurate measure of perceived speech quality under frame loss situations, we have chosen it

for overall perceived speech quality and perceptual distance calculation for each frame.

The MNB (Measuring Normalizing Blocks) developed by the US department of Commerce [10][11], is included as an Appendix in ITU-T P.861 Recommendation. The MNB does not generate a distortion value for each frame since each MNB is integrated over frequency or time internals.

EMBSD (Enhanced Modified Bark Spectral Distortion) was developed by Temple University in USA [12]. It estimates speech distortion in the loudness domain taking into account the noise masking threshold in order to include only audible distortions in the calculation of the distortion measure. As EMBSD only takes into account the non-silence frame for the final perceptual distortion calculation, the setting of the threshold of silence or non-silence will affect the final result.

In the paper, MNB and EMBSD are used for the overall quality measurement.

IV. SIMULATION SYSTEM

In order to investigate the impact of packet loss location on perceived speech quality, and the relationships between convergence time and loss location, we set up a simulation system. This includes speech encoder/decoder, loss simulation, perceptual quality measure and convergence time analysis, as shown in Figure 1. For codecs, we have a choice of G.729, G.723.1 and AMR. The standard 16 bit, 8 kHz sampled speech signal is processed by the encoder first. Then the parameter-based bit stream is sent to the decoder without frame losses (speech quality degradation in this case is only due to codec). The bitstream is also sent to the loss simulation module where the loss position and frame loss size can be selected. After loss simulation the bit stream is processed by the decoder to obtain the degraded speech signal with loss. The overall perceptual speech quality is measured between the reference speech signal and the degraded speech signal with loss by calculating the perceptual distance values using the PSQM+, MNB and EMBSD algorithms. The perceptual distance for each frame is also measured between the degraded speech without loss and the degraded speech with loss using PSQM+ for the analysis of convergence time. This eliminates coding impairment from the computation. The convergence time is also calculated using the normal Mean Square Error (MSE) method [1].

Loss simulation for each codec differs from the loss specification in the codecs. For G.729, if a parameter byte in the bit stream is set to zero, the frame is treated as a loss by the decoder and concealment is initiated automatically. For AMR, there is an extra byte for the transmit/receive frame type. For a lost frame, there is only a need to set the type as a BAD/ERASED frame. For G.723.1, a loss location mark file is created and serves as the input to the decoder.

V. EXPERIMENTS AND ANALYSIS OF RESULTS

A. Loss location and perceived speech quality

In the first experiment, the impact of loss position on the overall perceptual speech quality or the performance of concealment under different loss locations is investigated. The PSQM+, MNB and EMBSD perceptual distance values are calculated for the whole test speech sentence (about 6 seconds), while only one loss is produced each time and the loss position moves smoothly from left to right. The move is one frame each time and the frame size is decided by the codec chosen. At each loss location , the frame loss size can change by one, two, three or four frames to simulate different packet size or burst loss size.

The waveform for the first talkspurt for the test sentence "Each decision show (s)" is shown in Figure 2. It consists of four voiced segments - V (1) to V (4) corresponding to the vowels 'i', 'i', 'ə' and 'au'. The voiced segments are separated by unvoiced segments.

The overall perceptual distance values for PSQM+, MNB and EMBSD for G.729 are shown in Figures 3, 4 and 5, respectively. The values (using PSQM+) for G.723.1 (6.3 Kb/s) and AMR (12.2 Kb/s and 4.75 Kb/s mode) are shown in Figures 6, 7 and 8. In all the figures, the horizontal scales are in the unit of frames. As the frame sizes are 10, 20 and 30ms for G.729, AMR and G.723.1, respectively, the total number of frames for the test segments shown are 134, 67 and 45.

Examination of Figure 3 shows that the perceptual distance value varies between 1.4 and 2.4 as the loss location moves from left to right. In the PSQM+, a change in perceptual distance indicates a change in perceptual speech quality (the smaller the distance, the better the perceived quality). Similar changes in perceived speech quality can also be seen for the MNB (Figure 4) and EMBSD (Figure 5), as well as for the different codecs (Figure 6, 7 and 8). It is evident that the same loss condition (one packet loss for the whole test speech segment) causes an obvious variation in overall perceived speech quality, but the variation is dependent on speech content. A loss at unvoiced speech segments shows little impact on perceived speech quality (almost the same perceptual distance values as for no-loss cases). However, a loss at voiced segments has different effects on perceived speech quality depending on its location within the voiced segment. At the beginning of a voiced segment, it has the most severe impact (the peaks in the figures). At the end of voiced segments, the impact is small. In the middle voiced segments, perceptual distances change depending on the codec and frame loss size. For example, for the G.729 one-frame loss (Figure 3), the perceptual distance value reaches its peak when the loss is at the beginning of voiced segments. Then, as the loss position moves to the right (for each voiced segment), the perceptual

distance rapidly returns to the minimum value, showing a good convergence performance for voiced segments 1, 2 and 3. For voiced segment 4, the value varies depending on the speech content. As the frame loss size increases, the perceptual distance increases.

We explain this phenomenon from two perspectives:
(i). From the perspective of the codec or concealment algorithms

In the case of a loss at the beginning of voiced segment, as the previous frame is clearly an unvoiced frame or an unvoiced/voiced transition frame. The concealment algorithm will conceal the loss using the filter coefficients and the excitation for an unvoiced sound. It causes the lost frame to be concealed using the unvoiced features. In other words, during the unvoiced to voiced transition period, the shape of the vocal tract is in transition (not stable), and the LP filter coefficients will change rapidly for each frame. The excitation signal is also changing from unvoiced to voiced. The concealment algorithm can not conceal properly for the loss at this transition stage.

For a loss during the stationary part of a voiced segment, the concealment algorithm will conceal the current frame with the gain further reduced from the previous frame (adaptive codebook gain). The line spectral pair coefficients (or LP filter coefficients) of the last good frame are repeated. In other words, the vocal tract is at a stable stage (after the transition) and keeps the same shape. The LP filter coefficients are very stable during this stage. If the pitch delay does not change much within a short time period, a small loss can be concealed perfectly using the parameters of the previous frames. However, when there is an increase in burst loss size or frame size, it is difficult to conceal the losses adequately. The concealment performance degrades depending on the features in the voiced segments.

(ii). From the perspective of the perceptual quality measurement algorithms

The signal energy is very important for the overall perceived speech quality for all the perceptual algorithms. If a reference signal frame has a large signal energy (e.g. the beginning of a voiced segment), and the degraded signal has a very small energy (due to improper concealment), this will cause a significant increase in the perceptual distance. For a loss during the voiced segment, the degraded signal will normally have a rather large energy. Perceptual distance will vary for different loss size and loss location.

For different codecs (G.729, G.723.1 and AMR), the perceived speech quality shows large variations due to differences in the frame sizes. The perceptual distances using PSQM+ for the three codecs for a loss at the beginning of voiced segment 4 is summarized in Table 1 (including perceptual distances for no-loss cases).

Table 1: Perceptual distance using PSQM+

Codec Type	No-loss	1-frame	2-frame	3-frame	4-frame
G.729 (8 Kb/s)	1.36	1.62	1.83	2.11	2.42
G.723.1 (6.3 Kb/s)	1.51	1.79	2.84	3.54	4.03
AMR (12.2Kb/s)	0.98	1.35	1.6	2.06	2.45
AMR (4.75Kb/s)	1.92	2.17	2.42	2.81	3.34

From Table 1, it can be seen that the AMR (12.2 Kb/s) has the best perceptual quality and the AMR (4.75 Kb/s) the worst for no-loss cases. For a one-frame loss, the quality sequences remain the same. For a two-frame loss, the G.723.1 has the worst quality while AMR (12.2 Kb/s) remains the best. For three-frame and four-frame loss, G.729 and AMR (12.2 Kb/s) have similar perceptual quality, while G.723.1 remains the worst.

Of the three perceptual measurement methods (PSQM+, MNB and EMBSD), the PSQM+ provides perceptual distance values for most parts of the speech segment. The EMBSD and MNB only show the variations in perceived speech quality for frames with high energy. A loss at the unvoiced or voiced segments with small energy (see Figure 2) has no impact on perceived speech quality (flat line area in Figures 4 and 5). This is due to the different processing methods for silence and non-silence frames in the perceptual quality measurement algorithms. For EMBSD, the perceptual distance for an entire test speech segment is obtained by averaging over all non-silence frames (which are defined as the frames with the energy of the reference speech and the degraded speech both above their preset thresholds). For a loss at short and small energy voiced segments (e.g. voiced segment 1), the degraded speech with a loss has a limited energy. This is not taken into account by the EMBSD in the overall perceptual distance calculation and causes a flat area in Figure 5 (e.g. for voiced segments 1 and 3). A similar phenomenon exists for the MNB. The PSQM+ also classifies the frames as silence or non-silence. But it calculates all perceptual distances for silence or non-silence frames and uses different weighting factors for the overall perceptual distance calculation. Thus PSQM+ (Figure 3) also gives the perceptual distance value for a loss during small energy.

B. Convergence time with loss location

The second experiment was carried out to analyze the convergence time and its relationship to speech content or loss position. The convergence time is calculated by comparing the difference between the degraded signal without loss and the degraded signal with loss (as shown in Figure 1). First the MSE method [1] is used to calculate the convergence time for each loss position for a speech waveform such as that shown in Figure 2. Here the convergence time is defined as the first good frame received

after a burst of lost frames until the frame with its MSE value below a threshold (1% of the maximum MSE value seen so far). The convergence time for G.729 is shown in Figures 9, in units of frames (10ms/frame). From the figure, we can see that the convergence times are almost the same for different loss sizes. It shows a good linear relationship for loss at the voiced segments. It is at a maximum at the beginning of the voiced segments and decreases gradually to a minimum at the end of the voiced segments. The convergence time for a loss at the unvoiced segments appears stable. Similar results were also obtained for the AMR and G.723.1 codecs. It seems that the convergence time is only related to the speech content and not to codec and frame loss size.

We analyze further the convergence time based on perceptual distance. We measured the frame-based PSQM+ values between degraded speech without loss and degraded speech with loss. We choose two voiced segments in Figure 2. One with only voiced part (V(2) in Figure 2) and another one with the adjacent unvoiced part (V(4) in Figure 2). We change loss positions from the beginning to the end of the waveforms. The perceptual distance variation curves for selected loss positions are shown in Figure 10 and 11, in the unit of frames (here it is the frame of PSQM+ calculation, which is 32ms frame size with 50% overlapping resulting in 16 ms real frame size). Curves 1 to 5 (Figure 10) and 1 to 12 (Figure 11) correspond to the loss position from left to right. The loss position for each curve corresponds to the first non-zero point in the curve. The duration of the frames with non-zero (or over a threshold) perceptual distance is related to the convergence time.

From Figures 10 and 11, we can see that if a loss occurs during a voiced segment, then the convergence time is almost the remainder of the length of that voiced segment from the loss point (curve 1 to 5 in Figure 10 and curve 6 to 12 in Figure 11). The perceptual distance itself changes significantly with changes in the location of loss while the influence of the loss seems only limited to the voiced segment. The convergence times are almost the same as for a loss at unvoiced parts (curves 1 to 5 in Figure 11). The PSQM+ curves vary in a similar way. This explains the linear relationship of the convergence time during the voiced segments and flat variation during the unvoiced segments as shown in Figure 9. PSQM+ variation curves also show the overall PSQM+ values for the different loss position. We also tested other voiced segments and obtained similar results. The convergence time is more closely related to speech content and less affected by frame loss size and codec type. The convergence time is constrained by the duration of the voiced segments.

VI. CONCLUSIONS

We have investigated the impact of loss positions on perceived speech quality and the relationships between the convergence time and loss locations. Preliminary results show that a loss at unvoiced speech segment has almost no obvious impact on perceived speech quality. However, a loss at the beginning of voiced segments has the most severe impact on perceived speech quality. We have explained this effect from both the perspectives of the concealment and objective perceptual measurement algorithms. The impact of loss position on perceived speech or the concealment performance of three modern codecs (G.729, G.723.1 and AMR) have also been compared and analyzed. Three different perceptual speech quality measurement algorithms (PSQM+, MNB and EMBSD) are compared for the purpose of loss location analysis. We have analyzed the convergence times for different loss locations and different codecs by taking into account the normal MSE and perceptual PSQM+ measure. The results show that the convergence time is affected mainly by speech content (e.g. it is very stable within unvoiced segment whereas it varies but constrained by the duration of the voiced segments).

This work should help to fully understand the real impact of packet loss on perceived speech quality and the features of the convergence time in order to set the real loss constraint distance between the losses. This could be help for the development of a perceptually relevant packet loss metric, which could be valuable in non-intrusive VoIP measurements or to set up more efficient speech recovery systems.

Further research will focus on a more extensive analysis of the impact of packet loss on speech content.

ACKNOWLEDGEMENT

We are grateful to the Speech Processing Lab of the Electrical and Computer Engineering Department at Temple University, especially Dr. Wonho Yang and Prof. Robert Yantorno, for providing us with the Enhanced Modified Bark Spectral Distortion (EMBSD) software to evaluate the performance of concealment in this paper.

We are grateful to WWG/Acterna for sponsorship.

REFERENCES

[1] J. Rosenberg. G.729 Error Recovery for Internet Telephony. Project Report, Columbia University, May 1997

[2] R. Koodli and R. Ravikanth, One-way Loss Pattern Sample Metrics <draft-ietf-ippm-loss-pattern-03.txt>, Internet Draft, Internet Engineering Task Force, July 2000

[3] ITU-T Recommendation G.729, Coding of Speech at 8 kbit/s Using Conjugate-Structure Algebraic-Code-Excited Linear-Prediction (CS-ACELP), March 1996

[4] ITU-T Recommendation G.723.1, Dual Rate Speech Coder for Multimedia Communication Transmitting at 5.3 and 6.3 kbit/s, March 1996

[5] 3G TS 26.091, AMR Speech Codec; Error Concealment of Lost Frames

[6] H. Sanneck and N. Tuong Long Le, Speech Property-Based FEC for Internet Telephony Applications, Proceedings of the SPIE/ACM SIGMM Multimedia Computing and Networking Conference 2000, San Jose, CA, January 2000

[7] Q. Xie, S. Gupta, Error Tolerant RTP Payload Format for AMR <draft-xie-avt-et-rtp-amr-00.txt>, Internet Draft, Internet Engineering Tasks Force, October 2000

[8] ITU-T Recommendation P.861, Objective quality measurement of telephone-band (300-3400 Hz) speech codecs, February 1998

[9] ITU-T Contribution COM 12-20-E, Improvement of the P.861 Perceptual Speech Quality Measure, KPN Research, Netherlands, Dec. 1997

[10] S. Voran, Objective Estimation of Perceived Speech Quality – Part I: Development of the Measuring Normalizing Block Technique, IEEE Trans. on Speech and Audio Processing, Vol. 7, No.4. July 1999, pp. 371-382

[11] S. Voran, Objective Estimation of Perceived Speech Quality – Part II: Evaluation of the Measuring Normalizing Block Technique, IEEE Trans. on Speech and Audio Processing, Vol. 7, No.4. July 1999, pp. 383 -390

[12] W. Yang, Enhanced Modified Bark Spectral Distortion (EMBSD): An Objective Speech Quality Measurement Based on Audible Distortion and Cognition Model, Ph.D Dissertation, May 1999, Temple University, USA

[13] ITU-T Recommendation G.729 Annex B, A silence compression scheme for G.729 optimized for terminals conforming to Recommendation V.70, November 1996

[14] ETSI EN 301 704 V7.2.1 (2000-04), Digital cellular telecommunications system (Phase 2+); Adaptive Multi-Rate (AMR) speech transcoding (GSM 06.90 version 7.2.1 Release 1998)

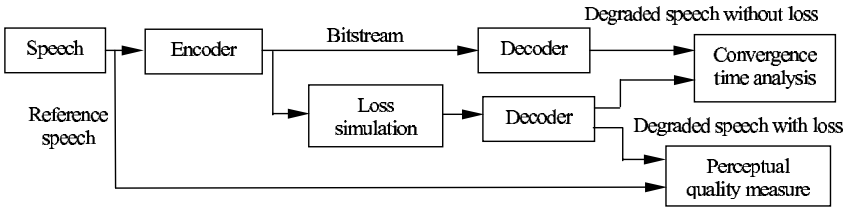


Figure 1: Structure of the simulation system

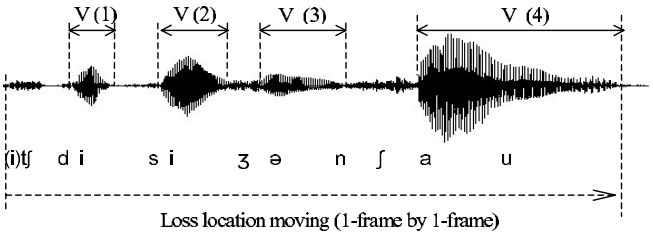


Figure 2: Speech waveform for the 1st talkspurt of test sentence
(The sentence is "_each decision show(s)_". V(1) to V(4) corresponds to 4 voiced segments)

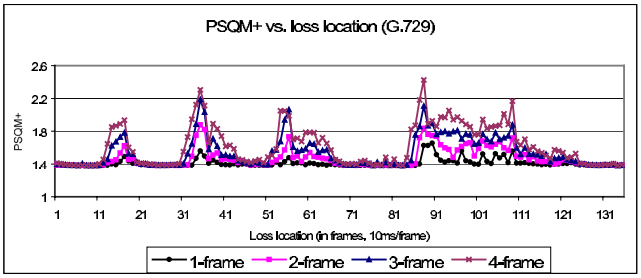


Figure 3: Overall PSQM+ values vs. loss location for G.729

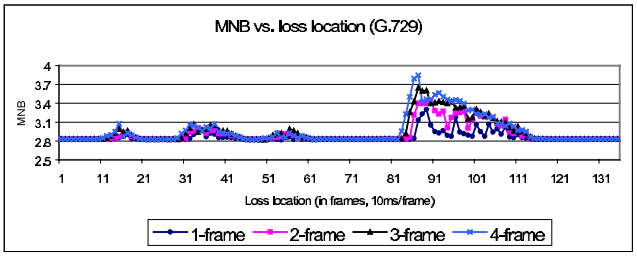


Figure 4: Overall MNB value vs. loss location for G.729

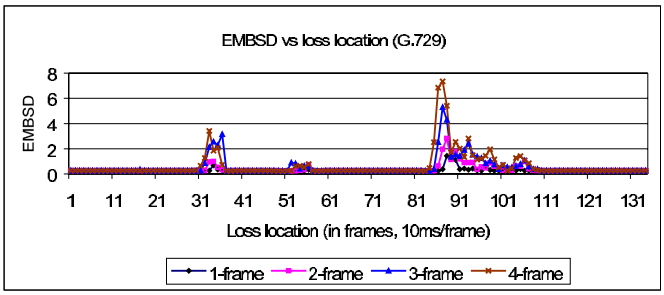


Figure 5: Overall EMBSD value vs. loss location for G.729

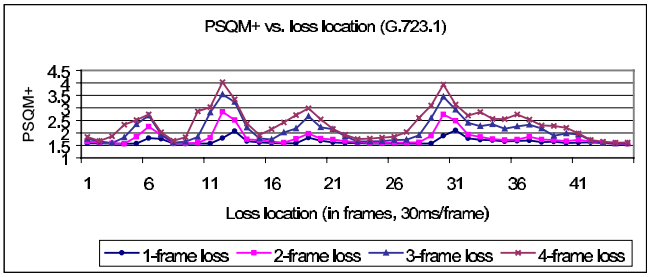


Figure 6: Overall PSQM+ value vs. loss location for G.723.1 (6.3 Kb/s)

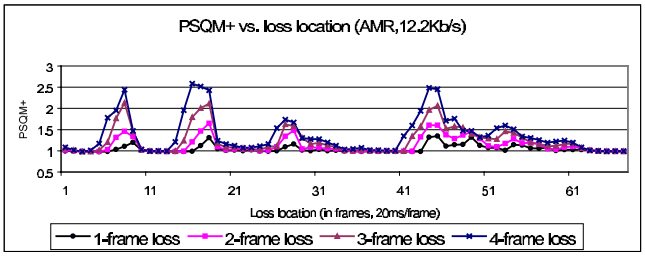


Figure 7: Overall PSQM+ value vs. loss location for AMR (12.2 Kb/s)

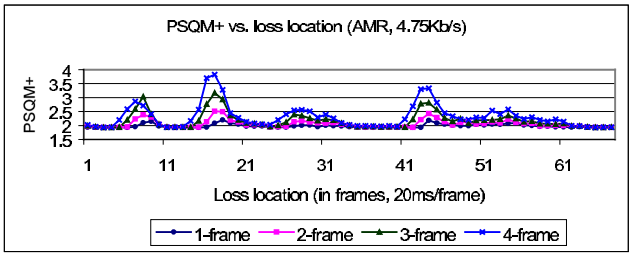


Figure 8: Overall PSQM+ values vs. loss location for AMR (4.75Kb/s)

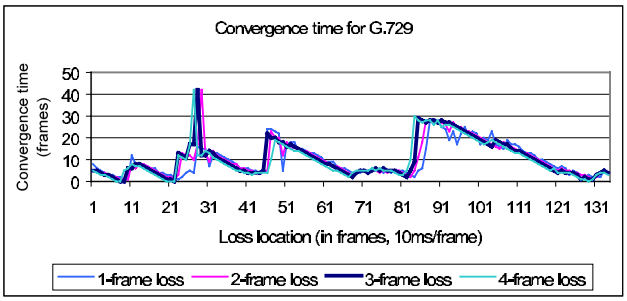


Figure 9: Convergence time vs. loss location for G.729

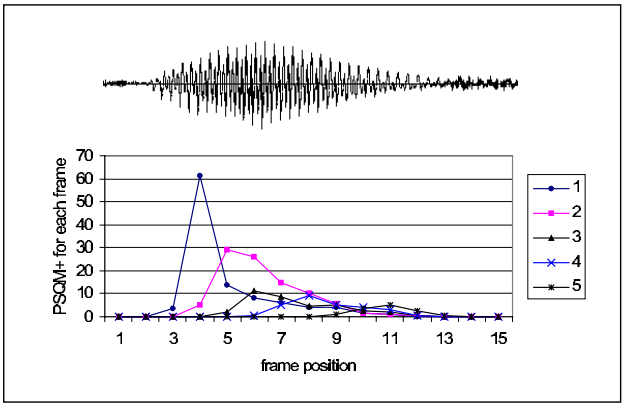


Figure 10: PSQM+ for voiced segment 2 (G.729, 2-frame loss)
(Curves 1 – 5 correspond to 5 loss locations from left to right)

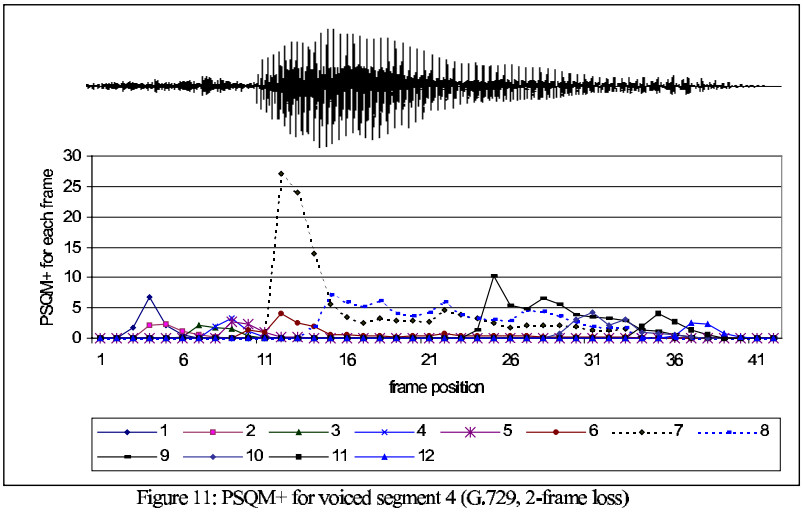


Figure 11: PSQM+ for voiced segment 4 (G.729, 2-frame loss)
(Curves 1 to 12 correspond to 12 loss locations from left to right)

Modeling the Effects of Burst Packet Loss and Recency on Subjective Voice Quality

A. D. Clark, Ph.D. Fellow IEE, Member IEEE

This paper describes VQmon, a non-intrusive monitoring technique for Voice over IP networks that is computationally efficient and suitable for integrating or embedding into VoIP gateways or IP Phones. This uses an extended version of the ITU G.107 E-Model incorporating the effects of time varying packet loss and recency . A 4 state Markov model is used to represent the time distribution of packet loss during a VoIP call.

QoS, Voice over Packet, E model, subjective quality

A. Introduction

Voice over IP networks differ from conventional telephone networks in that voice quality is affected by a wider variety of network impairments and can vary from call to call and even during a call. It is therefore desirable to monitor call quality in order that service providers can properly provision networks and that network resources are properly allocated.

Passive monitoring systems examine operating characteristics of a system in order to assess or measure performance level. This may involve examining elements of the system, for example buffer levels, or examining the data stream being transmitted through the system. This contrasts with Active measurement systems in which test data is inserted into the system and used to obtain performance measurements.

This paper describes a passive monitoring system (VQmon) for Voice over IP networks that is able to monitor per-call quality, providing feedback to a service management or CDR (Call Detail Record) system. The VQmon monitoring system also considers the effects of time varying impairments such as bursty packet loss and recency.

B. Embedded Passive Monitoring

Passive monitoring systems examine operating characteristics of a system in order to assess or measure performance level. This may involve examining elements of the system, for example buffer levels, or examining the data stream being transmitted through the system. This contrasts with Active measurement systems in which test data is inserted into the system and used to obtain performance measurements.

Telchemy Incorporated, 3360 Martins Farm Road, Suite 200, Suwanee, GA 30024, alan@telchemy.com

Embedded passive monitoring systems employ some form of monitoring function embedded into the equipment that comprises the system under test. This has the advantage of a closer relationship with system elements, allowing access to real time data and control information however has the disadvantage that implementation cost and complexity must be low. This contrasts with external passive monitoring systems which may, for example, be connected to T1 trunks or Ethernet LANs.

Within the context of a VoIP network embedded passive monitoring can be integrated into VoIP Gateways, IP Phones or other end-systems, providing access on a per-call basis to CODEC selection, packet loss and delay information. This permits per-call estimates of transmission quality to be made with minimal impact on the service being monitored.

Active monitoring systems typically make test calls through the VoIP network, transmit speech files and compare transmitted and received files using PSQM, PESQ or some similar method. This approach allows the CODEC performance to be directly measured however provides only a snapshot of network performance on a single connection.

C. Using the E Model to Estimate Voice Quality

The E Model [7] is a well established transmission quality model for telephone networks. This provides an objective method of assessing the mouth-to-ear transmission quality of a telephone connection and is intended to assist telecom service providers with network planning and performance monitoring. The E Model is described in some detail in ETSI Technical Report ETR 250 [7] and in ITU Recommendations G.107 [8] and G.108 [9].

The E Model has the following components:-

$$R = Ro - Is - Id - Ie + A$$

Which results in an R factor of between 0 and 100. The components of R are:-

Ro - representing the effects of noise and loudness ratio

Is - representing the effects of impairments occurring simultaneously with the speech signal

Id - representing the effects of impairments that are delayed with respect to the speech signal

Ie - representing the effects of equipment such as DCME or Voice over IP networks

A - the advantage factor, used to compensate for the allowance users make for poor quality when given some additional convenience (e.g. cellphone)

The equipment impairment factor Ie is generally used to represent the effects of Voice over IP equipment. Certain CODECs have been characterized through subjective testing to give a profile of the variation of Ie with packet loss [11].

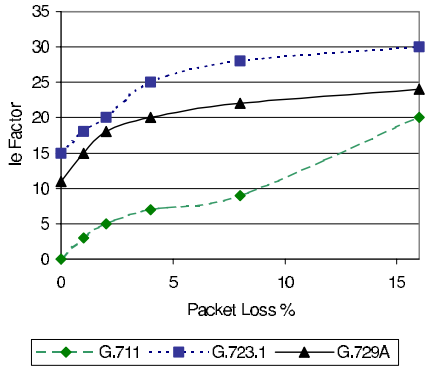


Figure 1. Mapping Packet Loss to Ie

D. TIME VARYING IMPAIRMENTS

The network impairment that has greatest effect on voice quality is packet loss. Packet loss may occur due to buffer overflow within the network, deliberate discard as a result of some congestion control scheme (e.g. Random Early Detection) or transmission errors. Several of the mechanisms that can lead to packet loss are of a transient nature and hence the resulting packet loss is bursty in nature. Bolot [6] studied the distribution of packet loss in the Internet and concluded that this could be represented by a Markovian loss model such as the Gilbert or Elliott models.

Jitter (or packet delay variation) also has an effect however the use of a jitter buffer generally replaces jitter by delay and packet loss. Incoming packets are buffered and then read out at a constant rate; if packets are excessively late in arriving then they are discarded. For this reason it is advisable to measure packet loss (or rather frame loss) between the jitter buffer and the CODEC. Jitter buffers are often adaptive and adjust their depth dynamically based on either the current packet discard rate or current jitter level.

Cox and Perkins [3] compared the impact of random and burst packet loss on G.711 and G.729A CODECs. They found that for low packet loss rates a burst distribution gave a higher subjective quality than a non-bursty distribution whereas for high packet loss rates the converse was true. One explanation for this effect is that at low packet loss rates the distortion due to the loss of two successive packets is not much greater than that of one lost packet and is counteracted by the greater distance between packet loss events.

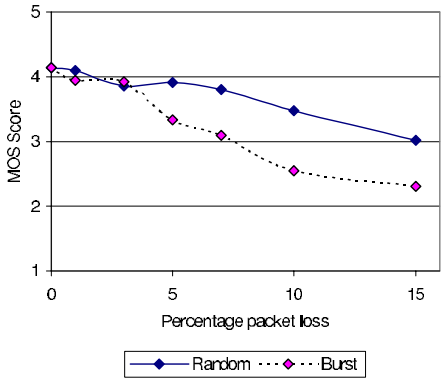


Figure 2. Effects of Random vs Burst Packet Loss

If the rate of packet loss varies during a VoIP call then the perceived call quality will also vary. The term instantaneous quality may be used to denote the measured or calculated quality due to packet loss or other impairments and the term perceived quality may be used to denote the quality that the user would report at some instant in time.

Intuitively, if instantaneous quality changes from good to bad at some moment in time then the listener would not immediately notice the change. As

time progresses the user would become progressively more annoyed or distracted by the impairment. This leads to the idea that the perceived quality changes more slowly than instantaneous quality.

In tests reported by Barriac et al [1] the packet loss rate during a 3 minute call was varied from 0 to 25%. In the example shown below the packet loss was set to 25% for most of the call and reduced to 0% for a 30 second period mid-call. Listeners were asked to move a slider to indicate their assessment of quality during the call and then asked to rate the call at the end. This showed the effect described above, with an approximately exponential curve with a time constant of 5 seconds for the good-to-bad transition and 15 seconds for the bad-to-good transition.

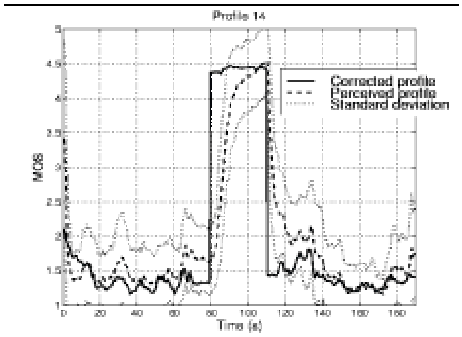


Figure 3. Relationship between Instantaneous and Perceived Quality Metrics (Source Barriac [1])

The recency effect reflects the way that a listener would remember call quality.

In tests conducted by AT&T [2] a 15 second burst of noise was moved from the beginning to the end of a 60 second call. When the noise was at the start of the call users reported a MOS score of 3.82 whereas when the noise was at the end of the call users reported a MOS score of 3.18, giving a change in MOS score of 0.64.

Tests reported by France Telecom [1] showed a similar effect. An improvement in MOS score of 0.68 was reported when a period of high packet loss was moved from the end to the beginning of a 60 second call.

The effect is believed to be due to the tendency for people to remember the most recent events [4] or possibly due to auditory memory which typically decays over a 30 second interval [1].

E. Extending the E Model to reflect time varying impairments

In many Voice over IP network implementations the connection between CODEC and telephone handset may be transient. For example a user may be dialing through an existing local loop and being routed to a Gateway located at the Central Office. This means that some elements of the E Model may not be measurable by equipment located within the network. Default values for many of the E Model parameters can be assumed (per G.107), giving an effective value for Ro Is of 94.

The E Model can then be represented as:

$$R = 94 \quad Id - Ie$$

The average value of Ie may be determined by taking the average of the perceived quality for the call.

For each time interval t(i), the instantaneous quality $I_{inst}(i)$ is determined by measuring the post-jitter buffer packet loss for the time interval and mapping the packet loss to an Ie value using the curves shown in Figure 1.

The perceived quality can be estimated from the instantaneous quality by assuming an exponential decay, modeling the effect described in Section D. A time constant of 5 seconds is assumed for a deterioration in quality and 15 seconds for an improvement in quality.

Over a series of N samples the average perceived quality is therefore

$$I_e = \text{sum}(I_{perceived}(i))/N$$

The recency effect can be modeled by assuming that perceived quality decays exponentially with time constant t3 from the exit value I_{exit} from a burst of noise or distortion towards the average Ie. The following model is proposed:

$$I_e \text{ (end of call)} = Ie + (k(I_{exit} - I_e)) e^{-y/t3}$$

F. Modeling packet loss

In order to meet the requirements of real time implementation within a VoIP Gateway it is essential to minimize processing overhead. The approach used in VQmon is to obtain some minimal amount of

information during a call and perform most computation at the call end.

A 4-state Markov model is used to represent the burst packet loss characteristics of the call. The four states represent the conditions of receiving or losing a packet within burst or gap conditions.

State 1 Gap state- receive packet
State 2 Burst state receive packet
State 3 Burst state lose packet
State 4 Gap state receive packet

A gap state is defined by the requirement that g_{min} successive packets must be received.

This model is similar to the more normal Gilbert or Elliott models however includes a state representing the loss of an isolated packet within a gap. The rationale for this is that packet loss concealment (e.g. replay last packet), can mask the effects of isolated lost packets.

A packet loss event driven model is used to count a minimum number of key transition events. It is assumed that Voice Activity Detection is being used and hence that packet loss reports relate to packets containing speech energy. When the call is completed then remaining transition counts can be derived and then the counts normalized to give probabilities. This model holds considerable information and can be used to determine average gap and burst size and density, successive lost packet distribution etc.

Packet loss event:-

$c_5 = c_5 + pkt$
if pkt >= g_{min} *then*
  if lost = 1 then
    $c_{14} = c_{14} + 1$
  else
    $c_{13} = c_{13} + 1$
  lost = 1
  $c_{11} = c_{11} + pkt$
else
  lost = lost + 1
  if lost > 8 then $c_5 = 0$
  if pkt = 0 then
    $c_{33} = c_{33} + 1$
  else
    $c_{23} = c_{23} + 1$
    $c_{22} = c_{22} + pkt$
pkt = 0

pkt is an input parameter representing the numbr of packets received since the last lost packet event.The series of counters c_{11} to c_{14} are used to determine the corresponding Markov model transition probabilities (i.e c_{11} is used to calculate p_{11}). Counter c_5 is used to measure the delay since the last significant burst of lost packets. Parameter g_{min}, the minimum gap size, is typically 16.

The equipment impairment value for the burst and gap condition is determined using the curves shown in Figure 1, giving I_{eb} and I_{eg} respectively.

Let I_1 be the quality level at the change from burst condition I_{eb} to gap condition I_{eg} and let I_2 be the quality level at the change from I_{eg} to I_{eb}

$$I_1 = I_{eb} - (I_{eg} - I_2)\, e^{-b/t1} \quad \text{where } t_1 \text{ is typically 5}$$

$$I_2 = I_{eg} + (I_1 - I_{eg})\, e^{-g/t2} \quad \text{where t2 is typically 15}$$

Combining these gives

$$I_2 = (I_{eg} (1 - e^{-g/t2}) + I_{eb} (1 - e^{-b/t1}) e^{-g/t2}) / (1 - e^{-b/t1-g/t2})$$

Integrating the expressions for I_1 and I_2 to give a time average gives

$$I_e(av) = (b\, I_{eb} + g\, I_{eg} - t_1 (I_{eb} - I_2) (1 - e^{-b/t1}) + t_2 (I_1 - I_{eg}) (1 - e^{-g/t2})) / (b + g)$$

This may be used to determine an R factor from the expression:-

$$R = 94 - I_e(av)$$

This R factor does not yet include the effects of delay or recency however is useful when examining the effects of packet loss, jitter and CODEC type on transmission quality. Within the context of VQmon this is the *Network R Factor*.

The effects of delay are well known [5] and easily modeled. Delays of less than 175mS have a small effect on conversational difficulty whereas delays over 175mS have a larger effect. A simple delay model is used in VQmon:

If delay < 175 mS then
  $I_d = 4$. delay
Else
  $I_d = 4 + (\text{delay} \quad 175) / 9$

The Recency effect is modeled using the approach described in E above. It is assumed that the I_1 represents the exit value from the last significant burst of packet

loss, y represents the time delay since the last burst, t3 is a time constant of typically 30-60 seconds and k is a constant (set to a nominal value of 0.7).

$$I_e \text{ (end of call)} = I_e(av) + (k(I_1 - I_e(av))) e^{-y/t3}$$

The *User R Factor* is determined from the expression below. This is intended to more closely approximate the user s perspective of quality and therefore does take into account both recency and delay.

$$\text{User R Factor} = 94 - I_e \text{ (end of call)} \quad I_d.$$

G. EXPERIMENTAL RESULTS

Some initial subjective comparison was made to validate the VQmon model. An audio file was corrupted using a burst error process which comprised a low loss state and a high loss state, the loss and state transition probabilities being selected randomly. A 10mS packet size was used and packet loss concealment applied.

Sets of five test files were created, and a group of six listeners used to rank the files from 1(best) to 5 (worst). The ranking was compared with that predicted by the algorithm described above.

File	Mean user rank	R factor rank
Vgnf	1.0	1
dgcs	3.0	2
cnkb	3.5	3
mxhr	2.5	4
gwav	5.0	5

Table 1 Comparison of R factor and User Ranking data set 1

File	Mean user rank	R factor rank
Ecen	1.2	1
Fhpc	1.8	2
Rlgd	3.2	3
Xknc	3.8	4
Dlwx	5.0	5

Table 2 Comparison of R factor and User Ranking data set 2

File	Mean user rank	R factor rank
Mvui	1.8	1
Fyok	1.2	2
Rkdi	3.5	3
Mtwt	3.5	4
okdu	5.0	5

Table 3 Comparison of R factor and User Ranking data set 3

The results showed reasonable correlation with user ranking however there were several obvious exceptions, for example file mxhr from data set 1. The locations of packet loss events for this file were reviewed and it became apparent that some loss bursts occurred either during silence periods or during periods when the sound produced by the speaker was not changing significantly, for example during an extended aaaah .

Further comparisons are being made using both ranking tests of the type described above and comparisons with well known objective test measures such as PSQM and PESQ.

H. SUMMARY AND CONCLUSIONS

VQmon represents a novel approach to embedded passive monitoring that incorporates the effects of burst packet loss and recency. The algorithm provides a computationally efficient method for estimating the transmission quality of a Voice over IP network, and produces results that correlate well with user ranking of impaired files.

One problem that arises as a result of analyzing only packet related statistics is that it is not possible to identify the exact effects of lost packets during talkspurts. Additional work is in process to incorporate smart packet loss events which would be generated by the CODEC.

I. REFERENCES

[1] France Telecom Study of the relationship between instantaneous and overall subjective speech quality for time-varying quality speech sequences: influence of a recency effect. ITU Study Group 12 Contribution D.139, May 2000

[2] Rosenbluth, J. H., Testing the Quality of Connections having Time Varying Impairments. Committee contribution T1A1.7/98-031

[3] Cox, R., Perkins, R., Results of a Subjective Listening Test for G.711 with Frame Erasure Concealment, Committee contribution T1A1.7/99-016, May 1999

[4] Baddeley, Human Memory

[5] Britt, R., Armstrong, M., Voice Quality Recommendations for IP Telephony. Committee contribution TR41.1.2/00-05-004, May 2000

[6] Bolot, J., Fosse-Parisis, S., Towsley, D., Adaptive FEC based Error Control for Interactive Audio in the Internet. Infocom 99

[7] ETSI Speech Communication Quality for Mouth to Ear for 3.1 kHz Handset Telephony across Networks. Technical Report ETR250, 1996

[8] ITU Recommendation G.107

[9] ITU Recommendation G.108

Multiparty Sessions

Scalable Floor Control in Conferencing Environments: The RBone Approach

Dirk Trossen
Nokia Research
5 Wayside Road, Burlington, MA 01803

Abstract- **Most features of conferencing applications are mostly independent from specific scenarios. Thus, it is useful to provide a generic service to accelerate and simplify the development of such applications. Scalability in terms of group size and distribution of conference members is a big issue in the design of such services especially when applying conferencing in large scaled Internet scenarios. Beside functionality for conference management and multipoint communication, floor control is a crucial issue for the provision of application state synchronization and controlled access to application resources. We present an approach to provide a generic floor control service even suited for large scaled environments. The proposal uses efficient multicast on local level combined with a tree-based routing on global level by introducing the Resource Backbone (RBone) approach, which promises to improve the responsiveness of the provided floor control. The service as well as the protocol mechanisms required for implementation are presented in this paper.**

Index terms - **floor control, resource backbone**

A. Introduction

Interactive collaborative scenarios like remote meetings, virtual classrooms, or sharing applications via the Internet have become more and more popular in the past ten years. From an application's point of view, the need for *tight control*, such as for synchronization of actions or controlled access to application resources, arises for these applications in contrast to loosely coupled scenarios like plain video streaming. Hence, coordination and synchronization means are required due to concurrent activities in these scenarios. Thus, it is crucial for these scenarios to provide means for the implementation of *floor control* [13], e.g., to map the real-life's *social protocol* [13] onto the distributed environment.

A crucial issue in the development of a floor control service is its *scalability* with respect to the responsiveness of the provided functionality depending on the number of participating users and their geographical distribution. Recently proposed conferencing toolkits and standards suffer especially from this key issue. Either unicast-based topologies, often simple stars, are used routing the requests to a centralized floor control manager or a multicast-based exchange of floor holder information is applied often leading to large response time of each request due to the overhead to ensure reliability of the exchanged floor information.

In this paper, a scalable floor control approach is presented combining both mechanisms. It starts with a presentation of the provided services. The main part of the paper is dealing with outlining the proposed protocol mechanisms in detail.

The basic idea of the proposal is to combine the usage of multicast and unicast by applying efficient multicast on local level and apply a tree-based unicast routing on global level. For that, similar to the MBone approach, multicast-capable local islands are interconnected using unicast *tunnels*. Within this tunnel topology, named as the *resource backbone* (RBone) throughout the paper, a floor control specific routing is used for optimization. Hence, an optimized interconnection of the distributed entities is achieved.

Thus, this approach is expected to improve the responsiveness of the provided service even in conferences of larger scale. This is especially true when considering scenarios in which the group of conference members is comprised of a few subgroups of participants each located in a fast local area network as for instance in internal corporate meetings among geographically distributed developer groups. However, a performance evaluation or measurements of real-life implementations are is not shown in this paper.

The remainder of the paper is organized as follows. Section B gives an overview of related work in the area of group communication toolkits and protocols. Section C outlines the provided floor control services, while in Section D the RBone approach is introduced for realizing the services. Finally, Section E concludes the paper, and Section F gives an outlook for future work.

B. Related Work

Since the importance of group communication has been increased significantly during the past ten years, there are several conferencing environments being proposed for the implementation of collaborative applications.

While environments proposed in [1][2] focus on specific aspects of conferencing functionality, more generic platforms like the Scalable Conferencing Control Service (SCCS [17]) or standard-based solutions of the ITU (*International Telecommunications Union*) or the IETF (*Internet Engineering Task Force*) aim to provide a wide spectrum of services for the creation of conferencing applications.

For that, the ITU specifies a set of standards (T.120 [11]) providing multipoint transfer, conference management, and floor control functionality for data applications. A tree-based approach of interconnected service providers is used to which applications are attached. Due to the centralized approach used for the floor control functionality, this approach leads to a bad scalability in terms of responsiveness as shown in [10].

In [17], the scalable conferencing control service (SCCS) is proposed with an ITU-compliant service model using more sophisticated protocol mechanisms to improve the scalability of the environment even in large scaled scenarios. For that, a resource management scheme is introduced leading to a higher responsiveness of the system when locally handling requests in the tree of providers. However, the proposed mechanism does not use underlying multicast facilities which can be seen as the major drawback similar to the centralized version of the ITU.

e-mail: dirk.trossen@nokia.com phone: +1 (781) 993 3605

The proposed *Internet Multimedia Conferencing Architecture* of the IETF [7] (see Figure 1) outlines the components and protocols to be used for realization of conferencing scenarios in the Internet.

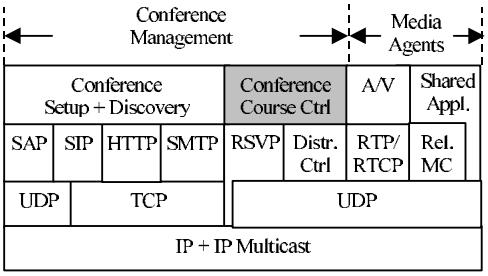


Figure 1: Internet Multimedia Conferencing Architecture

For the realization of the *conference course control* component for the control of tightly coupled conferences services like the ITU T.120 protocol stack or SCCS might be used. For the support of loosely coupled conferences (or *light-weight sessions* [6]) following the ALF (*application level framing* [6]) approach, conference control is realized by loosely interconnected participants using multicast-based information exchange with a lack of centralized control of membership and floor holder information. However, these approaches have to deal with large response times in large scaled conferences due to the distributed handling of the request, e.g., using quorum-based approaches [15].

An approach to provide floor control in the Internet as an extension to existing MBone tools is proposed by Malpani and Rowe in [14] using a centralized architecture. Since the approach only maintains a speakers list in a conference, the large response time of the system is not a critical issue. However, this lack of efficiency is not acceptable for generic floor control services.

Dommel [3] proposes group coordination in a larger scope including floor control for application state synchronization. The proposal uses a shared tree for multicast delivery, sub-group support, and floor control message routing. This approach extends the usual IP multicast routing by proposing a sub-group addressing. However, a single floor controller approach is used for the floor control protocol which applies a centralized approach for which the shared tree routing is used. Furthermore, the multicast tree routing has to be extended using the proposed sub-group addressing.

Hence, it can be summarized that the related work either implement tree- or simple star-based approaches by interconnecting conference members, or use multicast-based scheme by allowing temporary inconsistencies, or do not implement floor control at all. In the following two sections, a floor control protocol is presented using a combination of local multicast and global unicast.

C. PROVIDED SERVICES

Compared to the approaches presented in the related work section, the remainder of the paper will focus on the provision of floor control services. Hence, other conference course control functionality (see Figure 1), such as membership control, is not within the scope of the paper.

As proposed in [17], a floor control service should provide facilities to support application state synchronization and controlled access to application resources. Hence, the service shall enable to map social protocols, i.e., the rules to access application objects like audiovisual streams, onto distributed systems. The list of possible scenarios includes conducted meetings or even more complicated mediated conferences, but also access control on resources as for shared applications. However, the mapping of floors onto application semantics is not within the scope of the proposed service.

Each floor is identified using a conference-unique name. The naming pattern is not within the scope of the service. However, it is recommended to use a decimal naming scheme to simplify naming conflict resolving. The following floor control services are provided:

- *grab floor*: allocates a floor for exclusive use by the requesting participant
- *inhibit floor*: allocates a floor for non-exclusive use by several participants
- *release floor*: releases an allocated floor; changes the state of the floor accordingly
- *test floor*: asks for the current state (F_FREE, F_GRABBED, F_INHIBITED) of the floor
- *ask floor*: asks the current floor holder to grant an exclusive floor to the requesting entity
- *give floor*: grants an exclusive floor to another participant
- *holders of floor*: asks for a list of current floor holders

It can be seen that the provided floor control service is very similar to the T.122 [12] of the H.323 standard. However, requesting the current floor holders is not supported by the T.122 standard. Additionally, appropriate repairing mechanisms are not provided to recover from node or network failures.

D. SERVICE REALIZATION: THE RBONE APPROACH

In the following sections, the protocol environment of the service is presented together with a description of maintenance functions for the topology. Furthermore, it is outlined in detail how to handle the floor control service requests in this environment.

1st. Protocol Environment

Figure 2 shows the protocol environment in terms of the used topology to realize the proposed services. It can be seen that the topology is comprised of the conference participants which must join the *conference management group* (CMG). This group can also be used for other conference management service, e.g., to exchange membership information.

Furthermore, selected participants, the *RB providers*, are interconnected within a tree topology, the *Resource Backbone* (RBone or RB), with a dedicated top node. Each RB provider is responsible for handling floor control requests within its own local *floor control island* (FCI). This FCI is a local multicast group containing all joined conference participants within local scope.

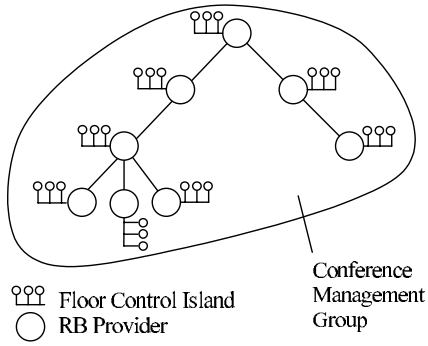


Figure 2: Protocol Environment

Hence, the RBone tree topology connects multicast-capable islands using unicast *tunnels* similar to the MBone approach [5]. However, a floor control specific routing is used within the unicast topology.

It is not within the scope of the approach how to define the different multicast group addresses. However, means like the *session description protocol* (SDP [9]) together with the *session announcement protocol* (SAP [8]) might be used to distribute the appropriate information.

It will be outlined in the following sections how the proposed scheme applies multicast-based floor control on local level and tree-based routing on global level. But first, the used transfer mechanisms of underlying transport layers are depicted.

2nd. Encoding and Transfer of Messages

In the following protocol description of the floor control service, a simple message transfer is applied. For that, it is assumed to use an underlying multicast transport service. In addition, dedicated participants, namely the RB providers, are interconnected using an underlying unicast transport service. Both services are assumed to provide reliable, consistent delivery of data units called *messages*. The encoding of these messages is not within the scope of the paper. However, the presented protocol description easily enables to extract a message format for the exchanged messages.

Reliability is bounded by the fact that member end systems may find that they no longer can reliably interact with the other members, e.g., due to network partitioning. For the unicast case, a *connection failure indication* is mandatory to be delivered to the participant. Messages are *globally ordered*. Thus, each message is assigned a message number by the appropriate transport service, and messages are delivered to participants in monotonic message number order. In the rest of this document, the term *distribute* will be used to indicate that a member end system sends a message using the appropriate transport service.

3rd. Floor Context

Each FCI member maintains state information of floors valid for the local FCI. This state information includes the name and state of locally allocated floors and the name of local holders. Additionally, each FCI member maintains a list of current FCI members. If a floor is indicated as free using a FLOOR_STATUS message, the appropriate floor entry is deleted from the floor context.

The top RB provider maintains an additional global floor context containing all local floor context information. In addition, each floor entry contains the information in which branch of the tree this floor entry is valid.

4th. Joining the Floor Control Island

Each participant wishing to use the floor control services must join the multicast group representing the FCI and sends an FCI_JOIN message to the FCI. For that, it is assumed that the underlying multicast protocol supports the establishment of locally scoped groups.

After sending that message, the *FCI general* timer is set to two seconds assuming a fairly small response time due to the local character of the FCI. If the newly joined participant receives an FCI_THERE before the FCI general timer expires, the join procedure is finished and the newly joined member becomes a normal FCI member. The FCI_THERE message contains the local floor context. All FCI members store the received member information of the new participant.

If the timer expires before receiving the FCI_THERE, it is assumed that either the old RB provider failed or the newly joined member is the first FCI member. In both cases, the newly joined member becomes the new RB provider and sends an FCI_INQUIRY to the FCI. Furthermore, the *heartbeat mechanism* (see Section D.6) is started. Each member of the FCI has to respond with an FCI_REGISTER containing its presence information and the local floor context within the time interval defined by FCI general. If the old RB provider is still working, it has to release all RB connections and has to respond to the FCI_INQUIRY, if needed.

After getting the local floor context and FCI member list, the RB provider continues with the RB backbone establishment procedure (see Section D.5).

5th. Establishing and Extending the Resource Backbone

An RB provider connects to the resource backbone by sending an RB_JOIN to the CMG containing its presence information. Note that this message is not sent by the initiator of the conference since this participant does not need to connect to the resource backbone as the first member of the conference. If the RB provider is connecting to the RB for the first time, an appropriate flag of the RB_JOIN message is set for indication.

Any other RB provider responds by sending an RB_THERE message to the CMG. This response is delayed when there is

another RB extension operation pending. Hence, any response is delayed until an RB_FINISH message is received at the CMG.

The RB_THERE message contains the responding RB provider's presence information. This send operation is delayed randomly to avoid message explosion. When other RB providers receive this message, they stop sending their own response because a responding RB provider was found.

All responding RB providers try to establish a unicast transport connection to the requesting RB provider. From the set of responding RB providers, only one connection request is positively confirmed while all other transport connection requests are refused. After connection establishment, the requesting RB provider sends an RB_JOIN message via the unicast connection, i.e., via the RBone topology. This message is routed upward in the tree until a) it reaches the top RB provider or b) it reaches the requesting RB provider again. Note that this message is only routed upward on connections which are valid. This means that the establishment procedure is finished for this connection. Furthermore, it can be seen that the established RB connection is an upward connection from the requesting provider's point of view.

In case a), the RB is established successfully and the completion of the procedure is signaled (see below). In case b), a loop was built. As a result, the requesting RB provider releases its connection again and becomes the new top RB provider. To collect valid floor state information, the new RB top provider sends an RB_GET_CONTEXT message downwards. This message is sent until it reaches a leaf node, which sends an RB_CONTEXT message back in upward direction. These messages are cumulated in each branching node until it reaches the new top RB provider with updated global floor context information.

The completion of the RB extension procedure is signaled by the current top RB provider sending an RB_FINISH to the CMG.

6th. Heartbeat

As an indication that a local RB provider is still alive, a heartbeat mechanism is used. For that, the current RB provider regularly sends an FCI_HEARTBEAT message to the FCI. The interval for sending the heartbeat is defined by the FCI general value. Since the FCI general interval determines the responsiveness of the system against RB provider failures, this interval is kept small to recover fast enough from an RB provider failure.

A failure of the RB provider is detected by missing FCI_HEARTBEAT messages. For this, a detection timer is used which is set to twice the value of the FCI general interval by default.

7th. Repairing the Resource Backbone

Repairing an existing resource backbone is necessary in three cases, namely when an RB connection fails, or a local RB provider quits, or a local RB provider fails. In the following, the mechanisms for all these cases are presented.

1) RB connection fails

In this case, it is assumed that both connection endpoints are still intact. Hence, both endpoints start the RB extension procedure (see Section D.5) again to find new RB endpoints.

2) Local RB Provider quits

In this case, the local RB provider quits orderly. This is done by sending an FCI_GIVE to the FCI containing the list of remaining FCI members and the presence information of any FCI member indicating the new RB provider. This member must take over the role of the new RB provider by sending an FCI_GIVEN message to the FCI. If the chosen member does not respond within FCI general seconds, a failure is assumed, the chosen member is removed from the list, and the selection process is restarted. If there is no member left on the list, the old RB provider deletes the FCI.

If the selection of a new RB provider was successful, the old RB provider must release all RB connections, and the new RB provider starts the RB extension process (see Section D.5) to find an RBone endpoint.

3) Local RB Provider fails

As indicated in Section D.6, an RB provider failure is detected by missing FCI_HEARTBEAT messages. In that case, the oldest remaining FCI member must sent an FCI_THERE message containing the current list of FCI members. If this message is not received within FCI general seconds, the next member on the list must send the message, and so forth. All members not sending the FCI_THERE message are deleted from the list. If they are still working, they are supposed to join the FCI again. If the old RB provider is still working, it has to release all RB connections, and it has to join the FCI again for usage of floor control services, if needed.

The newly selected local RB provider connect to the RB using the procedure of Section D.5.

8th. Service Request Handling

The basic rule for handling floor control service requests is that each FCI tries to respond to a request locally. If this is not possible due to missing information, the request is sent upward in the RB for further processing.

This general rule is explained in more detail for each floor control service request in this section. The mechanisms are depicted using an indented bullet notation for better illustrating the protocol functionality.

4) Grab Floor

The requesting participant sends a FLOOR_GRAB message to the FCI. The RB provider checks its floor context whether the floor is already grabbed locally.

- If yes, a FLOOR_ERROR message is sent to the FCI with error code E_GRABBED. Note that this case can be avoided by the requesting participant by checking its own floor context before sending the message.
- If not, the FLOOR_GRAB message is routed upward in the RB.

- If an RB provider is passed whose local FCI contains the current floor holder, a FLOOR_ERROR message is sent back to the originating RB provider immediately with error code E_GRABBED or E_INHIBITED, respectively.
- If the message reaches the top RB provider, the global floor context is checked.
  - If the floor status is F_FREE, the global floor context is changed, and a FLOOR_STATUS message is sent back via the RB indicating the new status F_GRABBED. The FLOOR_STATUS message is relayed to the local FCI by the originating RB provider. Each FCI member updates its floor context appropriately.
  - If the floor is allocated, a FLOOR_ERROR message is sent back to the originating RB provider with error code E_GRABBED or E_INHIBITED. This message is relayed to the FCI.

5) Inhibit Floor

The requesting participant sends a FLOOR_INHIBIT message to the FCI. The RB provider checks its floor context whether the floor is already grabbed, i.e., allocated exclusively.

- If yes, a FLOOR_ERROR message with error code E_GRABBED is sent to the FCI indicating the erroneous message.
- If not, the RB provider checks whether the floor is inhibited locally.
  - If yes, the RB provider updates its floor context and sends a FLOOR_STATUS message to the FCI indicating the new floor context entry to the other members.
  - If not, the FLOOR_INHIBIT message is routed upward in the RB.
    - If the message passes an RB provider whose local floor context indicates the floor as grabbed, a FLOOR_ERROR message with code E_GRABBED is sent back to the originator.
    - If the message passes an RB provider whose local floor context indicates the floor as F_INHIBITED, an appropriate FLOOR_STATUS message is sent back to the originating RB provider.
    - If the message reaches the top RB provider, the global floor context is checked.
      - If the floor status is F_FREE or F_INHIBITED, the global floor context is changed accordingly, and a FLOOR_STATUS message is sent back via the RB indicating the current status. The FLOOR_STATUS message is relayed to the local FCI by the RB provider. Each FCI member updates its floor context appropriately.
      - If the floor status is F_GRABBED, a FLOOR_ERROR message with error code E_GRABBED is sent back to the originating RB provider being relayed on the FCI for indication.

6) Release Floor

The requesting participant sends a FLOOR_RELEASE message to the FCI. The RB provider checks its floor context whether the floor is either inhibited or grabbed locally.

- If not, the message is ignored since a floor is released which has not been allocated before.
- If yes, the RB provider must consider the following cases:
  - *grabbed floor*: The FLOOR_RELEASE message is sent upward via the resource backbone to indicate the status change to the top RB provider which updates the global floor context and sends a FLOOR_STATUS message back downward the RB indicating the new status. The local RB provider updates its local floor context and relays the FLOOR_STATUS message to the FCI indicating the status update to the FCI members.
  - *inhibited floor*: the requesting FCI member is deleted from the local floor holder list in the floor context (if not a floor holder, the message is ignored), the status is changed accordingly, and the new context is indicated by sending a FLOOR_STATUS to the FCI.
    - If the requesting FCI member was the last local holder of the floor, the FLOOR_RELEASE message is sent upward via the RB.
      - If the FLOOR_RELEASE message passes an RB provider whose local FCI still has at least one holder of that floor, the message is not routed upwards anymore because there is no status change necessary to be indicated.
      - If the message reaches the top RB provider and there is no other branch in the tree containing floor holders, the top RB provider changes the status in its global floor context accordingly.

7) Test Floor

The requesting participant first checks its own floor context information for getting the local floor information.

- If there is no floor context entry, it sends a FLOOR_TEST message to the FCI. The RB provider forwards this message upward in the RB.
  - If the message passes an RB provider with sufficient information, i.e., a valid floor context entry, it generates a FLOOR_STATUS message to be routed downwards via the RB. This RB provider is finally the top RB provider holding the global floor context information.

8) Ask Floor

The requesting participant sends a FLOOR_ASK message to the FCI. The RB provider checks its floor context information and must handle three cases.

- If the floor is inhibited locally, the RB provider sends a FLOOR_ERROR message with error code E_INHIBITED to the FCI.
- If the floor is grabbed locally, there is nothing to do since the floor holder received the FLOOR_ASK message, too.
- If there is no floor entry in the local context, the RB provider forwards the FLOOR_ASK message upwards in the RB.
  - If the message passes an RB provider with sufficient information, this RB provider relays the message to its local FCI when the floor is grabbed.
  - If the floor is indicated as inhibited in a passed RB provider's floor context, a FLOOR_ERROR message with error code E_INHIBITED is sent back via the RB downwards.
  - If the message reaches the top RB provider, the global floor context is checked.
    - If the floor status is F_GRABBED, a FLOOR_ASK message is forwarded downward the RB on the appropriate branch of the floor holders.
    - If the floor is either inhibited or free, a FLOOR_ERROR message with error code E_INHIBITED or E_FREE is sent back to the originating RB provider which relays this message to the FCI.

It can be seen that there is no response for a FLOOR_ASK message. However, this message is usually supplemented by an appropriate floor passing operation of the application.

9) Give Floor

The giving participant sends a FLOOR_GIVE message to the FCI. If the giving participant is not the floor holder or the floor is indicated as being inhibited, the message is ignored by both the RB provider and the FCI members. If the giving participant is the current floor holder, two cases must be considered.

- If the given participant is a local FCI member, all FCI members, including the RB provider, change their local floor context information indicating the given participant as the new floor holder.
- If the given participant is not a local FCI member, the RB provider forwards the FLOOR_GIVE message upwards via the RB, sets the local floor status entry to F_GIVING, and indicates the temporary floor context entry to the FCI by sending a FLOOR_STATUS message.
  - If the FLOOR_GIVE message reaches the top RB provider, the message is forwarded downwards via the appropriate branch of the RB.
  - If the message reaches the top RB provider on the same branch on which the floor holder should reside and the floor holder is not in the FCI of the top RB provider, a FLOOR_STATUS message is sent back to the originating RB provider to be relayed on the FCI indicating the old floor holder as the new one. Hence, the old status is re-established.
  - If during forwarding the message either upwards or downwards the RB provider is reached whose FCI contains the given participant, an appropriate FLOOR_STATUS message is relayed to the FCI and forwarding is stopped. Furthermore, a FLOOR_GIVEN message is sent back by the receiving RB provider in the reverse direction.
    - If the FLOOR_GIVEN message passes the top RB provider, the RBone branch information is changed in the global floor context (storing the old and new one).
    - If the FLOOR_GIVEN message is received by the originating RB provider, a FLOOR_STATUS message is sent to the local FCI to indicate the free status of the floor.

10) Holders of Floor

The requesting participant sends a FLOOR_HOLDER message to the FCI. Three cases have to be considered.

- If the floor is grabbed locally, the current floor owner responds by sending a FLOOR_HOLDER_LIST message with its own presence information. This information exchange can be avoided by checking the local floor context information in the requesting participant.
- If the floor is grabbed in another FCI, the RB provider forwards the FLOOR_HOLDER message upwards via the RB.
  - If the message reaches the top RB provider, it is forwarded downwards on the appropriate branch.
  - If the message passes the RB provider whose local FCI contains the current floor holder, this provider generates a FLOOR_HOLDER_LIST message containing the presence information of the current floor holder and sends it back in the reverse direction.
    - If the FLOOR_HOLDER_LIST message reaches the top RB provider, the message is forwarded downwards on the appropriate branch.
    - If the message reaches the RB provider of the requesting participant, the message is relayed on the FCI.
- If the floor is inhibited, the FLOOR_HOLDER message is forwarded upwards by the RB provider via the RB.
  - If the message reaches the top RB provider, a FLOOR_HOLDER_ASK message with an empty floor holder list is sent downwards on all branches in which the floor is marked as allocated.
  - These messages are forwarded in all RB providers on all connected branches until they reach a leaf node. This leaf node inserts its local floor holder information, if available, and sends the message back in upward direction.

- Each branching node cumulates the FLOOR_HOLDER_ASK messages on all branches into one message to be forwarded until a single message reaches the top RB provider.
- An appropriate FLOOR_HOLDER_LIST message is sent back an the appropriate branch of the RB in downward direction.
- If the message reaches the RB provider of the requesting participant, the message is relayed on the FCI.

11) Handling during RB Repair operations

If an RB repair procedure (see Section D.7) is started during forwarding any floor control message, forwarding is halted until the repair procedure is finished, signaled by RB_FINISH.

After finishing the repair procedure, forwarding messages is restarted at the originating RB provider. All other forwarding messages in the RB providers are discarded.

If the originating RB provider failed or quit, the successor of this RB provider has to restart the forwarding procedure. Thus, each local FCI member has to be aware of pending operations. This is feasible due to the multicast-based state information change.

For the floor give case, the originating RB provider re-sends the FLOOR_GIVE message with the temporary floor entry (state F_GIVING). The following cases must be considered:

- If the FLOOR_GIVE message reaches the RB provider whose FCI contains the new floor holder, the procedure is continued as in the original case depending on the current floor context entry.
- If the FLOOR_GIVE message reaches the top RB provider, the message is forwarded on the old branch using the appropriate entry in the global floor context. Further forwarding operations are handled the same way.

As it was stated in the introduction, this section extensively outlined the protocol mechanisms to be used to realize the proposed floor control services following the RBone approach. Although the explicit message notations are not presented, their extraction from the presented description should be easily feasible.

E. CONCLUSIONS

This paper presented a floor control approach which offers sophisticated services for the implementation of distributed application state synchronization and application resource access.

The main focus in the development of the underlying mechanisms was on providing a scalable solution in terms of participating users in the conference to ensure high responsiveness of the services. For that, the proposal applied the idea of establishing a *resource backbone* (RBone) topology interconnecting multicast-capable floor control islands similar to the MBone approach in the Internet. The RBone is comprised of a tree of selected entities in the conference being responsible for routing floor control requests outside the local island.

The paper depicted the mechanisms for maintaining the tree in terms of establishing and updating the topology in node and network failure cases. Furthermore, the service request handling was presented which follows the idea to handle requests locally, if possible, and to forward the requests globally, if needed.

Hence, the proposal applies efficient multicast transfer on local level and tree-based unicast routing on global level using a floor control specific routing scheme. Furthermore, recovery from network and node failures, both on local and global level, is supported to some extent as well using a heartbeat mechanism in the FCIs.

This approach is expected to improve the responsiveness of the provided floor control service even in conferences of larger scale, especially if the conference is comprised of a few subgroups each located in a fast local area network.

F. FUTURE WORK

There are several issues to be addressed in the future work. The first one is the proposal for a *naming scheme* for the floors. Currently, the naming of the floors is not within the scope of the protocol. However, to avoid conflicting floor names, a unique naming scheme might be desirable to be added to the service. This could easily be done by using a simple numeric identifier naming scheme similar to the ITU standards.

Secondly, the cases of failures in the protocol, specifically in the RBone, have to be studied more extensively to improve the robustness of the protocol.

Thirdly, the scalability and robustness of the protocol has to be studied in form of simulations or protocol prototyping.

G. REFERENCES

[1] M. Altenhofen, J. Dittrich, R. Hammerschmidt, T. Käppner, C. Kruschel, A. Kückes, T. Steinig, "The BERKOM multimedia collaboration service" in Proc. of ACM Multimedia, pp. 457-463, August 1993

[2] I. Beier, H. Koenig, "GCSVA - A Multiparty Videoconferencing System with Distributed Group and QoS Management" in Proc. of IEEE International Conference on Computer Communication and Networks (IC3N'98), pp. 594-598, October 1998

[3] C. Bormann, J. Ott, D. Kutscher, D. Trossen, "Simple Conference Control Protocol – Service Specification", Internet Draft, Work in Progress, February 2001

[4] H.-P. Dommel, J.J. Garcia-Luna-Aceves, "Group Coordination Support for Synchronous Internet Collaboration" in IEEE Internet Computing Magazine vol. 3 No. 2, April 1999

[5] H. Eriksson, "MBONE: The multicast backbone" in Communications ACM, vol. 37, no. 8, pp. 54-60, August 1994

[6] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing" in ACM Computer Communication Review, vol. 25, no. 4, pp. 342-356, October 1995

[7] M. Handley, J. Crowcroft, C. Bormann, J. Ott, "The Internet Multimedia Conferencing Architecture", Work in Progress, Nov. 2000

[8] M. Handley, C. Perkins, E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000

[9] M. Handley, V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998

[10] T. Helbig, D. Trossen, " The ITU T.120 Standard Family as Basis for Conferencing Applications" in Proc. of SPIE International Symposium Voice, Video, & Data Communications, pp. 190-201, November 1997

[11] ITU-T, "Data Protocols for Multimedia Conferencing", ITU-T Recommendation T.120, 1998

[12] ITU-T, "Multipoint Communication Service - Service Definition", ITU-T Recommendation T.122, 1998

[13] A. McKinlay, R. Procter, O. Masting, R. Woodburn, "Studies of turn-taking in computer-mediated communications" in Interacting with Computers, vol. 6 no. 2, pp. 151-171, June 1994

[14] R. Malpani, L. A. Rowe, "Floor control for large-scale MBone seminars" in Proc. of ACM Multimedia, November 1997

[15] L. Y. Ong, M. Schwartz, "Centralized and Distributed Control for Multimedia Conferencing" in Proc. of IEEE International Conference on Communications, pp. 197-201, June 1993

[16] D. Sisalem, H. Schulzrinne, "The Multimedia Internet Terminal" in Journal on Telecommunication Systems, Vol. 9 No. 3, pp. 423-444, 1998

[17] D. Trossen, "Scalable Conferencing Support for Tightly-Coupled Environments: Services, Mechanisms, and Implementation Design" in Proc. of IEEE International Conference on Communications, pp. 889-893, June 2000

An Example of Using Presence and Availability in an Enterprise for Spontaneous, Multiparty, Multimedia Communications

Hyong Sop Shim, Chit Chung, Michael Long, Gardner Patton and Siddhartha Dalal
Applied Research, Telcordia Technologies New Jersey, USA
{hyongsop, chit, mlong, gcp, sid}@research.telcordia.com

Abstract In today s distributed, team-oriented work environment, it is critical that team members should be able to conduct multiparty, multimedia conferencing spontaneously. Spontaneous conferencing is analogous to ad-hoc, drop-in meetings and impromptu discussions in multi-person offices. The knowledge of the presence and availability of team members can be effectively used to facilitate spontaneous conferencing. In this paper, we present SEC, a communications system designed to support spontaneous, ad-hoc conferencing in an enterprise. SEC makes an extensive and innovative use of the Session Initiation Protocol (SIP) for presence and availability management, conference control, and text messaging. We also discuss in detail the design of SEC and report on the initial usage experience with SEC services.

Index Terms Presence and Availability, B uddy List, Voice over IP, Spontaneous Multimedia Conferences

A. INTRODUCTION

Critical to increasing the productivity of group work in today s team-oriented, distributed workplaces is the ability of group members to communicate with each other in an efficient manner. Group members frequently engage in spontaneous, multiparty communications. However, as group members become geographically distributed, ad-hoc, drop-in meetings and impromptu discussions in multi-person offices are often infeasible. One result is that group members often waste time and energy in scheduling efforts playing phone tag.

One effective approach to address this issue is to enable team members to see the presence and availability state of each other in real time. This way, team members know when to initiate new communications and when to invite other members to ongoing communications.

Existing commercial instant messaging applications, such as AOL IM and MSN Messenger, enable a group of users to communicate based on the presence and availability state of each other. However, most of these systems are designed for public use and thus lack certain features that are critical for enterprise use. For example, in most commercial instant messaging applications, the support for voice communications is limited to one-to-one. Most systems that do support voice conferencing require advanced scheduling and thus do not effectively support the spontaneity of enterprise group communications.

In this paper, we present Spontaneous Enterprise Communications (SEC). SEC is designed to support spontaneous, ad-hoc conferencing in an enterprise. In particular, SEC enables team members to subscribe to the presence and availability state of each other and to conduct conferencing with those who are available on the fly. SEC provides both text and voice conferencing capabilities and enables conference participants to switch to using different media on demand. For a voice conference, SEC enables users to use the VoIP capability in the SEC client application, PSTN phones, or both, depending on their current communication capabilities.

SEC makes an extensive and innovative use of Session Initiation Protocol (SIP) [1] for presence and availability management, conference control, and text messaging. In this paper, we discuss in detail the design of SEC and report on the initial usage experience with SEC services.

The rest of the paper is organized as follows. Section B discusses related work. Section C defines terms and concepts critical to discussing SEC services. Section D describes functional requirements for SEC services. Section E presents SEC services from the perspective of end users. Section F discusses architectural issues in providing SEC services and presents an architectural overview. Section G describes in detail how SEC services are provided. Section H reports on the initial usage experience with SEC services. Section I discusses future work and concludes the paper.

B. RELATED WORK

Using presence and availability for group communications in distributed enterprise workplaces has been extensively studied and found to be effective [7], [8], [9], [10], [11], [12]. However, its use has mostly been for text-based communications. SEC extends the use of presence and availability in enterprise group communications by supporting both text and voice communications in an effective manner.

Since the emergence of Mbone [14] and its tools for multimedia communications [6], enabling multimedia conferencing services on the Internet has received a tremendous attention. Beginning with [5], efforts to enable multimedia communications on Mbone have generated an important body of work that is in wide use today for VoIP, e.g., Real Time Protocol (RTP) [3] and Session Description Protocol (SDP) [2]. SEC is an example of providing a particular application-level service, i.e., spontaneous conferencing, using these underlying technologies.

There are many commercial products that provide multimedia communication capabilities, e.g., CU-SeeMe [15] and Microsoft NetMeeting [24]. However, in most of these systems, the communication is limited to one-to-one. Those that do allow multiparty conferencing for both PSTN and/or VoIP users, e.g., Microsoft Exchange Conference Server [25] and WebEx [27], require conferences to be scheduled in advance (usually via a Web interface) and thus do not address the spontaneity of enterprise group communications in an

efficient manner. ME.net [26] is an example of commercial services that provide spontaneous conferencing capabilities using buddy lists and Web pages. However, ME.net only supports PSTN phones and does not allow users to switch to a different communication media type on demand.

C. TERMINOLOGY

In SEC, *presence* is defined as data that indicates whether or not a user has logged into a communications system, and *availability* refers to the user s willingness to communicate with other users in the system. The presence and availability state of a user is called the *PA state* of the user. *PAL* stands for presence and availability list. A PAL is equivalent to a buddy list. Each item on a PAL identifies a contact or buddy whose current PA state the owner of the PAL is interested in knowing. A PAL provides the main interface through which users access SEC conferencing services. We distinguish PALs from standard buddy lists in that a contact on a PAL may not only represent a human but also any object that is an event source in SEC. PAL is part of an ongoing work on using PA subscriptions and notifications in an distributed enterprise environment.

D. REQUIREMENTS

In this section, we describe the functional requirements on SEC. SEC is primarily designed to provide spontaneous conferencing in an enterprise. Hence, SEC should:

- Integrate with a PAL. Users should be able to initiate new conferences from their PALs. Users should also be able to add new participants to ongoing conferences from their PALs.
- Integrate with enterprise directory. An enterprise directory is the main source of contact information in an enterprise. Thus users should be able to manage their PALs using the enterprise directory, e.g., add a new contact to a PAL and/or add a new participant to an ongoing conference from a directory search result.
- Support PSTN phones. Today, PSTN phones are the main means of communication in an enterprise. For mobile users, cellular phones are still the only viable option for interactive voice communications. Therefore, while providing VoIP capabilities, SEC should also enable users to use their PSTN phones.
- Allow mobility. Today s workspace often spans multiple geographical locations, e.g., a different corporate location and on the road. At the same time, one often needs to get in touch with his or her project members, managers, or customers from a remote location. Therefore, SEC should allow users to access its services from different locations. In addition, the communication devices available to users may change as they move from location to location. Thus SEC should enable users to use different communication devices.
- Provide multiple types of communications media. Different work contexts require different types of communications media. For example, for a sparse exchange of short ideas over a long period of time, text may be better suited than voice. For long, detailed discussions and presentations, voice may be a better choice. Hence, SEC should support multiple types of communications media. Currently, SEC supports text and voice. Support for other media types, e.g., images and video, and application sharing is also planned.
- Enable dynamic change of communications media type. The work context in which a communication takes place may dynamically change. For example, a group of users who have been asynchronously exchanging ideas in a text conference may wish to have a more interactive discussion and thus wish to switch to a voice conference adding a voice component to the text conference. Therefore, SEC should allow users to switch or add a different communications media type on demand.
- Hide communications devices. Users do not need to know the communication devices of those that they communicate with. Rather, users only need to specify the identities of the parties with whom they wish to communicate and the type of media, e.g., text or voice. SEC will then establish appropriate connections between the communication devices of choice.
- Be easy to use. SEC should streamline the process through which end users access its services as much as possible. In particular, creating a new conference and inviting a new participant to ongoing conferences should be simple without the need to schedule and reserve communication resources.

Another important consideration is security. Different work contexts and corporate cultures may have different Security requirements. In fact, such requirements may range from no security to protection from all possible attacks. We are currently designing and developing a flexible security framework to be used in SEC. Issues related to providing secure communications in an enterprise environment is beyond the scope of this paper

.

E. USAGE EXAMPLE

In this section, we describe in detail SEC services from the perspective of end users. Figure 1 shows how an enterprise user creates his or her PAL from enterprise directory search results. In Figure 1, the window on the left is SEC s interface for accessing an enterprise directory, called DQ. Using this window, the user can submit a query to DQ in order to search for the contact information of other employees. Once the search results are returned and displayed in the window, the user can then select one or more entries in the window and click on the Add to Contact List button. This results in the selected users being added to the user s PAL. The current version of SEC does not place any restrictions on who can add whom on their PALs. However, in practice, different policies may be required, depending on corporate security policies, organizational structures, and cultures. Part of our future

research efforts is to design a flexible policy framework that can be adapted to different enterprise workplaces.

In Figure 1, the window on the right is the main client interface for accessing SEC services. Using this window, the user can call a PSTN phone. In addition, the user sees the PA state changes of each contact on his or her PAL in real time. In SEC, the PA state of a contact may currently assume one of the following values: AVAILABLE, BUSY, and OFFLINE. When the PA state is AVAILABLE, the contact is willing to accept invitations to communicate. When the PA state is BUSY, the contact is not willing to accept invitations to communicate. When the PA state is OFFLINE, the contact is not presently logged into SEC. After logging in, the user can manually set his or her PA state via the SEC main client window.

Users can communicate only with the AVAILABLE contacts on their PALs. Using their PALs, users can instantly create new text or voice conferences or invite AVAILABLE contacts to ongoing conferences on demand.

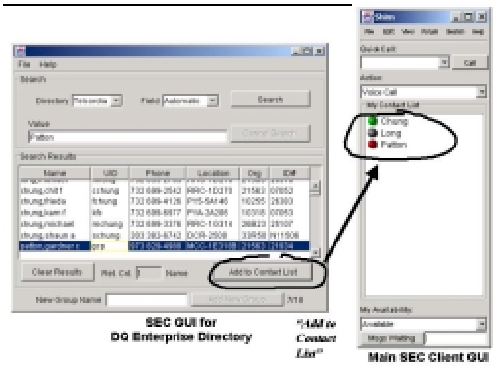


Figure 1: PAL Management with DQ Corporate Directory

Note that the PA state of a contact does not include the contact s presence and availability on his or her specific communication device(s). This is in contrast with some current proposals for presence management [16], [19], which allow users to subscribe to and get notifications of the PA state changes on the individual communication devices of their contacts. We argue that it is easier for end users to manage their subscriptions to the PA state of other users and to control access to their own PA state at the user level than to do so at the device level. Providing a PA service at the user level still enables end users to have control over their communication devices. One approach would be to allow users to have profiles, and the system would automatically make a decision about which device(s) to use, depending on which profile is active when the user makes or receives a call. In the current implementation of SEC, each user makes a decision about which device to use on a call-by-call basis (see Section G.4).

Also note that the PA state of a contact could further be refined to include the type of communication the contact can and/or is willing to participate in, i.e., text, voice, or both. In the current implementation, SEC assumes that a user is always capable of and is willing to participate in both voice and text communications.

In Figure 2, the window on the left is the interface for a voice conference, and the window on the right is the interface for a text conference. A voice conference has a tabbed interface because users generally participate in one voice communication at a time. A text conference has its own window because users tend to participate in multiple text communications simultaneously.

In both voice and text conferences, the user can add new participants to an ongoing conference by first selecting the corresponding entries on the user s PAL and then clicking on the Add button on the respective windows. Subsequently, the invited contacts are alerted and are allowed to accept or reject the user s invitation. As part of accepting the invitation, the invited contacts decide on the communication devices to be used in the conference. Currently, for a voice conference, they can use either the VoIP capabilities of the SEC client application or a PSTN phone. Once they accept the invitation, they automatically join the conference and immediately start communicating with the other participants. In the case of a voice conference, a participant may also invite a participant by dialing a phone number from within the conference. When the called party answers the phone, s(he) is automatically placed in the conference through SEC and can immediately start communicating with the other conference participants.

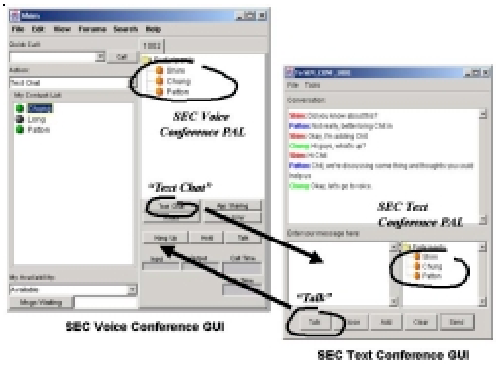


Figure 2: SEC GUI for Voice and Text Conferences

As shown in Figure 2, the windows for voice and text conferences show a participant list. This list is dynamically updated as the conference membership changes. In SEC, the conference participant list is provided as an extension to its PAL service. That is, when participants join a conference, they automatically subscribe to the PA state of the conference. Part of the PA state of the conference is its membership, and whenever the conference membership changes, SEC proactively notifies the conference participants of the new

membership. This is an example of subscribing to the PA state of a non-human entity and is an effective means for conference control and management. For example, SEC could allow an authorized user to subscribe to the PA state of a conference without requiring the user to first join the conference. Then, the user could monitor the conference membership changes with minimal overhead.

Figure 2 also shows that a conference of one media type can spawn a new conference of a different media type. Hence, the participants in a text conference may dynamically switch to using voice by clicking on the Talk button on the text conference window, and vice versa. This ability to switch to using a new media type seamlessly, and on demand, enables conference participants to adapt to the changing context in which their communication takes place and thus increases the effectiveness of their group work.

The SEC client application shown in Figure 1 and Figure 2 is designed for desktop computers. SEC also has a Web client designed for networked Pocket PCs or PDAs running Windows CE, e.g., Compaq iPAQ. The SEC Web client enables mobile users to access and manage their PALs and participate in voice and text conferences. A mobile user who uses the SEC Web client participates in a voice conference using his or her cellular phone. Figure 3 shows the main interfaces of the SEC Web client as displayed on a Pocket PC emulator running Packet Internet Explorer for Windows CE.

Figure 3: SEC Web Client

To support mobile users who do not have a networked Pocket PC or PDA, SEC could provide a voice interface to its services. For example, the live addressbook system [23] allows users to dial an 800 number, set their PA state, discover the PA state of their contacts, and call a contact. Through such a dial-in interface, SEC can allow those mobile users without networked PDAs or Pocket PCs to register their current phone numbers and automatically manage their PA state. For instance, when such a user receives or makes a call to a contact using SEC, SEC can notify the subscribers to the PA state of the user that the user is busy. Likewise, when the user hangs up the call, SEC can notify the subscribers that the user is now available. In this capacity, SEC would be playing the (limited) role of a client proxy for the user.

F. ARCHITECTURAL OVERVIEW

In SEC, the concept of a *SEC client* is separated from the concept of a *communication device*. The SEC client is capable of speaking to SEC server components in order to access SEC services. A communication device generates and renders media, i.e., voice and text. This separation significantly increases the flexibility and availability of access to SEC services. For example, an office user who prefers the desktop phone for voice communications or whose desktop PC is not multimedia-capable, can fully utilize SEC services by running the SEC Java client application on the desktop PC and participating in voice conferences using the desktop phone. Likewise, a mobile user who has a networked PDA running Windows CE and a cell phone can run the SEC Web client on the PDA for SEC signaling and text communications and use the cell phone for voice communications.

SEC uses a centralized conference control mechanism, in which a central coordinator creates and assigns globally unique addresses, i.e., URLs, to conferences on demand. For membership control, SEC uses a SIP signaling to dial-out to potential conference participants. In a sense, SEC employs the model of a dial-out bridge [18] but adapts the model in order to support spontaneous conferencing. See Section G.4.

Another approach would have been to distribute the administrative tasks related to conference control among SEC clients. However, in general, managing group membership is more difficult in a distributed architecture than doing so in a centralized architecture [21]. Especially for tightly coupled conferences, in which all the participants have to agree on the full membership at all times, the group membership, in effect, can only change in lock step [13]. Therefore, a fully distributed conference control scheme may not be effective in a dynamic environment, such as a team-oriented workplace, where conferences are created in an ad-hoc manner, and the conference membership may need to change on demand and on the fly. The disadvantage of a centralized conference control scheme is that the central controller represents a single point of failure and that the scalability is limited. In a client-server architecture, server components are generally built with a high degree of fault tolerance and scalability, e.g., using multiple replicas of the same component. However, it remains a research issue to design a centralize conference control mechanism that effectively enables spontaneous conferencing in a fault-tolerant and scalable manner.

In a SEC conference, the participating clients connect to a SEC conference bridge, and the media stream from each client is routed through this bridge to the other clients. That is, a SEC conference has a star configuration. Another approach is a fully meshed configuration, in which each client is directly connected with all the other clients in the conference, and all the media streams are directly sent from client to client. In general, without support for IP multicasting in the network, the fully-meshed configuration demands a higher network bandwidth than the star configuration. For a conference with n number of participants, the fully meshed configuration requires $n(n-1)$ network connections, whereas the star configuration requires $2n$ network connections. See Figure 4.

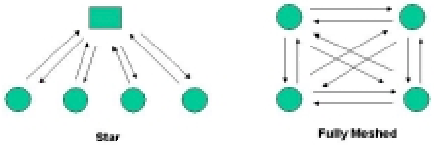


Figure 4: Example Conference Configurations

However, as in a centralized conference controller, the central bridge in the star configuration represents a single point of failure. Furthermore, the star configuration introduces a longer delay in the end-to-end delivery of voice packets than the fully meshed configuration. In SEC, because a client is connected to a conference bridge when it joins a conference, different bridges may be dynamically allocated to handle different conferences. In our initial experience with the prototype implementation of SEC, we have not experienced significant delay in conferences on our corporate intranet that spans two corporate sites, Morristown and Piscataway in New Jersey. However, a study is needed to evaluate the performance of SEC in a more general environment.

In SEC, a voice conference bridge does not mix voice streams in the middle for two reasons. First, because the mixing is a CPU intensive operation, it limits the scalability of the bridge. Second, it limits control over simultaneous voice streams that end points may require to provide application-specific services. For example, different users may wish to focus on different speakers. Providing this kind of service is infeasible with a mixing bridge.

Therefore, in SEC, the voice conference bridge simply routes voice packets to their destinations and leaves the task of mixing to SEC clients. For a PSTN phone, the bridge routes voice packets to a PSTN proxy that performs mixing and then sends the mixed stream to the phone. This approach takes advantage of the increasing availability of multimedia-enabled PCs with high processing power. Distributing the mixing task to clients helps the voice bridge support a larger number of simultaneous conferences than otherwise possible. This approach also allows the voice bridge to control network resource usage by limiting the number of voice packet streams for a given voice conference that it routes out to the clients.

Figure 5 shows the architectural overview of SEC. As previously described, SEC is a distributed client-server system. The server components include Communications Controller (CC), PAL Manager/Registrar, Multipoint Control Unit (MCU), Multipoint Text Control Unit (MTCU), PSTN Gateway Proxy, and HTTP Proxy. The CC, PAL Manager/Registrar, and HTTP Proxy form the main interface through which SEC clients access SEC services. The MCU, MTCU, PSTN Gateway Proxy, and HTTP Proxy are responsible for routing media streams between communicating clients.

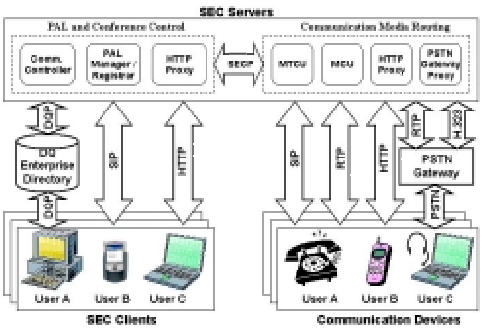


Figure 5: SEC Architectural Overview

The CC is responsible for setting up conferences and maintaining the current contact addresses of registered users. The CC also interfaces with the integrated enterprise directory.

As its name implies, the PAL Manager maintains PALs in the system. It keeps track of who is subscribing to whose PA data and who is participating in which conferences. The PAL Manager is also a system registrar; that is, a SEC client logs into the SEC system by registering with the PAL Manager. Upon successful registration, a SEC client learns from the PAL Manager the contact address of the CC to be used for its session.

The MTCU is responsible for routing text messages to appropriate SEC clients. It maintains the participant membership of each ongoing text conference in the system. A SEC client learns the contact address of the MTCU to be used for its conference(s) at conference setup time (see Section G.4).

The MCU is the SEC voice conference bridge and as previously discussed, is mainly responsible for routing voice packet streams to appropriate SEC clients. Like the MTCU, it maintains the participant membership of each ongoing voice conference in the system. A SEC client learns the contact address of the MTCU to be used for its conference(s) at conference setup time (see Section G.4).

The PSTN Gateway Proxy facilitates the participation of users using PSTN phones in multiparty SEC voice conferences. Without the PSTN Gateway Proxy, a user using a PSTN phone would be limited to 2-party voice calls. Given any PSTN phone used in a SEC voice conference, the PSTN Gateway Proxy receives the voice packet streams of the other clients in the conference from the MCU handling the conference, mixes the received streams, and sends the mixed stream to the PSTN gateway that is connected to the phone. In the reverse direction, the PSTN Gateway Proxy receives the voice packet stream from the phone via the same PSTN gateway and sends the stream to the same MCU, which in turn sends it to the other clients in the conference.

The HTTP Proxy allows users to access the SEC services using Web browsers. The HTTP Proxy receives user commands in the form of HTTP requests and transforms them into SEC operations before sending them to SEC servers.

Likewise, the HTTP Proxy receives the results of these operations from SEC servers, transforms them into HTTP responses, and then sends these responses to the user. The HTTP Proxy is designed to support mobile users who have a networked PDA running Pocket Internet Explorer. The HTTP Proxy enables such a user to enter a (cell) phone number where he or she can be reached. Subsequently, the user can initiate and participate in a SEC voice conference using the phone. The user can also initiate and participate in a SEC text conference via the HTTP Proxy.

The SEC servers and clients communicate with each other using a variety of Internet protocols for PAL management, conference control, and media transport. Specifically, SEC makes extensive use of the Session Initiation Protocol (SIP) [1] and its proposed extensions [16], [17] to register users, manage PALs, set up conferences, and transport text messages. SIP is chosen mainly because of its simplicity and flexibility. The basic SIP consists of a small number of methods and allows for incorporating application-specific semantics into its methods. In addition, SIP messages are encoded in plain text. This greatly helps streamline the testing and debugging process during implementation.

Note that the SEC server components are NOT SIP proxies. In fact, from SIP s perspective, they can be viewed as SIP end points. Also note that the PAL Manager works as a registrar in SEC only. It is not currently meant to function as a general-purpose SIP registrar.

The Real-time Transport Protocol (RTP) is used to transport voice packets between the communication devices used in voice conferences. H.323 [4] is currently used to communicate to PSTN gateways to support PSTN phones or IP phones not capable of mixing voice streams.

G. DESIGN

In this section, we discuss in detail the integration with a corporate directory, PAL management, and conference control in SEC.

1) Integration with Corporate Directory

One of the main goals in SEC is to enable enterprise employees to use their existing employee identifiers for accessing SEC services. Hence, SEC currently interfaces with a directory system, called DQ. DQ is a directory lookup service that has been developed and used in-house. The SEC client enables the user to look up other users via DQ and add search results as contacts to his or her PAL (see Figure 1). The SEC client retrieves the employee identifiers from DQ search results when adding new contacts (see Section G.3). The SEC client also allows the user to place a phone call from DQ search results.

2) Registration

When the user logs in, the user s SEC client sends a SIP REGISTER message to the PAL Manager/Registrar. In SEC, this REGISTER message contains the user s contact addresses for conference invitations (SIP INVITE), new subscription notifications (SIP SUBSCRIBE [16]), notifications of the PA state of the user s contacts (SIP NOTIFY [16]), and incoming text messages (SIP MESSAGE [17]). Note that after with registration, the user always receives SIP INVITE messages whenever the user is being invited to voice conferences. This is the case even when the user specifies use of a PSTN phone, in which case the user dynamically redirects the invitation to the phone. While this design incurs extra messaging overhead in conference control, it allows users to have a fine-grained control over how many voice conferences they are willing to participate in simultaneously and the devices to be used.

Another benefit of this approach is that it allows the user to see the full membership of the conference to which the user is being invited, even when the user uses a PSTN phone in conjunction with a PC or PDA. The conference membership data is mostly unavailable on the standard Caller-Id service today.

3) PAL Management

As discussed in Section E, SEC provides PAL management at the level of individual users, rather than individual devices. To illustrate, say User A wishes to subscribe to User B. First, User A looks up User B in DQ and then instructs the SEC client to add User B to his or her PAL. At this point, the SEC client retrieves the employee identifier of User B from the DQ search results and then composes a URL, e.g., UserB@research.telcordia.com. Subsequently, the SEC client creates a SIP SUBSCRIBE message with User B s URL in the To header field and sends the message to the PAL Manager. If User B is currently not logged in, the PAL Manager sends a 200 Ok response to the SEC client of User A. The response indicates that User B is OFFLINE.

If User B is in the system, the PAL Manager creates a new SUBSCRIBE message and sends it to the SEC client of User B. This SUBSCRIBE message is to alert User B that User B has a PA watcher, in this example, User A. However, this SUBSCRIBE message specifies the PAL Manager as the PA watcher and has the address of the PAL Manager as the NOTIFY contact. If User B has other watchers, and if the PAL Manager has already sent and received a successful response to a similar SUBSCRIBE message, the PAL Manager does not send another SUBSCRIBE message. Figure 6 graphically illustrates this design.

The goal of this design is two-fold. First, it enables the PAL Manager to keep track of all the PA subscriptions in SEC and their current status. In turn, as is discussed shortly, this removes from users the administrative burden of maintaining and downloading their PALs to new locations. Second, it facilitates an efficient implementation of the open PA subscription policy that allows every user to subscribe to all other users in SEC. By not alerting a user of all of his or her subscribers, the message traffic between the PAL Manager and the SEC client of the user is greatly reduced. If the user wishes to find out the list of his or her subscribers, the user can contact the PAL Manager.

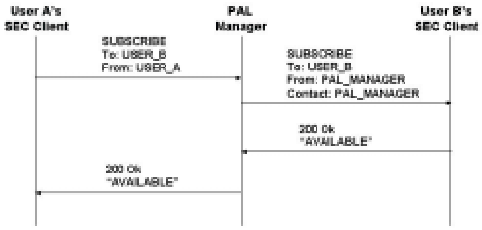


Figure 6: PA Subscription in SEC

When User B s SEC client returns a successful response to its SUBSCRIBE message, the PAL Manager records User B s current PA state, which is included in the response and sends a successful response with the same PA state to User A s SEC client. Subsequently, whenever User B s PA state changes, User B s SEC client sends a SIP NOTIFY message to the PAL Manager. The PAL Manager records User B s PA state and sends a NOTIFY message to User A s SEC client. This NOTIFY message has User B s URL in its From header.

SEC actively manages PALs on behalf ofend users in order to better support mobility. To illustrate, say User A in the previous example logs out of SEC, moves to a new terminal, and logs back into SEC from the new location. In order to download his or her PAL without any support from SEC, User A somehow has to remember the contents of the PAL and re-subscribe to the PA state of his or her contacts. A better approach would be to have User A store the PAL somewhere in the network and then download the PAL file to the new terminal. This would work well, except that User A has to remember the location of the PAL file. To improve upon this approach, the PAL Manager saves the PAL to a permanent store periodically and when User A logs out. When User A logs backs in, the PAL Manager retrieves the PAL file for the user and sends the PAL contents to the user in its response to the REGISTER message. This approach frees users from having to remember the locations of their PAL files, no matter where they are logging in.

4) Conference Set up and Management

As described in Section F, SEC employs voice/text bridges, namely MCU and MTCU, to provide its spontaneous conferencing services. In SEC, spontaneous conferencing means that the process of setting up and managing conferences should be streamlined to the extent possible. In particular, users should not have to schedule conferences or determine conference membership in advance. To this end, SEC makes an extensive use of the SIP redirection feature. In this section, we discuss in detail the conference setup and management schemes in SEC.

Specifically, SEC combines the model of users calling a conference bridge with that of a conference bridge calling users. In addition, SEC makes use of the proposed SIP SUBSCRIBE and NOTIFY methods to convey conference participant lists. That is, in SEC, each conference participant SUBSCRIBEs to the PA state of his or her conference. Henceforth, SEC NOTIFies the participant whenever the PA state of the conference changes, i.e., a new participant has joined the conference. Figure 7, Figure 8, Figure 9, and Figure 10 graphically illustrate the SEC conference set up and management schemes in the context of User A inviting User B to a conference. In these figures, the communication between the SEC server components is described in terms of the SEC proprietary protocol.

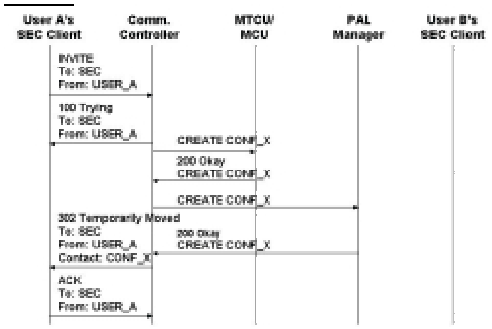


Figure 7: Creating a Conference in SEC

In SEC, when a user wishes to communicate with another user, the SEC client first creates a conference Figure 7 shows how a conference is created in SEC. In the figure, User A s SEC client sends a SIP INVITE message to the Communication Controller (CC) server. This INVITE message is addressed to a user who is a default super user, SEC. In SEC, all the INVITE messages that initiate a conference are addressed to this user. This INVITE message also includes a Require header that specifies the type of conference to be created, i.e., voice or text. Upon receiving this INVITE message, the CC creates a new conference and assigns a unique identifier, e.g., CONF_X in Figure 7). In addition, the conference is assigned an URL that is created by combining its identifier with the name of the domain where SEC is used, e.g., CONF_X@research.telcordia.com. SEC makes extensive use of conference URLs in providing its conference services.

Subsequently, the CC chooses an MTCU or MCU server to be used for this conference, and notifies the server of the new conference. The CC also notifies the PAL Manager of the new conference so that the participants of this conference can SUBSCRIBE to the PA state of the conference. Finally, the CC sends a 302 Moved Temporarily response to the SEC client of User s A. The 302 response is a standard SIP response for redirecting calls. In SEC, the 302 response includes the URI of the new conference in its Contact: header. The SEC client of User A acknowledges the receipt of this response by sending a SIP ACK message to the CC.

Once a conference is created, the caller joins the conference. This is equivalent to a conference participant calling a PSTN bridge in order to join a conference. Figure 8 shows User A joining CONF_X. As shown in the figure, User A s SEC client sends a new SIP INVITE message to the CC. This INVITE message is addressed to the URI of CONF_X. It also contains the session description that User A wishes to use

in this conference. For a voice conference, this session description either contains the IP address and port number at which User A s SEC client listens for incoming RTP/RTCP packets or the telephone number of the PSTN phone that User A wishes to use for this conference. For a text conference, it has User A s contact address to which SIP MESSAGEs should be sent.

Upon receiving the INVITE message from User A s SEC client, the CC alerts the MTCU/MCU for this conference that User A is joining the conference. In addition, the CC sends User A s session description to the MTCU/MCU. In turn, if User A wishes to use a phone, the MCU establishes an H.323 session with a PSTN Gateway Proxy, which in turn calls User A s phone via a PSTN gateway.

Meanwhile, the MTCU/MCU acknowledges to the CC that User A has joined CONF_X. This acknowledgement includes the IP address and port number to which User A s SEC client should send its voice (if User A is not using a PSTN phone) or text messages. Subsequently, the CC alerts the PAL Manager that User A has joined CONF_X. Finally, the CC sends a SIP 200 Okay response to User A s SEC client. This response contains the IP address and port number that the CC has received from the MTCU/MCU for CONF_X.

Next, User A s SEC client sends a SIP SUBSCRIBE message to the PAL Manager to subscribe to the PA state of the CONF_X conference. This SUBSCRIBE message is addressed to the URL of CONF_X. Upon receiving this SUBSCRIBE message, the PAL Manager verifies that User A is a participant of the CONF_X conference and sends a SIP 200 Okay response. This response contains the current participant list of CONF_X, i.e., User A at this point. The PAL Manager also records that it should send a SIP NOTIFY message whenever the PA state of CONF_X changes.

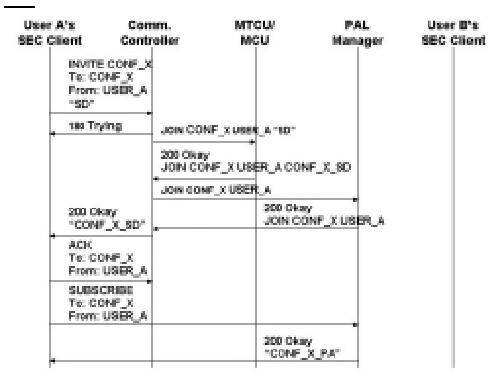


Figure 8: Caller Joining Conference in SEC

Subsequently, the SEC client of User A invites User B to CONF_X. To this end, one standard approach is that User A first calls User B and then transfers User B to CONF_X. However, this approach requires the SEC client of User A to be involved with two simultaneous calls unnecessarily and incurs extra message overhead. Therefore, SEC takes more a direct approach, in which a conference invites a potential participant on behalf of a caller. Figure 9a and Figure 9b graphically illustrate this approach.

In Figure 9a, the SEC client of User A sends a INVITE message to CONF_X at the CC. The request URI of this INVITE message contains the URL of CONF_X, but the To header contains the URL of User B. The CC interprets this message as User A inviting User B to CONF_X. Thus the CC sends an INVITE message to the SEC client of User B. This INVITE message has as its From and Contact headers the URL of CONF_X. Furthermore, this INVITE includes as part of its message body the current participant list of CONF_X, i.e., User A at this point. A better approach would be to have a header to convey this information. Such a header is not yet defined.

Upon receiving the INVITE message from the CC, the SEC client of User B sends a 200 Ok response that includes the session description of User B. The semantics of this session description is the same as that of User A. Upon receiving this response from User B, the CC alerts the appropriate MTCU/MCU that User B has joined CONF_X. This alert includes the session description of User B. Then the MTCU/MCU processes User B s session description, including, if CONF_X is a voice conference, calling the PSTN phone of User B s choice as specified in User B s session description. Subsequently, the MCU/MTCU acknowledges User B s joining CONF_X to the CC. This acknowledgement includes the IP address and port number to which User B s SEC client should send its voice (if User B is not using a PSTN phone) or text messages. Then the CC alerts the PAL Manager that User B has joined CONF_X.

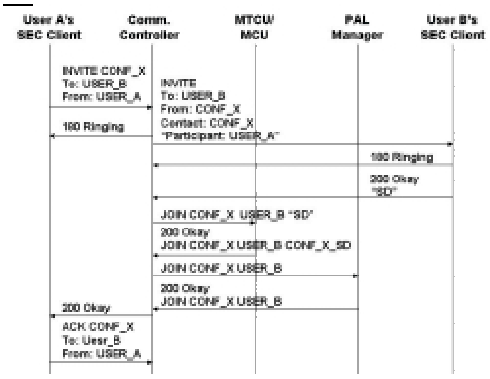


Figure 9a: Inviting User To Conference in SEC

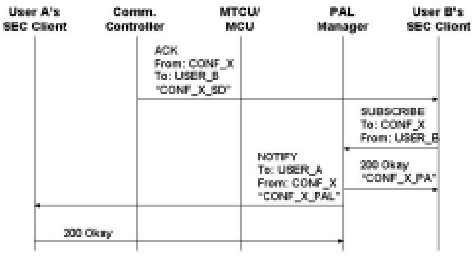


Figure 9b: Inviting User to Conference in SEC

Next, in Figure 9b, the CC sends an ACK message to User B s SEC client. If necessary, this ACK messages contains the CONF_X session description that the MTCU/MCU has sent for User B. Subsequently, User B s SEC client subscribes to the PA state of CONF_X by sending a SUBSCRIBE message to the PAL Manager. The PAL Manager sends a 200 Ok response that includes the current PA state of the CONF_X, namely User A and User B. In addition, the PAL Manager sends a NOTIFY message to User A s SEC client. This message includes the current PA state of CONF_X, namely User A and User B.

5) Conference Spawning

Spawning a conference means creating a new conference out of an existing conference. The new conference inherits the membership of the existing conference. In SEC, conference spawning is provided as a means of changing the type of communication media on demand, e.g., from a text conference to a voice conference and vice versa.

One issue to consider in providing conference spawning is concurrency control. Without any provision, it is possible that multiple participants will try to spawn the same conference. One approach to addressing this issue is to designate a participant as a conference administrator and enforce a policy that allows only the administrator to spawn the conference. However, in SEC, it is difficult to determine who should be the administrator, especially considering that the main idea behind spontaneous conferencing is to allow participants to freely interact with each other. Yet another approach is to have the conference spawning operation require a lock. Whoever acquires the lock first gets to spawn the conference. However, the extra step of having to acquire a lock could disrupt the fluidity of participant interactions in a spontaneous conference. In addition, always requiring a lock demands, unnecessarily, a high processing overhead, especially considering the presence of a social protocol that tends to prevent conflicting, concurrent operations in group work [20].

Therefore, SEC allows multiple participants to make a request to spawn the same conference. In order to prevent the same conference from being spawned multiple times, each request is made as originated, not from individual participants, but from the conference itself. This allows SEC to detect multiple requests to spawn the same conference as duplicates and process them accordingly.

As discussed in Section G.4, to create a conference, an INVITE is sent to the CC. The To header in this INVITE has the URL of a super user, called SEC. When spawning a conference, the From header in this INVITE contains the URL of the spawned conference. The CC honors only one such INVITE, e.g., the first one to be received, and ignores the rest. The rest of the conference spawning process is similar to the process of creating a new conference. The SEC client whose request to spawn the conference is honored is responsible for inviting the rest of the participants in the spawned conference.

6) Text Communications

SEC provides instant messaging service using the proposed SIP MESSAGE method [17]. In SEC, an instant messaging is primarily between two users. For a text communication between more than two users, a text conference is created. The process of creating a text conference is similar to that of creating a voice conference and is described in Section G.4.

In order to send a text message to the participants in a text conference, the SEC client of the sender sends a MESSAGE addressed to the URL of the conference. Upon receiving this MESSAGE, the MTCU for the conference sends to each conference participant a new MESSAGE containing the sender s message in its message body. However, one issue is how to convey the identity of the sender in the new MESSAGE. One approach is to have the sender s URL in the From header and the recipient s URL in the To header. However, this approach loses the context information that specifies that this message is sent within a text conference. Another approach is to include the URL of the conference in the From header in the new MESSAGE and the URL of the recipient in the To header. However, this approach loses the identity of the sender. Currently, SEC adapts the latter approach, including the sender URL in the message body.

H. IMPLEMENTATION AND INITIAL USAGE EXPERIENCE

The current implementation uses a SIP stack being built in-house. The CC, PAL Manager, MTCU, and SEC client are written in Java. The MCU, PSTN Gateway Proxy and the audio module embedded in the SEC client are written in C++. The audio mixing in the PSTN Gateway Proxy is completely done in software and requires no specialized hardware. The audio module uses the Microsoft DirectX technology to capture mic input and mix and play back incoming voice packets. When available, the mixing capability on sound cards is automatically utilized by DirectX. The SEC Web client is a Java Servlet that interfaces with SEC on behalf ofa user. A Cisco AS5300 gateway is used to call PSTN phones. For voice conferences, G.711 is used throughout the system Server and client machines are all PCs running either NT 4.0 Workstation/Server or Windows 2000 Professional.

SEC is in a prototype stage, and as such, no formal study has been done on the usage of SEC services. However, the current prototype has been in regular use among 4 users, who are also members of the SEC team, for the past few months. Two of the users have been actively using the AOL IM application, while the others have had a limited exposure to IM applications in general. Not surprisingly, much of use of

SEC has been related to SEC itself, e.g., reports of new bugs and bug fixes, discussions on future plans, etc.

One of the benefits of using SEC for instant messaging is security. Although the current prototype does not yet provide a security measure, the fact that instant messages do not go over the public network generates a sense of security and has been the main motivation for using SEC for exchanging work-related instant messages. Voice is preferred for long, focused discussions. Often times, in text conferences, one of the participants would inevitably send a message, saying, go to voice? for topics that require a lot of explaining. Another feature that is often used is the ability to directly call PSTN phones from the SEC client. This enables users to contact other users even when they are not logged into the system and thus greatly extends the usability of SEC.

Our usage experience also shows that the interaction with PSTN features should be improved. For example, when directly dialing a phone number in a voice conference, an answering machine or voice mail sometimes answers. In such a case, the ability to disconnect the dialed number from the conference is mandatory.

We plan to deploy SEC within our department and to conduct a formal usability, performance, and scalability study in near future. The issues of privacy and use of PA state within an enterprise organizational hierarchy should be explored.

I. FUTURE WORK AND CONCLUSIONS

In this paper, we presented SEC, a communications system designed to provide spontaneous conferencing in distributed enterprise workplaces. Spontaneous conferencing is analogous to ad-hoc, drop-in meetings and impromptu discussions in multi-person offices. SEC enables users to initiate and invite participants to ongoing conferences on demand from their PALs, where they can observe the changes to the presence and availability state of their colleagues in real time. To better support enterprise communications, SEC provides both text and voice conferencing capabilities and the ability to spontaneously switch to a different communication media type, i.e., from voice to text and from text to voice. For voice conferences, SEC supports both VoIP and PSTN phone users. SEC also provides a Web interface so that mobile users with networked PDAs and cellular phones can access SEC services. SEC makes extensive and innovative use of SIP to provide its services.

SEC is a work in progress. In particular, we plan to conduct a formal performance and scalability study of SEC s server-based approach to multiparty communications. Given that the PSTN phone is still the device of choice for voice communication in a typical enterprise workplace, increasing the performance and scalability of the SEC PSTN Gateway Proxy is especially critical.

We also plan to include support for automatic availability management. A popular approach is to use the frequency of user actions at the keyboard and/or on the mouse. We are currently looking to improve the performance of this approach by taking into account other parameters that better represent users availability, such as the state of a user s computer desktop. Another issue is access control to presence and availability in an enterprise environment. One possible approach is to hierarchically organize access rights [22] to reflect an organizational hierarchy. However, this approach may not work well for team-oriented environments, where the organization of a team may not reflect the organizational hierarch of an enterprise. We are also investigating ways to apply SEC PAL and conferencing services to better address data and document sharing needs in a distributed enterprise workplace.

REFERENCES

[1] M. Handley, H. Schulzrinn, E. Schooler and J. Rosenberg, SIP: session initiation protocol, Request for Comments (Proposed Standard) 2543, Internet Engineering Task Force, Mar. 1999.

[2] M. Handley and V. Jacobson, SDP: session description protocol, Request for Comments (Proposed Standard) 2327, Internet Engineering Task Force, Apr. 1998.

[3] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, RTP: a transport protocol for real-time applications, Request for Comments (Proposed Standard) 1889, Internet Engineering Task Force, Jan. 1996.

[4] International Telecommunication Union, Packet based multimedia communication systems, Recommendation H.323 Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Feb. 1998.

[5] S. Casner and S. Deering, First IETF Internet Audiocast, *ACM SIGCOMM Computer Communication Review*, pp. 92-97, ACM, July 1992.

[6] V. Jacobson and S. MacCanne, vat manual pages, LBL, UC Berkeley, CA.

[7] E.F. Churchill. and S. Bly, Virtual Environments at Work: ongoing use of MUDs in the Workplace. *Proceedings of the 1999 International Joint Conference on Work Activities Coordination and Collaboration (WACC 99)*, San Francisco, CA, USA. ACM Press, 1999, pp. 99-108.

[8] E.F. Churchill. and S. Bly, It s all in the words: Supporting work activities with lightweight tools. *Proceedings of the 1999 ACM International Conference on Supporting Group Work (GROUP 99)*, pp. 40-49, Phoenix, Arizona, USA. ACM Press, 1999,

[9] R. Evard, Collaborative Networked Communication: MUDs as System Tools. *Proceedings of the Seventh Administration Conference (LISA VII)*, pp. 1-8, Monterey, CA, Nov. 1993.

[10] M. Roseman and S. Greenberg, TeamRooms: network places for collaboration. *Proceedings of the 1996 ACM Conference on Computer-Supported Cooperative Work (CSCW 96)*, pp. 325-333. Cambridge, MA, USA, 1996.

[11] J.H. Lee, A. Prakash, T. Jaegert, and G. Wu, Supporting multi-user, multi-applet workspaces in CBE. *Proceedings of the 1996 ACM Conference on Computer-Supported Cooperative Work (CSCW 96)*, pp. 344-353. Cambridge, MA, USA, 1996.

[12] S. Subramanian, G.R. Malan, H.S. Shim, J.H. Lee, P. Knoop, T. Weymouth, F. Jahanian, and A. Prakash, Software architecture for the UARC Web-based collaboratory, *IEEE Internet Computing*, vol. 3, Issue 2, pp. 46-54, Mar.-Apr. 1999.

[13] K. Birman, A. Chipper, and P. Stephenson, Lightweight causal and atomic group multicast, *ACM Trans. On Computer Systems*, vol. 9, no. 3, pp. 272-314. Aug. 1991.

[14] H. Eriksson, Mbone: the Multicast Backbone, *Communications of the ACM*, vol. 87, no. 8, pp. 54-60, Aug. 1994.

[15] T. Dorcey, CU-SeeMe Desktop Video Conferencing Software, *Connexions*, vol. 9, no. 3, March 1995.

[16] J. Rosenberg, D. Willis, R. Sparks, B. Campbell, H. Schulzrinne, J. Lennox, B. Aboba, C. Huitema, D. Gurle, and D. Oran, SIP Extensions for Presence, Internet Draft, Internet Engineering Task Force, June 2000, Work in progress.

[17] J. Rosenberg, D. Willis, R. Sparks, B. Campbell, H. Schulzrinne, J. Lennox, B. Aboba, C. Huitema, D. Gurle, and D. Oran, SIP Extensions for Instant Messaging, Internet Draft, Internet Engineering Task Force, June 2000, Work in progress.

[18] J. Rosenberg and H. Schulzrinne, Models for Multi Party Conferencing in SIP, Internet Draft, Internet Engineering Task Force, Nov. 2000, Work in progress.

[19] M. Day, J. Rosenberg, and H. Sugano, A model for presence and instant messaging, Request for Comments 2778, Internet Engineering Task Force, Feb. 2000.

[20] S. Greenberg and D. Marwood, Real-time groupware as a distributed system: concurrency control and its effect on the interface, *Proceedings of the 1994 ACM Conference on Computer-Supported Cooperative Work (CSCW 94)*, pp. 207-217, Chapel Hill, NC, 1994.

[21] J. Lauwers, T. Joseph, K. Lantz, and A. Romanow, Replicated architectures for shared window systems: a critique, *Proceedings of ACM Conference on Office Information Systems*, pp. 249-260, March 1990.

[22] H. Shen and P. Dewan, Access Control for Collaborative Environments, *Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work (CSCW 92)*, pp. 51-58, 1992.

[23] A. Milewski and T. Smith, Providing Presence Cues to Telephone Users, *Proceedings of the 2000 ACM Conference on Computer-Supported Cooperative Work (CSCW 00)*, pp. 89-96, 2000.

[24] Microsoft, http://www.microsoft.com/windows/netmeeting.

[25] Microsoft, http://www.microsoft.com/exchange/productinfo/conferencing.htm.

[26] ME.net, http://www.me.net.

[27] WebEx, http://www.webex.com.

Easy Accessible Voice Gateway between Mbone and ISDN/PSTN Networks

Linqing Liu, Torsten Braun

Institute of Computer Science and Applied Mathematics, University of Bern

Neubrückstr. 10 CH-3012 Bern, Switzerland

{liu|braun}@iam.unibe.ch

Abstract—This paper presents a solution for setting up an easy accessible voice gateway between Mbone and ISDN/PSTN networks. It implements gateway control and administration mechanisms and utilizes some available public domain software to achieve necessary functionalities. With this solution, users can easily set up voice conferences with mobile WAP phones or web browsers, and join in multicast sessions using (mobile) telephones.

Index Terms—Gateway, Mbone, ISDN, PSTN, multicast.

I. Introduction

IP networks begin to play an active role in voice communication. IP telephony exploits open IETF and ITU standards to carry multimedia traffic over IP networks, offering users more flexibility. IP based telephony has the potential to significantly reduce the costs of long-distance voice communication. In addition to cost effective impacts with IP telephony, it would also be convenient to set up audio conferences over the Internet without the need of an expensive Multipoint Control Unit (MCU). IP multicast provides efficient many-to-many data distribution in an internet environment. The multicast backbone (Mbone) is a very suitable media for serving the purpose of multiparty conferences.

In this paper, we describe a solution to implement an easy accessible voice gateway between Mbone and ISDN networks based on public domain software. When users wish to join a multicast session, they can get a list of available session information from a HTML/WML web site. After selecting desired sessions and registering with the telephone number and email address etc. via a WAP capable mobile phone or a regular web browser, they get contact information such as gateway telephone numbers and session dial-in numbers. With the gateway contact information they can dial into the gateway to join the sessions. To initiate a conference, a user can create a new session via a WAP mobile phone or a web browser, and send the session information to other participants via email or SMS messages. It is also possible that the gateway calls other registered users connected with the gateway for joining the sessions.

The rest of the paper is organised as the follows: In section II we give a brief review of related work. Section III provides an overview of the proposed easy accessible solution. In section IV we elaborate the audio forwarding and support unit of the gateway, section V presents the control unit of the gateway, and finally section VI concludes the paper.

II. Related Work

Many approaches related to ISDN and Mbone applications have already been developed. Some of them have been used and extended in our gateway architecture.

ISDN4Linux [12] is a set of Linux kernel modules, which consists of the main module ISDN and the actual hardware driver that controls some specific card. ISDN4linux can control ISDN cards that are connected to the PC's ISA or PCI bus. Basically, ISDN4Linux can receive and transmit data via ISDN in several ways.

An architecture and prototype implementation to allow PSTN users to participate in Mbone conferences has been introduced in [1]. The basic operation allows users to dial up the gateway with a conventional phone to join multicast sessions. The speech generation and recognition component is integrated into the solution to enable users to get session information and select session by voice.

The Audio Mbone-Telephony Gateway "AudioGate" developed in the MECCANO project [13,14] provides a dial-in interface that allows users on an arbitrary telephone network (PSTN, ISDN and GSM) to call a phone number and automatically be transferred into a pre-selected Mbone session. AudioGate uses an ISDN BRI to connect to the phone network. Upon connection setup, functions such as dynamic conference selection will be provided. As soon as a "connection" to an Mbone session is established, additional services such as user identification, muting etc. could be provided.

III. Overview

The target scenario of the easy accessible gateway is shown in Fig.1. The gateway provides a user-friendly HTML/WML

interface via the World Wide Web in addition to basic gateway functions. The gateway users can get all the necessary information regarding to multicast session information, gateway contact information etc. from web pages which can easily be accessed by a regular web browser or a WAP mobile phone before dialing up to the gateway.

The gateway users can also create new multicast sessions via HTML/WML pages by filling the form with necessary information such as session name, session description, user contact information, session duration, regional scale etc. The gateway will announce the new session over the Mbone and display the new session address and port in HTML/WML page as well as send them by email or SMS message to the users.

Here we give two examples of applications for gateway users.

First, we assume a user wishing to retrieve an existing live audio session with a mobile phone. This is a simple and straightforward process. What a user needs to do is to use a WAP capable mobile phone or any web browser to access the web page provided by the gateway, and select the session s/he wants to join. Then, s/he registers necessary user information such as telephone number and email address etc. with the gateway. The gateway will display access information including dialing number and session entry number via HTML/WML page or SMS/Email message. The user can then dial up the number and join the session by telephone. The client-server mechanism implemented in the gateway is shown in Fig.2.

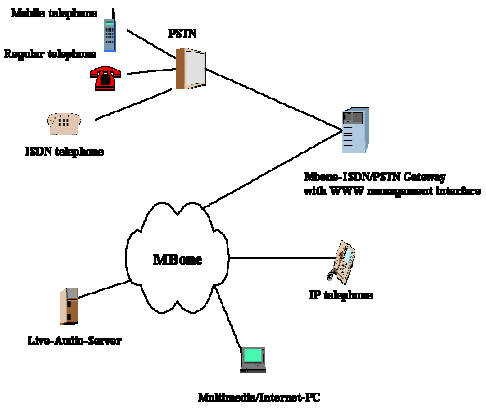


Fig. 1 Easy accessible Mbone-ISDN/PSTN gateway scenario

In the second case, the user is supposed to launch an audio conference over the Mbone with telephone access. There are two steps required for setting up the conference. The first step will be to create a new session for the conference and send the session information to other participants via email/SMS to invite them to join the conference with whatever facilities they prefer. By accessing HTML/WML pages in the gateway's web server, the user can easily create a new session over the Mbone with a WAP capable mobile phone or any web browser. The user will be asked to input session name, session description, user information, session scope, and session duration etc. The new session address/port/TTL will immediately be available upon the completion of the session creation procedure. The second step is to join the new created session which is identified by session address/port, and it will be the same procedure for this step as in the first case. The conference scenario is shown in Fig. 3.

The configuration of the gateway is required for allowing users to access the gateway with their telephone numbers to be authenticated in the gateway audio forwarding software. The available sessions' addresses and ports and TTLs as well as those session selecting access numbers should also be added to the gateway initialization files.

The main functionality of the gateway can be accessed via HTML/WML pages including user registration, gateway configuration, creating new multicast sessions, and gateway management. Gateway users and system operators can access the gateway very easily.

The gateway management and initialization can be done via a HTML interface for the gateway operator. The gateway system administrator can configure the gateway from the web.

The internal gateway architecture is depicted in Fig.4. It consists of two units: the gateway control unit and the audio forwarding and support unit. The audio forwarding and support unit is responsible for converting audio data between Mbone and ISDN/PSTN networks, listening to existing Mbone sessions and announcing new sessions over the Mbone. The main functionalities for this unit are realised by integrating public domain software such as AudioGate[15], mSD[10], and mAnnouncer[11]. The gateway control unit is implemented to achieve easy accessibility of the gateway configuration and initialization. Also accounting can be supported. The detailed information of this unit is described in section E.

IV. AUDIO FORWARDING AND SUPPORT UNIT

The audio forwarding and support unit (AFSU) carries out the basic functionalities of the gateway including converting audio

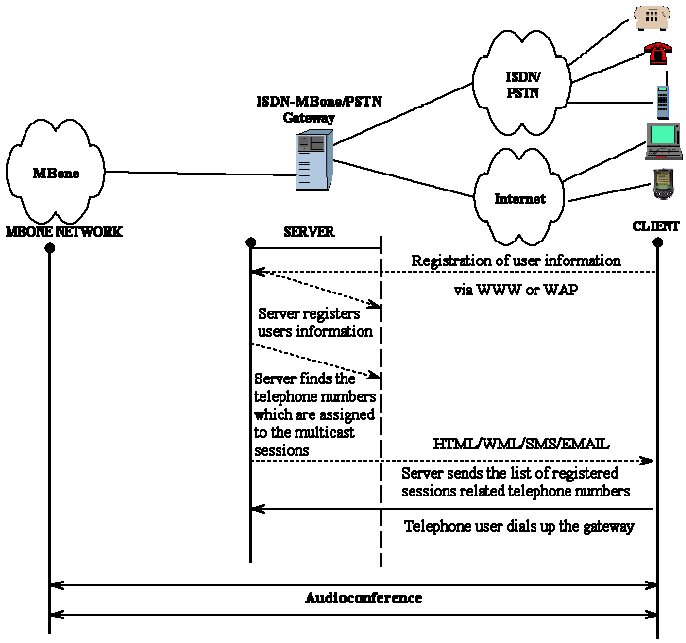


Fig. 2 Client-server mechanism

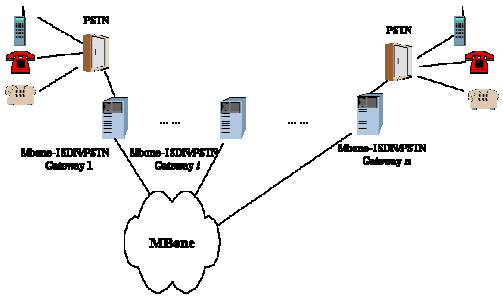


Fig. 3 A scenario for audio conference over Mbone using Mbone-ISDN/PSTN gateway

data between Mbone and ISDN/PSTN networks, listening to multicast session information and announcing new sessions over the Mbone.

The teleconferencing applications in Internet and ISDN/PSTN networks are currently using different standards. The IETF has

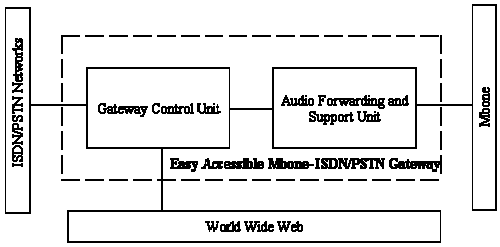


Fig. 4 Gateway architecture

developed a conferencing architecture [2] that primarily supports loosely coupled teleconferences in the Internet while the ITU-T published series of Recommendations for more tightly coupled multimedia communications in various networks including ISDN (H.320) and the Internet (H.323). To enable the communication between ISDN/PSTN and Internet multicast backbone (Mbone), a gateway with audio forwarding mechanisms to achieve interoperability for different systems is fundamental.

Announcements are delivered to the audio forwarding and support unit to be multicasted. The Mbone announcements

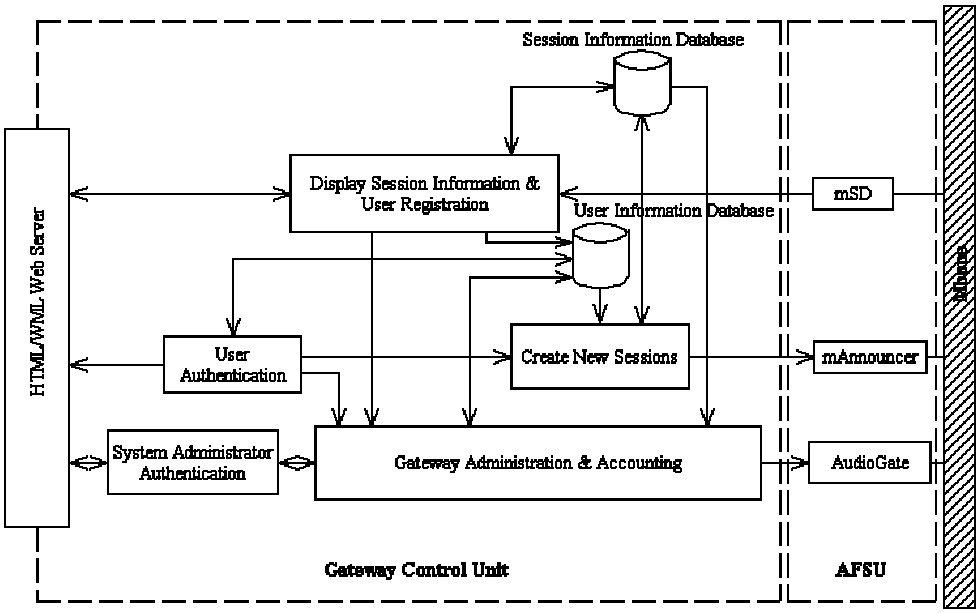


Fig. 5 Gateway control unit

are sent using the Session Announcement Protocol (SAP) with contents specified using the Session Description Protocol (SDP) [5,6]. The multicast group address assigned to Mbone session announcements is 224.2.127.254 that is known as sap.mcast.net via the Domain Name Service (DNS), the well-known UDP port 9875 is used.

The MECCANO project [13] has developed various gateways for PSTN, ISDN and GSM users to realize audio forwarding mechanisms between Mbone and ISDN/PSTN, and the software is available in the public domain. In our solution, AudioGate from MECCANO has been selected to serve for establishing connections between Mbone and ISDN/PSTN. It can be integrated into the solution for the implementation of an easy accessible gateway, which provides users access to multicast audio sessions via arbitrary telephone networks.

Basically, AudioGate provides a dial-in point for users on any telephone network and enables them to participate in Mbone conferences. The main features include [15]:

- Calling users can be authenticated based upon their Calling Line Identification (CLI), i.e. the telephone numbers they are using when making the call. AudioGate can selectively allow or disallow access to certain users, as well as to users that do not provide the calling party number information.
- For users that do provide a calling party number, AudioGate allows to assign a full name to each of these numbers for presentation in the Mbone session.
- AudioGate provides a simple (easily configurable) voice menu for prompting callers for a conference selection.
- Conferences can either be explicitly selected through the voice menu or callers can automatically be put into a particular conference. Optionally, this may depend on the phone number they dial. The mode of operation is controlled through the configuration file.
- AudioGate imposes no artificial limits on the number of simultaneous calls it can handle. Besides processing power for audio coding, the only practical limit is the number of B channels that are available. AudioGate supports one or more ISDN BRI boards.
- AudioGate supports up to nine simultaneous conferences. The calls from the telephone network may all lead into the same conference or into different ones.
- AudioGate can be configured to use any number of ISDN Multiple Subscriber Numbering (MSN). Some of

them can be directly mapped into a conference while others may first lead into a voice menu.

- AudioGate supports muting/unmuting callers using Dual-Tone Multi-Frequency (DTMF) for control. The initial status for the callers can be configured in the AudioGate initialization file.
- All codecs and transmission modes (particularly including redundancy coding) supported by Robust Audio Tool (RAT) [16] are supported by AudioGate.

The configuration of AudioGate is initialized based on simple format ASCII files which will be managed through the gateway control unit described in the next section.

The Mbone session information are fetched with mSD[10], a public domain software which can be integrated in the audio forwarding and support unit as a daemon program. mSD generates a series of session information files in its cache directory that can be transformed into HTML/WML pages with Perl programs at the gateway's web server.

For session announcement, we exploit mAnnouncer [11] to send out new session information over the Mbone. mAnnouncer is a Java application available in the public domain for sending out session announcements periodically using the Session Announcement Protocol [6] and the Session Description Protocol (SDP) [5]. Comparing with other available tools for session announcement such as SDR [14,15], mAnnnouncer interacts more conveniently with Perl programs which we use for building the gateway control unit.

V. GATEWAY CONTROL UNIT

To achieve the easy accessibility of the gateway and other high-level gateway management functionalities, we implemented the gateway control unit on top of the audio forwarding and support unit. The gateway control unit plays a key role for providing the following functionalities:

- It maintains a database and displays the current available Mbone sessions via HTML and WML pages. The database is automatically updated upon the request from the gateway users via HTML/WML interface.
- It provides a registration mechanism via HTML and WML pages and maintains a database for registered users.
- It displays session specific contact information for users via HTML/WML/SMS/Email.
- It provides the mechanisms for creating new multicast sessions via HTML and WML pages.
- The AudioGate configuration is controlled via HTML pages and the AudioGate settings are updated automatically according to the session database and user information database.
- It provides gateway administration functionalities including gateway configuration and accounting.

The main functionalities of the gateway control unit and its internal structure of the implementation are shown in Fig.5. This unit has three main blocks in addition to two databases for realizing user registration, creating new sessions, and gateway administration.

Basically, the existing Mbone session information is saved in the Session Information Database, which serves for displaying session information, registering user information with feedback of available session contact information, and checking existing session addresses/ports when generating new session addresses/ports. The user information database will be used to further processing of user-selected sessions, user authentication, and billing purposes.

A. Mbone session selection and user registration

When gateway clients wish to join multicast sessions, they have first to contact the gateway web server via a conventional web browser or a WAP mobile phone, to get a list of current available multicast sessions. Then, they register to the gateway with their telephone numbers and email addresses etc. for enabling the gateway to allow them to call in with their Calling Line Identifications (CLIs).

In order to display current available multicast sessions, a set of Perl programs have been implemented to fetch cache files generated by the daemon program mSD running on the server and to transform them into appropriate HTML and WML pages together with user registration forms. The program is capable to generate multicast session information which will be selected according to the keywords entered by user via HTML/WML pages.

The output is sent out to the users' web browser or WAP capable mobile phone for further registration. The session information will be saved in the session information database for providing session addresses/ports/TTLs to generate the AudioGate initialization file. Since the session information is dynamically changed and updated all the time, it is necessary to update the session information database frequently.

In the user registration form appearing as HTML/WML pages, there will be a list of multicast sessions including the session names and session information. The users can tick the box corresponding to the desired sessions and input their names, telephone numbers and email addresses. All the selected session information including session addresses, ports, TTLs as well as the users' personal information will also be saved in the user information database.

The user information database will be automatically updated whenever a user registers a new session or changes user information. A snapshot of the user registration with a conventional web browser is shown in Fig.6, which shows a selected list of multicast sessions containing user-defined keywords together with user registration form in HTML pages. Fig. 8 shows a list of available sessions from a WAP mobile phone after user entering some keywords.

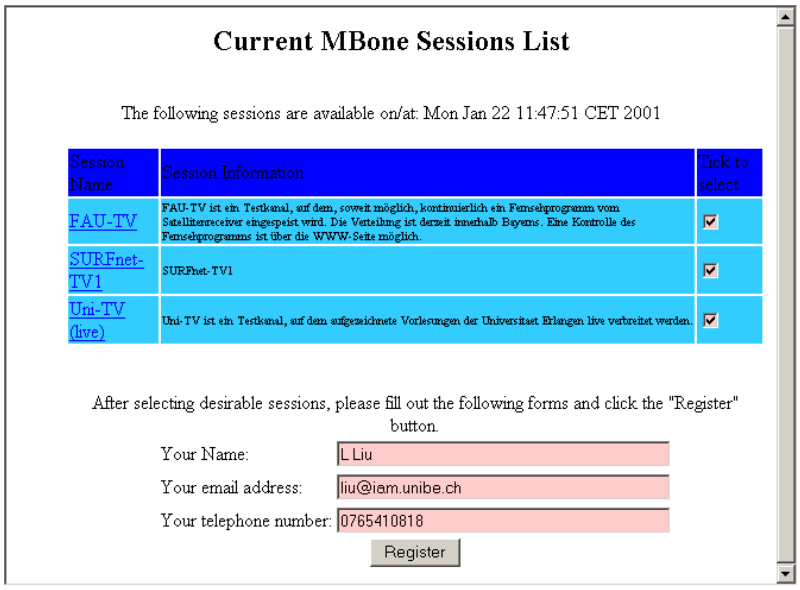


Fig. 6 Display available Mbone sessions and user registration in a conventional web browser

Fig. 7 Display available Mbone sessions contact information in a conventional web browser

(a) (b)

Fig. 8 Search available Mbone sessions in WAP mobile phone

Once users complete the registration process, the following HTML/WML pages will display all the necessary information for contacting the gateway, which consists of gateway telephone numbers and extension numbers for selected sessions. The same information can also be sent out automatically to the users as email or SMS. The users' information stored in the user information database will be used for user authentication and other user specific service such as user account information etc.

Fig. 7 shows the selected sessions contact information following the session selecting and user registration shown in Fig. 6. It is clearly shown that the users could then follow the contact information provided here and dial up the gateway to join the sessions.

If the sessions that users have selected are not assigned to the gateway, a message will automatically be sent to the gateway administrator including the unassigned session information. When the gateway administrator completes the session assignment procedure, the users can receive the session contact information.

B. Create new multicast sessions via HTML/WML

Creating a new multicast session is crucial for user-initiated conferences. It is convenient for users to be able to create new multicast sessions via HTML/WML for their own conference. To announce a new session in Mbone, a SDP[5] file needs to be generated according to session information. SDP is a textual description, which describes the formats, protocols, contents, timing, etc, used in the session. A session description consists of a session-level description (details that apply to the whole session and all media streams) and optionally several media-level descriptions (details that apply onto to a single media stream). An announcement consists of a session-level section followed by zero or more media-level sections.

When SDP is conveyed by SAP, only one session description is allowed per packet. When SDP is conveyed by other means, many SDP session descriptions may be concatenated together (the `v=' line indicating the start of a session description terminates the previous description). Some lines in each description are required and some are optional (marked with a `*' below) but all must appear in exactly the order given below (the fixed order greatly enhances error detection and allows for a simple parser) [5].

Session description:

v= (protocol version)
o= (owner/creator and session identifier).
s= (session name)
i=* (session information)
u=* (URI of description)
e=* (email address)
p=* (phone number)
c=* (connection information - not required if included in all media)
b=* (bandwidth information)

One or more time descriptions (see below):

z=* (time zone adjustments)
k=* (encryption key)
a=* (zero or more session attribute lines)

Zero or more media descriptions (see below):

Time description:

t= (time the session is active)
r=* (zero or more repeat times)

Media description:

m= (media name and transport address)
i=* (media title)
c=* (connection information - optional if included at session-level)
b=* (bandwidth information)
k=* (encryption key)
a=* (zero or more media attribute lines)

An example SDP description for a test session is shown as follows:

```
v=0
o=LLiu 3189143230 3190007230 IN IP4 130.92.66.147
s=A test session
i=Just a test.
u= http://coyote.unibe.ch/test.wml
c=L Liu <liu@iam.unibe.ch>
p=L Liu <0765410818>
t=3189143230 3190007230
a=type:broadcast
```

```
a=tool:mAnnouncer
m=audio 26210 RTP/AVP 0
c=IN IP4 224.20.146.105/63
a=ptime:40
```

We implemented Perl programs to create SDP files for user-defined new sessions. The generated SDP files are then used to trigger mAnnouncer[8] to send out the new sessions information via SAP[6] over the Mbone. In compliance with the requirements of mAnnouncer, a new line should be added on top of the SDP descriptions.

The generation of the SDP file for the new multicast session follows the requirements of Multicast Address Allocation (MALLOC) [3,4,8], SAP and SDP. The session information database and mSD cache files are used to avoid address collision since the new generated addresses/ports are selected randomly.

Fig.9 shows a set of screen snapshots for using a WAP mobile phone to create a new session. By selecting the scope of the new session (in Fig.9a) and entering necessary information such as session/user information and session duration etc. (in Fig. 9b-9f), the information of the created session including session address/port/TTL is displayed (in Fig. 9g). The new session information can then be sent out to all the participants who are invited to join the conference.

By utilizing the functionality provided by AudioGate, we enable gateway users to call other users connected to the gateway via WML/HTML page, i.e. provide "click-to-dial" for mobile and regular users to call other participants to join the sessions.

C. Gateway administration

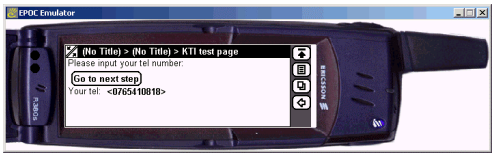
The gateway administration includes the gateway initialization, assigning existing sessions to different entry positions from the voice menu of AudioGate, user account maintenance as well as authentication. Currently, AudioGate supports nine sessions for each access number, which is required to be assigned in its initialization file.

The solution has been implemented so that the main tasks including gateway configuration, initialization and accounting are achieved via HTML page access after system administrator authentication.

The configuration of the gateway depends on the available multicast session addresses, ports, TTLs, registered users telephone numbers. The gateway administrator can manually assign the gateway telephone numbers and session selection extension numbers to the AudioGate initialization file via an HTML page (an example snapshot is shown in Fig. 10).

It is also possible to generate the AudioGate initialization file automatically from the database of the session information together with the user information database whenever a user submits registration information. The initialization of the gateway or refreshing with a new initialization file can also

(a)



(b)

(c)

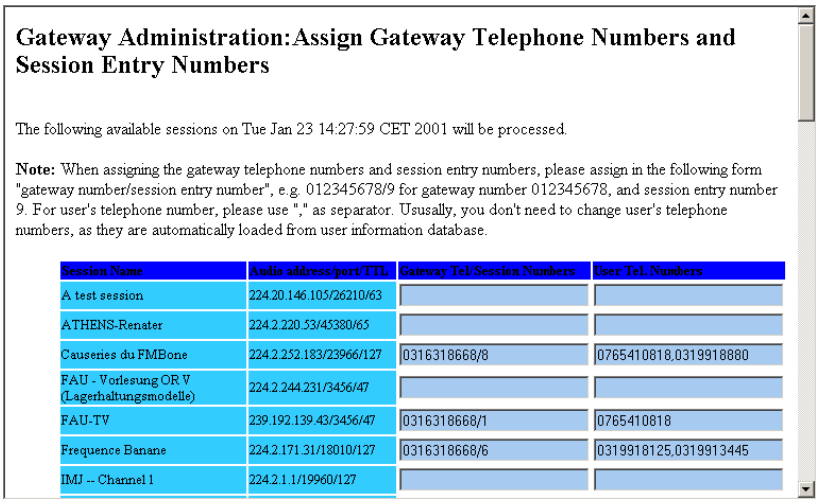
(d)

(e)

(f)

(g)

Fig. 9 Create a new session with WAP mobile phone



Gateway Administration: Assign Gateway Telephone Numbers and Session Entry Numbers

The following available sessions on Tue Jan 23 14:27:59 CET 2001 will be processed.

Note: When assigning the gateway telephone numbers and session entry numbers, please assign in the following form "gateway number/session entry number", e.g. 012345678/9 for gateway number 012345678, and session entry number 9. For user's telephone number, please use "," as separator. Ususally, you don't need to change user's telephone numbers, as they are automatically loaded from user information database.

Session Name	Audio address/port/TTL	Gateway Tel./Session Numbers	User Tel. Numbers
A test session	224.20.146.105/26210/63		
ATHENS-Renater	224.2.220.53/45380/65		
Causeries du FMBone	224.2.252.183/23966/127	0316318668/8	0765410818,0319918880
FAU - Vorlesung OR V (Lagerhaltungsmodelle)	224.2.244.231/3456/47		
FAU-TV	239.192.139.43/3456/47	0316318668/1	0765410818
Frequence Banane	224.2.171.31/18010/127	0316318668/6	0319918125,0319913445
IMJ -- Channel 1	224.2.1.1/19960/127		

Fig. 10 Assign gateway telephone number and session entry numbers.

be done in either manual or automatic ways.

Accounting which includes the cost allocation and billing mechanism is supported by setting up an accounting database, with parsing the log file of AudioGate and allocating cost table. A clip of the AudioGate log file is shown below:

... ...

```
Jan 12 08:45:42 coyote AudioGate (0.3)[8871]:
  accepting call from 59
Jan 12 08:45:42 coyote AudioGate (0.3)[8871]:
  call established
Jan 12 08:45:43 coyote AudioGate (0.3)[8871]:
  switching to session 233.4.218.44/23718/127
Jan 12 08:45:48 coyote AudioGate (0.3)[8871]:
  hanging up
Jan 12 08:45:48 coyote AudioGate (0.3)[8871]:
  call duration: (00:00:06)
Jan 12 08:45:48 coyote AudioGate (0.3)[8871]:
  switching to session 224.224.2.224/2222/127
Jan 12 08:45:50 coyote AudioGate (0.3)[8871]:
  accepting call from 59
Jan 12 08:45:50 coyote AudioGate (0.3)[8871]:
  call established
Jan 12 08:45:52 coyote AudioGate (0.3)[8871]:
  switching to session 224.2.246.13/30554/127
Jan 12 08:46:08 coyote AudioGate (0.3)[8871]:
  hanging up
Jan 12 08:46:08 coyote AudioGate (0.3)[8871]:
  call duration: (00:00:18)
```

... ...

Here it is clearly indicated that the time and the duration of the call from user 59 have been recorded, and hence they could be used for the accounting purposes.

D. Limitations and future work

The current solution has been implemented on a Debian2.2/Linux 2.2.15 platform with ISDN BRI (BRI: basic rate interface) boards. The number of lines the gateway can support depends on the number of ISDN boards in the gateway. The current version of AudioGate can only support up to nine sessions simultaneously.

Due to the limitation of AudioGate's maximum allocated sessions and inability for dynamic reconfiguring, it is difficult to get a seamless service to accommodate large number of sessions and users. It is desirable to develop a mechanism in future to provide seamless service without any interruption for current users when refreshing AudioGate with newly generated configuration file. The AudioGate's Mbus interface also provides the possibility to simplify the integration with other component in future development.

VI. CONCLUSION

This paper presents an easy accessible solution for a voice gateway between Mbone and ISDN/PSTN networks. The solution focused on the establishment of an easy accessible mechanism to control the gateway which enables mobile users as well as regular users to join or create audio conferences over Mbone via ISDN/PSTN networks. The presented solution can exploit existing resources to easily set up audio conferences while it is time consuming and complex to set up such a conference over ISDN/PSTN networks. The current solution has been implemented on a Linux platform and

makes use of some public domain software to realize necessary functionalities. The limitation for the scalability is partially relied on the available number of ISDN boards and the functionality of integrated public domain software. The latter is expected to be improved soon.

ACKNOWLEDGMENT

This work has been done within the project "Voice Gateway between Mbone and ISDN networks" funded by Bundesamt für Berufsbildung und Technologie (BBT) / Kommission für Technologie und Innovation (KTI) Project No. 4486.1 KTS and Telscom AG.

REFERENCES

[1] R. Ackermann, J. Pommnitz, L. Wolf, R. Steinmetz, MBone2Tel – Telephone Users Meeting the MBone, *Sixth International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS'99)*, October 12-15, 1999, Toulouse, France.

[2] M. Handley, J. Crowcroft, C. Bormann, J. Ott, The Internet Multimedia Conferencing Architecture, INTERNET-DRAFT, July, 2000.

[3] M. Handley, "Multicast Session Directories and Address Allocation", Chapter 6 of PhD Thesis entitled "On Scalable Multimedia Conferencing Systems", University of London, 1997.

[4] M. Handley, S. Hanna, Multicast Address Allocation Protocol (AAP), Work in Progress.

[5] M. Handley, V. Jacobson, SDP: Session Description Protocol, RFC 2327, April 1998.

[6] M. Handley, C. Perkins, E. Whelan, Session Announcement Protocol, RFC 2974, October 2000.

[7] M. Handley, D. Thaler, D. Estrin, The Internet Multicast Address Allocation Architecture, INTERNET-DRAFT, Dec 1997.

[8] S. Hanna, B. Patel, and M. Shah, Multicast Address Dynamic Client Allocation Protocol (MADCAP), RFC 2730, December 1999.

[9] http://www.aciri.org/malloc/

[10] http://www.cdt.luth.se/~peppar/progs/mSD/

[11] http://www.cdt.luth.se/~peppar/progs/mAnnouncer/

[12] http://www.isdn4linux.de

[13] http://www-mice.cs.ucl.ac.uk/multimedia/projects/meccano/

[14] http://www-mice.cs.ucl.ac.uk/multimedia/projects/meccano/deliverables/d67.2/mec-d67.2.pdf

[15] http://www-mice.cs.ucl.ac.uk/multimedia/projects/meccano/deliverables/d4.3/mec-d4.3.html

[16] http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/

Interworking and Feature Interaction

Interworking Internet Telephony and Wireless Telecommunications Networks

Jonathan Lennox, Kazutaka Murakami, Mehmet Karaul, Thomas F. La Porta
Bell Laboratories, Lucent Technologies
{lennox,kmurakami,karaul,tlp}@bell-labs.com

Abstract— **Internet telephony and mobile telephony are both growing very rapidly. Directly interworking the two presents significant advantages over connecting them through an intermediate PSTN link. We propose three novel schemes for the most complex aspect of the interworking: call delivery from an Internet telephony (SIP) terminal to a mobile telephony (GSM) terminal. We then evaluate the proposals both qualitatively and quantitatively. We also describe our implementation of one of the proposals on the Bell Labs RIMA platform.**

Keywords— **Internet telephony, mobile telephony, SIP, GSM, mobility, interworking.**

I. Introduction

TWO of the fastest growing areas of telecommunications are wireless mobile telephony and Internet telephony. Second and third-generation digital systems such as the Global System for Mobile communications (GSM) [1], the Universal Mobile Telecommunications System (UMTS) [2], and wideband CDMA [3] are bringing new levels of performance and capabilities to mobile communications. Meanwhile, both the Internet Engineering Task Force's Session Initiation Protocol (SIP) [4] and the International Telecommunications Union's H.323 [5] enable voice and multimedia telephone calls to be transported over an Internet Protocol (IP) network. Subscribers to each of these networks need to be able to contact subscribers on the other. There is, therefore, a need to interconnect the two networks, allowing calls to be placed between them.

Some research has been performed investigating various aspects of interworking mobile communication systems with IP-based systems. The iGSM system [6] allows an H.323 terminal to appear to the GSM network as a standard GSM terminal, so that a GSM subscriber can have his or her calls temporarily delivered to an H.323 terminal rather than a mobile device. Several papers [7], [8], [9] describe a system for interworking GSM's in-call handover procedures with H.323. However, neither of these approaches solves the general interworking question: what is the best way for calls to be delivered and routed between the two networks?

As both mobile and Internet telephony are already designed to interconnect with the Public Switched Telephone Network (PSTN), the easiest way to interconnect them would be simply to use the PSTN as an intermediate link. This is, however, inefficient and suboptimal, as compared to connecting the networks by interworking the protocols directly, for a number of reasons.

First of all, routing calls via the PSTN can result in inefficient establishment of voice circuits. This is a common problem in circuit-switched wireless systems called "triangular routing," as illustrated in Figure 1. Because a caller's local switch does not have sufficient information to determine a mobile's correct current location, the signalling must travel to an intermediate switch which can locate the subscriber correctly.[1] This intermediate switch can be far away from the caller and the destination even if the two are located in a geographically close area. Since voice circuits are established at the same time as the call signalling message is routed, the voice traffic could be transported over a long, inefficient route.

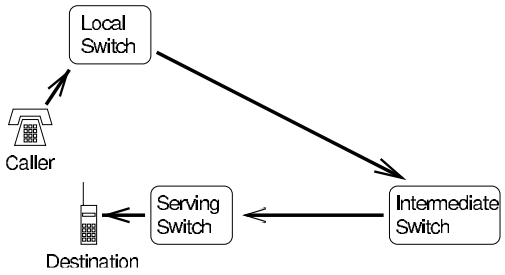


Fig. 1. Illustration of triangular routing in mobile networks

In Internet telephony, by contrast, the path of a call's media (its voice traffic, or other multimedia formats) is independent of the signalling path. Therefore, even if signalling takes a triangular route, the media travels directly between the devices which send and receive it. Since each device knows the other's Internet address, the packets making up this media stream are sent by the most efficient routes that the Internet routing protocols determine.

As we interwork Internet telephony with mobile telephony, we would like to maintain this advantage. We can accomplish this by supporting a direct IP connection between mobile base stations and IP terminals. With PSTN signalling, this is not possible, so IP telephony signalling must be used to establish this connection.

Another motivation for direct connection between mobile and Internet telephony is to eliminate unnecessary media transcod-

[1] There is an architectural difference here between the American mobile system based on ANSI 41 [10] and the European systems based on GSM MAP. In the American system, calls are always routed through a home mobile switching center, which is in a fixed location for each subscriber, so the voice traffic for all of the subscriber's calls travels through that switch. By contrast, GSM improves on this routing by sending calls through a gateway mobile switching center, which can be located close to the originating caller. However, as discussed in [11], there are some cases, such as international calls, where an originating PSTN switch does not have enough information to conclude that a call is destined for the GSM network, and thus routes it to the subscriber's home country. Because there is no way for circuit paths to be changed once they have been established, the call's voice traffic travels first to the user's home country and only then to his or her current location.

ing. The Real-Time Transport Protocol (RTP) [12], the media transport protocol common to both H.323 and SIP, can transport almost any publicly-defined media encoding [13]. Most notably, the GSM 06.10 encoding [14] is implemented by many clients. If a GSM mobile device talks to an RTP-capable Internet telephone with an intermediate PSTN leg, the media channel would have to be converted from GSM 06.10 over the air, to uncompressed (μ-law or a-law) audio over a PSTN trunk, and then again (likely) to some compressed format over the RTP media channel. The degradation of sound quality from multiple codecs in tandem is well known, and multiple conversions induce unnecessary computation. A direct media channel between a base station and an IP endpoint allows, by contrast, communication directly using the GSM 06.10 encoding without any intermediate transcodings.

Finally, on a broader scale, an integrated architecture supporting Internet and mobile telephony will evolve naturally with the expected telecommunications architectures of the future. Third-generation wireless protocols will support wireless Internet access from mobile devices. New architectures such as RIMA [15] for Mobile Switching Centers (MSCs) are using IP-based networks for communications between MSCs and base stations. In the fixed network, meanwhile, IP telephony is increasingly becoming the long-haul transport of choice even for calls that originate in the PSTN. The direct connection between Internet telephony and mobile networks takes advantage of all these changes in architecture and allows us to build on them for the future.

In this paper, we will consider the issue of how to interwork Internet telephony and mobile telecommunications, such that all the issues discussed above are resolved. For concreteness, we will illustrate our architecture using SIP for Internet telephony and GSM for mobile telephony.

The rest of the paper is structured as follows. Section II gives an architectural background on the mobility and call delivery mechanisms of GSM and SIP, to provide a basis for the following discussions. Section III proposes three different approaches to interworking GSM and SIP. Section IV provides mathematical and numerical analyses of the three proposals. In Section V, we discuss our implementation, and we finish with some conclusions in Section VI.

II. Background

In this section we review the mobility and call delivery mechanisms of GSM and of SIP.

GSM Mobility and Call Delivery

Some of the elements of a GSM network are illustrated in Figure 2. The MSC is a switching and control system in a wireless network. The MSC controlling the service area where a mobile is currently located is called its serving MSC. It routes calls to and from all the mobile devices within a certain serving area, and maintains call state for them. Associated with the serving MSC is a Visitor Location Register (VLR), a database which stores information about mobile devices in its serving area. (For the purposes of this paper we assume the predominant configuration in which the serving MSC and VLR are co-located.) Elsewhere in the fixed network we can find two other classes of entities. A Home Location Register (HLR) maintains profile information about a subscriber and keeps track of his or her current location. A gateway MSC directs calls from the PSTN into the mobile access network.

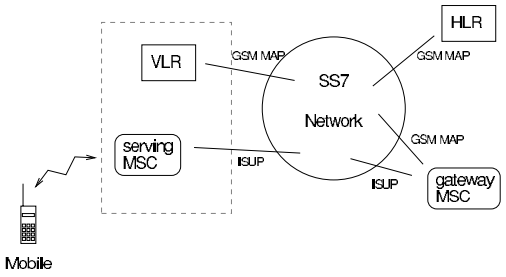


Fig. 2. Elements of a GSM Network

When a GSM mobile device first powers up or enters the serving area of a new serving MSC, it transmits a unique identification code, its International Mobile Subscriber Identity (IMSI) to the MSC. From the IMSI, the serving MSC determines the mobile's HLR and informs this HLR of the mobile's current location using the GSM Mobile Application Part (GSM MAP) protocol. The HLR stores this information and responds with profile data for the subscriber.

When a call is placed to a mobile subscriber, the public telephone network determines from the telephone number called (the Mobile Station ISDN number, or MSISDN) that the call is destined for a mobile telephone. The call is then directed to an appropriate gateway MSC. Call delivery from the gateway MSC is performed in two phases. In the first phase, the gateway MSC obtains a temporary routing number called a Mobile Station Routing Number (MSRN) in order to route the call to the serving MSC. For this purpose, the gateway MSC first locates the subscriber's HLR based on the MSISDN and requests routing information from it using GSM MAP. The HLR then contacts the VLR at the serving MSC. The VLR returns an MSRN that the HLR forwards to the gateway MSC. In the second phase, the gateway MSC routes the call to the serving MSC using the standard ISDN User Part (ISUP) protocol of the PSTN.

The MSRN is a temporarily assigned number which is allocated at the time the HLR contacts the VLR; it is valid only until the associated call is set up, and it is then recycled. This dynamic allocation of an MSRN is required because ISUP messages can only be directed to standard telephone numbers, and the quantity of these that can be allocated to a given serving MSC is limited. This has some costs, however, in the time needed to set up a call, as the serving MSC must be contacted twice during call setup.

When a subscriber moves from one location to another while a call is in progress, two possible scenarios result: intra-MSC or inter-MSC handovers. An intra-MSC handover occurs when a subscriber moves between the serving areas of two base stations controlled by the same serving MSC. In this case, the serving MSC simply redirects the destination of the media traffic. No signalling is necessary over the PSTN or GSM MAP. An inter-MSC handover, on the other hand, occurs when the subscriber moves from one serving MSC's area to another. The old serving

TABLE I
ANALOGOUS ENTITIES IN SIP AND GSM

GSM	SIP
HLR	Registrar
Gateway MSC	Home proxy server
Serving MSC	End system (for REGISTER)
MSISDN	User address (in INVITE)
IMSI	User address (in REGISTER)
MSRN	Device address

MSC contacts the new one in order to extend the call's media circuit over the PSTN. The old serving MSC then acts as an "anchor" for both signalling and voice traffic for the duration of the call.

All of the globally-significant numbers used by the GSM system — in particular, for the purposes of this paper, the MSRN, and the identifying number of the MSCs, in addition to the MSISDN — have the form of standard E.164 [16] international telephone numbers. Therefore they can be used to route requests in Signalling System no. 7 (SS7), the telephone system's signalling transport network.

SIP Mobility and Call Delivery

Architecturally, a pure SIP network (illustrated in Figure 3) is rather simpler than a GSM network, as it is significantly more homogeneous and much of the work takes place at the network layer, not the application layer. All devices communicate using IP, and all signalling occurs with SIP. Although many of the specific details are different, mobility in a SIP environment is conceptually similar to that of GSM. Table I lists some analogous entities in GSM and SIP networks.

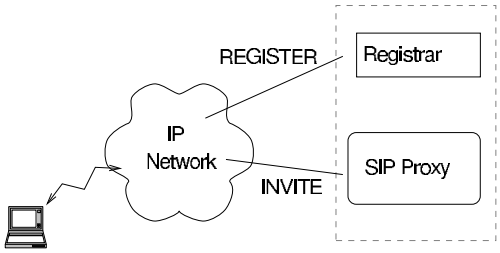


Fig. 3. Elements of a SIP network

There are two significant architectural differences between mobility in SIP and GSM. First of all, a SIP network does not have an intermediate device analogous to the serving MSC. Instead, end systems contact their registrars directly. Second, in SIP a two-phase process is not needed to contact the device during call establishment.

When a SIP subscriber becomes reachable at a new network address (either because she is using a new network device or because her device has obtained a new IP address through a mobility mechanism), the SIP device sends a SIP REGISTER to the user's registrar to inform it of the new contact location. This registration is then valid for only a limited period of time. Because end systems are assumed not to be totally reliable, registration information must be refreshed periodically (typically, once per hour) to ensure that a device has not disappeared before it could successfully de-register itself.

Unlike systems that use traditional telephone-network numbering plans, addresses in SIP are based on a "user@domain" format, similar to that of e-mail addresses. Any domain can, therefore, freely create an essentially unlimited number of addresses for itself. For the purposes of this discussion, it is useful to consider two types of addresses — "user addresses," analogous to an MSISDN number, to which external calls are placed, and "device addresses," roughly comparable to a non-transient MSRN. A device can create a temporary address for itself and have it persist for any period it wishes.

When a SIP call is placed to a subscriber's user address, a SIP INVITE message is directed to a proxy server in the domain serving this address. The proxy server consults the recipient's registrar and obtains his or her current device address. The proxy server then forwards the INVITE message directly to the device. Because the device address is not transient, the two-stage process used by GSM is not necessary. Once the call is established, media flows directly between the endpoints of the call, independently of the path the signalling has taken.

Though not explicitly defined as part of the basic SIP specification, in-call handover mobility is also possible within SIP. A mechanism for an environment based entirely on SIP, with mobile devices which have an Internet presence, is described in [17]. This mechanism does not use Mobile IP, as it suffers from a similar triangular routing issue as does circuit switching, and its handovers can be slow. Instead, it exploits SIP's in-call media renegotiation capabilities to alter the Internet address to which media is sent, once a device obtains a new visiting address through the standard mobile IP means. Therefore, Internet telephony calls can send their media streams to mobile devices' visiting addresses directly, rather than forcing them to be sent to the home addresses and then relayed by a home agent as in mobile IP.

III. ARCHITECTURE

In this section we describe our proposals for interworking SIP and GSM networks. In our design GSM mobile devices and their air interfaces and protocols are assumed to be unmodified. They use standard GSM access signalling protocols and GSM 06.10 media atop the standard underlying framing and radio protocols. Some GSM entities within the fixed part of the network, however, are upgraded to have Internet presences in addition to their standard GSM MAP and ISUP interfaces. Serving MSCs send and receive RTP packets and SIP signalling. In some of the proposals other GSM fixed entities, such as HLRs, have Internet presences as well. These entities still communicate with each other using GSM MAP and other SS7 signalling protocols, however.[2]

There are three primary issues to consider when addressing this interworking: how calls may be placed from SIP to GSM,

[2]It is possible that this SS7 signalling itself takes place over an IP network, using mechanisms such as the Stream Control Transmission Protocol (SCTP) [18], currently in development.

how they may be placed from GSM to SIP, and how in-call mobility (handovers) are handled. The second and third of these points are relatively straightforward, and we will address them first. The first one is more challenging and represents the main focus of this paper.

SIP/GSM Interworking: Calls from GSM to SIP

Calls originating from a GSM device and directed at a SIP subscriber are not, in principle, different from calls from the PSTN to a SIP subscriber. The primary issue when placing calls from a traditional telephone network to SIP is that traditional telephones can typically only dial telephone numbers, whereas SIP addresses are of a more general form, based roughly on e-mail addresses, which cannot be dialed on a keypad. Work is ongoing to resolve this problem, but the currently envisioned solution is to use a distributed database based atop the domain name system, known as "Enum," [19] which can take an E.164 international telephone address and return a SIP universal resource locator. For example, the E.164 number +1 732 332 6063 could be resolved to the SIP URI 'sip:lennox@bell-labs.com'.

Since globally significant GSM numbers take the form of E.164 numbers, several of the proposals below use Enum-style globally distributed databases in order to locate Internet servers corresponding to these addresses. However, for such databases it would not be desirable to use the actual global Enum domain, for security reasons.

SIP/GSM Interworking: In-Call Handover

As explained earlier, there are two categories of in-call handover: intra-MSC and inter-MSC. Intra-MSC handover does not need to be treated specially for SIP-GSM interworking. Because this happens between the serving MSC and the base stations, the network beyond the serving MSC is not affected. As an optimization, however, a serving MSC could use different IP addresses corresponding to different base stations under its control. In this case, a mechanism for SIP mobility as described before could be used to change the media endpoint address in mid-call.

Inter-MSC handover does affect SIP-GSM interworking, and remains for future study. We anticipate that a mechanism similar to that of [9], as described in the introduction, could be adapted to SIP for this purpose.

SIP/GSM Interworking: Mobile-Terminated Calls

The most complex point of SIP/GSM interworking is the means by which a SIP call can be placed to a GSM device. As discussed in the introduction, it is desirable to set up media streams directly between the calling party and the serving MSC. In order to accomplish this, SIP signalling must travel all the way to the serving MSC, as only the serving MSC will know the necessary IP address, port assignment conventions, and media characteristics.

We propose three methods as to how SIP devices can determine the current MSC at which a GSM device is registered. These have various trade-offs in terms of complexity, amount of signalling traffic, and call setup delay.

Proposal 1: modified registration

Our first proposal is to enhance a serving MSC's registration behavior. The basic idea is that a serving MSC registers not only with the subscriber's HLR, but also with a "Home SIP Registrar." This registrar maintains mobile location information for SIP calls.

The principal complexity with this technique lies in how the serving MSC locates the SIP registrar. Our proposal, illustrated in Figure 4, is to use a variant of the Enum database described above. Once the serving MSC has performed a GSM registration for a mobile device, it knows the mobile's MSISDN number. From this information, an Enum database is consulted to determine the address of the device's home SIP registrar, and the serving MSC performs a standard SIP registration on behalf of the device. A SIP call placed to the device then uses standard SIP procedures.

Because of authentication needs, this proposal uses either eight or ten GSM MAP messages (depending on whether authentication keys are still valid at the VLR) and six DNS messages per initial registration, and four SIP messages per initial or refreshed registration. Call setup requires a single SIP message and four DNS messages, though some DNS queries may be cached.

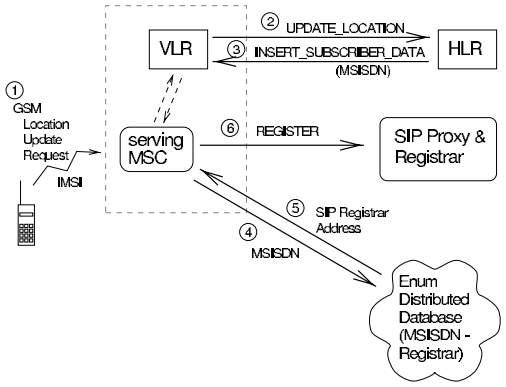


Fig. 4. Registration procedure for proposal 1

Compared to our other proposals, this proposal has two primary advantages. First, the only changes to the existing infrastructure are the modifications in the serving MSC and the addition of a variant Enum database to find registrars. Neither the SIP registrar and proxy server, nor the GSM HLR and gateway MSC, need to be altered. Second, because the complexity of the proposal occurs only in registration, call setup shares the single-lookup efficiency of SIP and is therefore relatively fast.

The disadvantages of this proposal, however, also arise due to the separation of the two registration databases. First, once a system requires the maintenance of two separate databases with rather incomparable data, the possibility arises that the information in the databases becomes inconsistent due to errors or partial system failure. This is especially true because of the differing semantics of SIP and GSM registrations — GSM registra-

tions persist until explicitly removed, whereas SIP registrations have a timeout period and must be refreshed by the registering entity. Furthermore, when mobility rates are low, the dual registration procedure imposes significantly more signalling overhead than GSM registration alone, since SIP registrations must be refreshed frequently.

Proposal 2: modified call setup

By contrast, our second proposal does not modify the GSM registration procedure. Instead, it adds complexity to the call setup procedure. Essentially it adapts the GSM call setup to SIP. This is illustrated in Figure 5. When a SIP call is placed to a GSM user, the user's home SIP proxy server determines the MSISDN corresponding to the SIP user address, and queries the GSM HLR for an MSRN. The HLR obtains this through the normal GSM procedure of requesting it from the serving MSC's VLR. The SIP proxy server then performs an Enum lookup on this MSRN, and obtains a SIP address at the serving MSC to which the SIP INVITE message is then sent.

This approach uses either eight or ten MAP messages, as with standard GSM, for registration, and four MAP messages, six DNS messages, and one SIP message for a call setup.

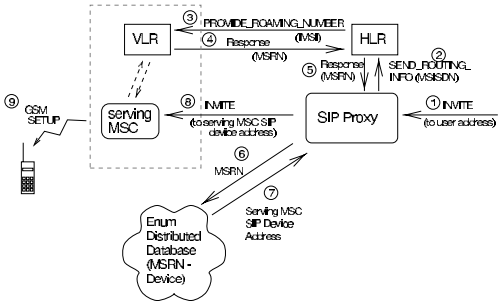


Fig. 5. Call setup procedure for proposal 2

Because this proposal does not modify the GSM registration database, it has several advantages over the previous proposal. Specifically, there is no possibility for data to become inconsistent, and the overhead of registration is as low as it is for standard GSM. However, both the signalling load and the call setup delay are high, as call setup now involves a *triple*-phase query: a GSM MAP query for the MSRN, an Enum lookup for the SIP device address, and finally the actual call initiation. Additionally, we have a new requirement that the SIP proxy server and the HLR need to be able to communicate with each other. This imposes additional complexity in both these devices, as it requires new protocols or interfaces.

Proposal 3: modified HLR

Our final proposal is to modify the GSM HLR. In this proposal, the serving MSC registers the mobile at the HLR through standard GSM means. The HLR then has the responsibility to determe the mobile's SIP device address at the serving MSC.

The overall registration procedure for this proposal is illustrated in Figure 6. When a serving MSC communicates with an HLR, the HLR is informed of the serving MSC's address, which, as mentioned earlier, is an E.164 number. The HLR performs a query to a specialized Enum database to obtain the name of the serving MSC's SIP domain, based on the serving MSC's address. While the previous two proposals treat the SIP device address as an opaque unit of information whose structure is known only to the serving MSC, this proposal takes advantage of its structure.

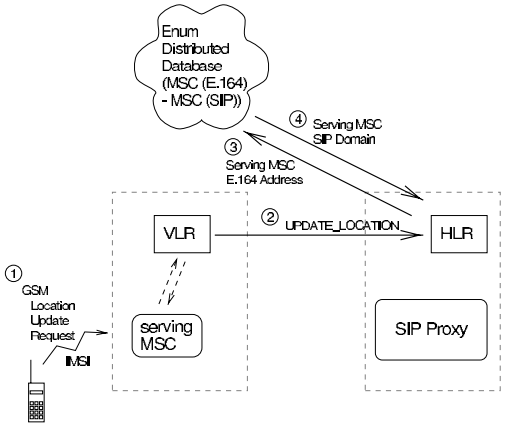


Fig. 6. Registration procedure for proposal 3

Figure 7 shows how a SIP call is placed. The SIP proxy server queries the HLR for a SIP address and the HLR returns an address of the form "MSISDN@hostname.of.serving.MSC" to which the SIP proxy then sends the call. This proposal uses either eight or ten MAP messages, and two DNS messages, for registration, and four DNS messages and one SIP message for call setup.

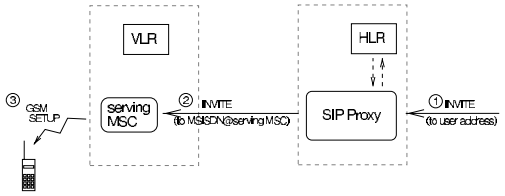


Fig. 7. Call setup procedure for proposal 3

This approach has the advantage that its overhead is relatively low for registration and quite low for call setup. The time requirements for call setup are similarly low. It does, however, require invasive modifications of HLRs. Additionally, the SIP proxy server and the HLR must be co-located, or else they must also have a protocol defined to interface them.

IV. Analysis

Two important criteria for evaluating the signalling performance of these three proposals for interworking SIP and GSM

TABLE II
MESSAGE WEIGHTS

Symbol	Parameter	Value
w_{sip}	Weight of a SIP message	1.0
w_{dns}	Weight of a DNS message	0.5
w_{map}	Weight of a MAP message	1.5

TABLE III
MOBILITY PARAMETERS

Symbol	Parameter	Value
$r_{\text{in}}, r_{\text{out}}$	Rate of call delivery / origination	variable
r_{bc}	Average boundary crossing rate	variable
$P_l(l)$	Boundary crossing rate prob. distribution ($P(l_0 \geq l)$)	$e^{-r_{\text{bc}}l}$
s	Call / mobility ratio	$\frac{r_{\text{out}}+r_{\text{in}}}{r_{\text{bc}}}$
P_{nr}	Prob. that a device is new to a serving MSC	50%
P_{ur}	Prob. that a device has a unique registrar at its serving MSC	20%
P_{us}	Prob. that a device has a unique serving MSC at its HLR/registrar	20%

TABLE IV
PROTOCOL PARAMETERS

Symbol	Parameter	Value
l_{sip}	SIP registration refresh interval	3 hr
l_{dns}	DNS cache time-to-live	24 hr
c_{auth}	Number of pieces of authentication data cached at VLR	5

TABLE V
WEIGHTED PACKET COUNTS FOR EACH PROPOSAL

Case	Formula
Modified Registration	
Registration	$r_{\text{bc}}\big((8+2/c_{\text{auth}})\,w_{\text{map}}+ (2P_{\text{nr}}+4P_{\text{ur}})\,w_{\text{dns}}+ 4\,(1+\sum_{i=1}^{\infty}P_l(il_{\text{sip}}))\,w_{\text{sip}}\big)$
Call setup	$r_{\text{in}}\,(4P_{\text{us}}w_{\text{dns}}+1w_{\text{sip}})$
Modified Call Setup	
Registration	$r_{\text{bc}}\,(8+2/c_{\text{auth}})\,w_{\text{map}}$
Call setup	$r_{\text{in}}\,(4w_{\text{map}}+6P_{\text{us}}w_{\text{dns}}+1w_{\text{sip}})$
Modified HLR	
Registration	$r_{\text{bc}}\big((8+2/c_{\text{auth}})\,w_{\text{map}} +2P_{\text{us}}w_{\text{dns}}\big)$
Call setup	$r_{\text{in}}\,(4P_{\text{us}}w_{\text{dns}}+1w_{\text{sip}})$

are signalling load and call setup delay. A detailed study of call setup delay remains for future investigation. In this paper we focus on performance in terms of signalling load.

Each of the proposals involves the use of several different protocols, in varying ratios. In order to compare total signalling load imposed by each protocol, we assigned signalling messages of each protocol a weight. The default values of these weights are listed in Table II. We discuss the effect of these weights on the total signalling load in our sensitivity analysis later in this section.

Tables III and IV list the parameters for our model. We assume equal rates of call delivery r_{in} and r_{out}, as is commonly observed in European settings. We assign an exponential distribution to the probability $P_l(l)$ that a mobile remains in a particular MSC's serving area for longer than time l. DNS caching was accounted for by assigning the probabilities P_{nr}, P_{ur}, and P_{ns} to the likelihood that particular DNS queries have been performed recently, within the DNS time-to-live period.

Table V shows the equations for the weighted signalling loads for registration and call establishment in each proposal. These equations are based on the packet counts for each proposal in Section III.

Figure 8 graphs the total weighted signalling load (registration plus call setup costs) for each of the three proposals, as both the incoming call rate and the call / mobility ratio vary. The intersection line at which modified registration and modified call setup are equal is shown in bold.

From this graph, we can observe some general characteristics of the proposals' signalling load. First, the modified HLR proposal consistently has the lowest signalling load of the three, typically 20 – 30% less than the others. This corresponds to intuition, as it combines the "best" aspects of each of the other two proposals, unifying both an efficient registration and efficient call setup procedure.

Second, the relative signalling loads for the other two proposals depend on the values of the traffic parameters. Modified call setup is more efficient for a low incoming call rate or a low call / mobility ratio (i.e., fast mobility), while modified registration is more efficient when both parameters are high. A closer look at the equations in Table V reveals the reasons. Consider the relative efficiency of the two approaches for varying incoming call rates: modified call setup performs less well for high incoming call rates because its call setup procedure requires four additional GSM MAP messages and possibly two additional DNS messages compared to that of modified registration. Similarly,

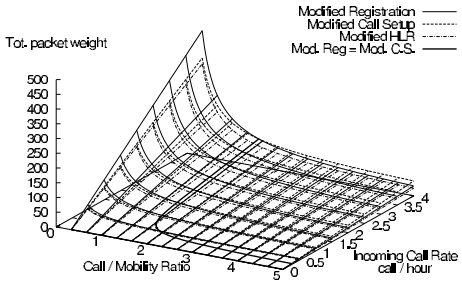


Fig. 8. Weighted signalling load of the three proposals

modified call setup outperforms modified registration for low call / mobility ratios because the latter has higher registration message overhead due to dual registration and SIP registration soft-state.

In order to increase the confidence in the above results, we performed sensitivity analyses to validate our choice of various parameters.

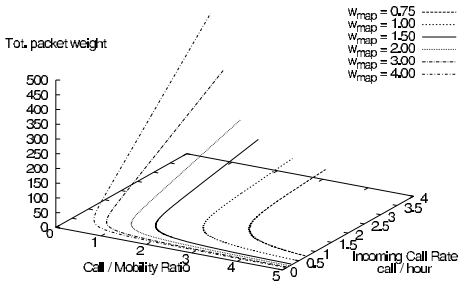


Fig. 9. Line of Intersection: Mod. C.S. = Mod. Reg. (w_{map} varying)

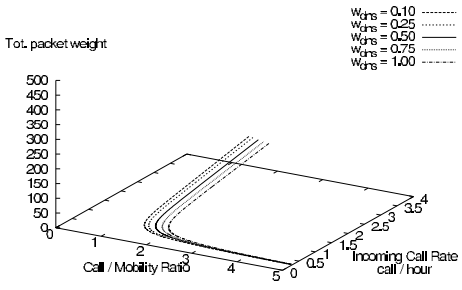


Fig. 10. Line of Intersection: Mod. C.S. = Mod. Reg. (w_{dns} varying)

Sensitivity analyses for the weights assigned to MAP and DNS messages are shown in Figures 9 and 10, respectively. These graphs illustrate how, as the protocol weighting changes, the position of the intersection line in Figure 8 changes.

Figure 9 shows that as the weight assigned to the MAP protocol increases, the area in which modified registration is more efficient — the right-hand side of the graph, where call rate and call/mobility ratio are both high — increases as well. This fits with the intuitive understanding of the approaches, as modified registration uses fewer MAP messages than modified call setup. Similarly, Figure 10 shows that as the weight assigned to the DNS protocol increases, the area in which modified registration is more efficient shrinks slightly. This also fits with intuition, as modified registration uses more DNS packets. However, the total packet load is generally less sensitive to the weight assigned to DNS messages, which explains why the lines in Figure 10 are relatively close to each other.

The signalling load of the modified HLR proposal is always less than the other two. Thus, it is not shown in our sensitivity graphs. In regards to the other two protocols, though the crossover point moves as the weights assigned to the protocols vary, these sensitivity analyses show that the general shape of the graph, and therefore the conclusions we draw from it, do not change.

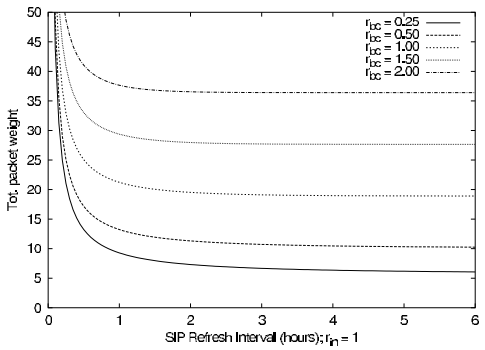


Fig. 11. Total weight of modified registration

Figure 11 shows the effect of various choices of values for the SIP registration timeout period. (This value only affects the modified registration proposal, as the other proposals do not use SIP registration.) The value for this parameter should be chosen so that the additional cost of SIP registration is relatively minor, that is, so that the graph has roughly flattened out. This optimal value therefore depends on the boundary crossing rate, but generally, a timeout of three hours is a good choice for most reasonable boundary crossing rates. This value can be larger than the standard value of one hour used by SIP, as serving MSCs can be assumed to be more reliable and available than regular SIP end systems.

V. Implementation

To prove the feasibility of our proposal, we implemented the modified call setup scheme atop the Enhanced Mobile Call Processing (EMCP) component of the Bell Labs Router for Integrated Mobile Access (RIMA) [15]. Figure 12 illustrates the overall architecture of this system. The modified call setup scheme was selected partly because it appears to be more applicable than modified registration scheme in the future mobile networks where a higher mobility rate is expected. It also requires substantially less modification to GSM equipment than the modified HLR scheme.

As opposed to traditional MSCs, RIMA is inherently IP based and uses packet networks for both transport and signalling. It is built on top of an IP router based network and is composed of a cluster of commodity processors and various gateways performing media conversion and transcoding. It supports standard circuit voice for wireless terminals like GSM phones and connects

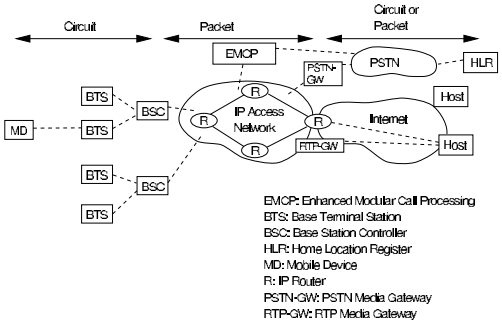


Fig. 12. RIMA-based Network

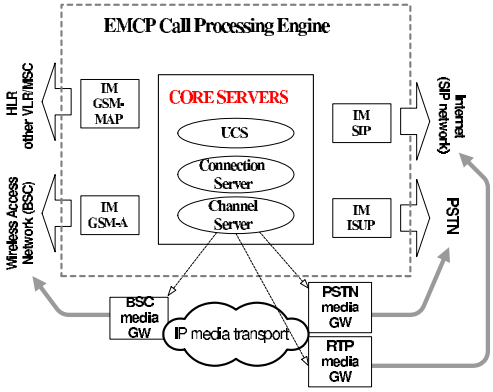


Fig. 13. Structure of EMCP Call Processing Engine

to existing circuit networks like the PSTN. It was designed with the idea in mind of connecting to packet voice networks like the Internet.

RIMA provides wireless access to mobile users through a packet based wireless access network. A RIMA network has four major components: a Base Station Controller (BSC), a PSTN media gateway (PSTN-GW), an RTP media gateway (RTP-GW), and the EMCP call processing engine, connected *via* an IP network.

Each BSC has an IP interface and translates voice and signalling information between circuit and packet format. It serves as a media gateway translating between circuit voice and RTP/IP packet voice. With respect to signalling, it terminates the standard GSM interface towards mobile devices to accommodate existing radio networks and tunnels these signals in IP packets on the RIMA wireless access packet network.

A PSTN-GW performs media conversion between RTP/IP packet voice in the RIMA access network and circuit voice over the PSTN. It is controlled by the call processing engine, and it may perform possible transcodings between different coding schemes such as compressed wireless (e.g. GSM speech) and PCM (e.g. (μ-law).

We added the RTP-GW to provide RIMA with media connections to the Internet. Though the RIMA access network uses RTP internally, it was useful to centralize advanced functionality such as buffering, jitter adaptation, and handling of the Real-Time Control Protocol (RTCP) into a single location. In this way, other RIMA entities do not need to support the entire suite of complex RTP behavior. The RTP-GW also performs transcoding between coding schemes as necessary, if for example a remote SIP endpoint does not indicate support for GSM encoding but wishes only to send and receive PCM. We implemented this gateway using the Bell Labs RTPlib [20] library, which we ported to the same single-board computers as the PSTN-GW.

RIMA's MSC and VLR functionality is realized by the EMCP call processing engine, whose structure is shown in Figure 13. It is deployed on a cluster of commodity processors such as workstations or single board computers. The engine is separated from the IP media transport network and can be viewed as a signalling gateway by IP telephony networks. It consists of a collection of functionally distributed servers. Call processing and mobility management tasks are accomplished by their collaboration.

The call processing engine is comprised of two server classes: core servers and interworking managers (IMs). Core servers perform call processing and mobility management tasks common to any wireless system. Interworking managers act as protocol gateways to internal core servers, isolating them from external signalling protocols thereby allowing the core servers to evolve independently of these protocols.

There are three core servers: a channel server, a connection server, and a user call server (UCS). The channel server manages switching device resources, such as transport channels and DSPs for vocoding, allocated during call setup and deallocated during call release. The connection server coordinates the allocation of channel resources to establish an end-to-end connection. The UCS maintains information on the registration status of mobile devices currently located within the service area of the RIMA system and records call activities involving a particular mobile device. The UCS also handles other mobility management tasks such as paging, handover, mobile user authentication, and ciphering.

Interworking managers allow core servers to accommodate different sets of standard interfaces. As originally developed, EMCP has interworking managers supporting the GSM A standard protocol between an MSC and a BSC (IM-GSM-A), GSM MAP to the HLRs (IM-GSM-MAP), and ISUP to the PSTN (IM-ISUP). To realize the architecture described in this paper, we added a new interworking manager, IM-SIP, which supports SIP towards the Internet. Implementing this IM was straightforward. Due to the modularity of the EMCP architecture, IM-SIP could use the same interfaces as IM-ISUP. Because we chose the modified call setup model, we did not have to alter EMCP's registration procedures.

For the Home SIP Proxy, we extended an experimental Bell Labs SIP proxy server and registrar to allow it to communicate with an HLR. This proxy server was programmed to recognize that certain blocks of addresses corresponded to GSM users. For these numbers it invokes a special procedure in which it asks

the HLR for an MSRN. Because Enum has not yet been standardized, we instead used a table lookup to find SIP addresses corresponding to the MSRN returned.

VI. Conclusion

We proposed three novel schemes to directly interconnect GSM mobile and SIP Internet telephony systems. Compared with the conventional approach of routing a call through PSTN, direct interconnection prevents triangular routing and eliminates unnecessary transcodings along its path. We analyzed the signalling message load of three proposals under a wide range of call and mobility conditions. The modified HLR scheme always imposes less signalling burden, typically 20-30% less than the other schemes, although it requires significantly greater modification to GSM equipment. The efficiency of the other two proposals, modified registration and modified call setup, depends on the traffic parameters. When the incoming call rate and call / mobility ratio are both high, modified registration is more efficient. Modified call setup performs better otherwise. We further demonstrated our implementation of one of the proposed schemes, modified call setup, in the Bell Labs next generation wireless access system RIMA.

References

[1] European Telecommunications Standards Institute, "Digital cellular telecommunications system, network architecture," GSM 03.02 version 7.1.0 release 1998, European Telecommunications Standards Institute, Sophia Antipolis, France, Feb. 2000.

[2] European Telecommunications Standards Institute, "Universal mobile telecommunications system (UMTS), general UMTS architecture," 3G TS 23.101 version 3.0.1 release 1999, European Telecommunications Standards Institute, Sophia Antipolis, France, Jan. 2000.

[3] Bijan Jabbari, Ed., "Special issue on wideband CDMA," *IEEE Communications Magazine*, vol. 36, no. 9, Sept. 1998.

[4] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: session initiation protocol," Request for Comments 2543, Internet Engineering Task Force, Mar. 1999.

[5] International Telecommunication Union, "Packet based multimedia communication systems," Recommendation H.323, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Feb. 1998.

[6] Herman C. H. Rao, Yi-Bing Lin, and Sheng-Lin Cho, "iGSM: VoIP service for mobile networks," *IEEE Communications Magazine*, vol. 38, no. 4, pp. 62–69, Apr. 2000.

[7] Wanjiun Liao, "Mobile internet telephony: Mobile extensions to H.323," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, Mar. 1999.

[8] Wanjiun Liao, "Mobile internet telephony protocol: An application layer protocol for mobile internet telephony services," in *Conference Record of the International Conference on Communications (ICC)*, Vancouver, British Columbia, June 1999.

[9] Wanjiun Liao and Jen-Chi Liu, "VoIP mobility in IP/cellular network internetworking," *IEEE Communications Magazine*, vol. 38, no. 4, pp. 70–75, Apr. 2000.

[10] Telecommunications Industry Association and Electronics Industry Association, "Cellular radiotelecommunications intersystem operations," TIA/EIA ANSI-41-D, Telecommunications Industry Association, Arlington, Virginia, Dec. 1997.

[11] Yung-Jan Cho, Yi-Bing Lin, and Herman Chung-Hwa Rao, "Reducing the network cost of call delivery to GSM roamers," *IEEE Network*, vol. 11, no. 5, pp. 19–25, Sept. 1997.

[12] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications," Request for Comments 1889, Internet Engineering Task Force, Jan. 1996.

[13] H. Schulzrinne, "RTP profile for audio and video conferences with minimal control," Request for Comments 1890, Internet Engineering Task Force, Jan. 1996.

[14] European Telecommunications Standards Institute, "Digital cellular telecommunications system, full rate speech," GSM 06.10 version 5.0.1, European Telecommunications Standards Institute, Sophia Antipolis, France, May 1997.

[15] Thomas F. La Porta, Kazutaka Murakami, and Ramachandran Ramjee, "RIMA: router for integrated mobile access," in *Proceedings of the 11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communication (PIMRC)*, London, United Kingdom, Sept. 2000, to appear.

[16] International Telecommunication Union, "The international public telecommunication numbering plan," Recommendation E.164, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, May 1997.

[17] Elin Wedlund and Henning Schulzrinne, "Mobility support using SIP," in *Second ACM/IEEE International Conference on Wireless and Mobile Multimedia (WoWMoM'99)*, Seattle, Washington, Aug. 1999.

[18] R. R. Stewart et al., "Stream control transmission protocol," Internet Draft, Internet Engineering Task Force, June 2000, Work in progress.

[19] P. Faltstrom, "E.164 number and DNS," Internet Draft, Internet Engineering Task Force, May 2000, Work in progress.

[20] Henning Schulzrinne, Jonathan Lennox, Daniel Rubenstein, and Jonathan Rosenberg, "RTPlib: Bell Labs RTP library," Available from http://www.bell-labs.com/topic/swdist/.

An Open Source H.323-SIP Gateway as Basis for Supplementary Service Interworking

R. Ackermann[1], V. Darlagiannis[1], M. Görtz[1], M. Karsten[1], R. Steinmetz[1,2]
1 - Darmstadt University of Technology
Industrial Process and System Communications (KOM)
2 - German National Research Center for
Information Technology (GMD IPSI)
**{Ralf.Ackermann, Vasilios.Darlagiannis, Manuel.Goertz,
Martin.Karsten, Ralf.Steinmetz}@KOM.tu-darmstadt.de**

Abstract— **IP telephony is currently evolving from a more or less still experimental towards a carrier grade service which has the potential of extensive use both within the Internet as well as in Intranets. Currently we see the two signaling protocol families H.323 and SIP existing and further evolving simultaneously. For both, efforts are done to not only establish basic calls but to enable so called Supplementary Services. This is generally considered one precondition for replacing the functionality of existing conventional PBXs on top of a standard protocol. Nevertheless solutions that support more then just basic call scenarios are at the moment still often based on proprietary protocols or protocol extensions.**

Since we assume that both H.323 and SIP are going to coexist for a longer future period, gateways between both protocol families are of large interest and research, standardization and development activities have been spent on those.

As part of an industry research cooperation we have (independently from other efforts) developed and deployed a fully Open Source H.323/SIP gateway. The paper shows its concepts and describes its use as a powerful reference implementation basis for further development targeting at the mapping and gatewaying of Supplementary Services. Our gateway's modular, flexible and extensible architecture as well as the usage of scripting functionality both for configuration as well as internal protocol processing enables the usage of different basic software components (e.g. protocol stacks) in a fast prototyping way. We consider this especially important, since the existing freely available stacks (such as OpenH323) do not support H.450 or SIP Supplementary Services at the moment.

Keywords—**IP Telephony, SIP, H.323, Gateway, Supplementary Services, Rapid Prototyping and Testing of Services**

I. Introduction

Providing gateway functionality between different protocol stacks is a ongoing task and challenge. For making an operational prototype or even a product, designers and implementors do not only have to find and describe the mappings between the protocol primitives of both signaling "worlds" but also to establish a framework for "glueing" protocol stacks and their implementations together. In most cases this work can not start from scratch, but has to consider the existing design of the components that get connected. Thus, their implementation, interfaces and dynamic behavior may be more or less suited for usage within a common application. In general, combining two software products that have been implemented independently and without considering further combined use, is not an easy task.

In addition to basic call connectivity IP telephony has to offer services that are comparable or even more sophisticated services than those of the PSTN.

The paper is organized in the following main parts. After a short introduction of the IP signaling protocols H.323 and SIP, their specifics and implications for interworking we present the requirements for the gateway followed by a description of the architecture that we have developed and implemented. This includes a critical evaluation of the specifics of basic software components as well as the features of our system. Then we show our approach for enhancing the gateway for Supplementary Services on the example of a H.450 to SIP mapping scenario. Finally we conclude our paper and give an outlook on future tasks and activities.

II. IP Telephony signaling protocols

The two existing IP telephony signaling protocols H.323 [1] and SIP [2] are currently evolving simultaneously. Both deal with the standardized setup, communication parameter negotiation/exchange and teardown of two- or multi-party communication sessions. While H.323 has been within the focus of especially vendors with a strong PBX and classical telephony background for a longer period, SIP meanwhile gains a very high attraction both from the research community as well as from equipment and service providers. There has been a number of publications [3], [4], [5] that critically review and compare the protocols features and do mainly address complexity and communication overhead.

Especially when doing experiments on porting as well H.323 as SIP software to small end systems (such as the most recent generation of PDAs running WinCE or Linux) we have found, that the more lightweight and extendable SIP approach has a number of benefits concerning criteria such as necessary computing power and even more serious memory footprint[1].

Both protocol families agree upon the usage of RTP [6] for exchanging media and its companion RTCP for controlling and specfiy a number of well-defined audio codecs (e.g. [7], [8], [9]) for transmitting media data, but differ in the way control information is exchanged and processed within the internal protocol state machines.

Since the coexistence of both protocol families is expected for a longer future period, gatewaying between both – thus enabling interconnected end systems to use their specific signaling and protocol semantics while mapping them transparently – is a challenge for appropriate interworking units. Their concepts have been described in a number of publications [10], [11] and companies as well as number of implementors within the re-

[1]OpenH323 ohphone and necessary libraries (cross-)compiled for a Compaq iPAQ PDA (StrongARM processor, 32MB RAM, 16MB Flash) running Linux have a size of 9236292 bytes, a basic SIP UA plus RTP stack can be implemented considerably smaller

search community [12] are working on building such solutions.

III. The H.323/SIP gateway prototype

A. Starting situation

An interworking unit between the two protocols that has to meet the requirements of protocol conformity, call sequence mapping and direct RTP/RTCP media exchange between the communication endpoints (described in detail in [13], [14]) should try to use existing or evolving protocol stacks thus lowering the necessary implementation effort. Out initial requirements for an implementation are listed in Table I.

Requirement	Implications
Basic Interworking Functionality	Support for H.323- as well as SIP-originated calls from terminals / User Agents with at least minimal interoperable media encodings
Handling of different locating and addressing mechanisms	Support for co-location with H.323 gatekeepers and SIP UAs becoming "virtual" H.323 subscribers as well as for participants in "protocol clouds" with configurable way and location of address mapping
Interoperability with applications using different protocol versions and variants	Support for different interworking protocol sequences depending on what time media endpoint descriptions are available (e.g. H.323v1 vs. H323 versions with support for Fast Connect)
Scalability	Initial support for one call at a time as well as for multiple parallel calls in a full-featured version
Extensibility	Support for different protocol stacks as well as additional features (such as Supplementary Services) as they become available

TABLE I
REQUIREMENTS FOR THE GATEWAY

In order to meet those requirements in an efficient way we did an evaluation and pre-selection of H.323 and SIP base components that has been based on the following criteria:

- Availability, stability and functionality of the package (client-only vs. full-featured including server component)
- Development and runtime platform (Operating System, implementation language and portability)
- Obtainability of program source code and adaptability
- Interoperability
- Estimated complexity of direct integration with other components
- Usage Conditions (Open Source vs. Evaluation Version vs. Commercial Only Version)

After an evaluation of several software packages (additionally to the used stacks we considered the DynamicSoft SIP stack [15], the software forming the sipd package [16] and libdissipate [17]) we have chosen the SIP protocol stack provided by Vovida [18] to integrate with the OpenH323 [19] H.323 implementation. Mainly because they are both Open Source, supported by a large developer community, have evolved rather fast, do support multiple platforms and have meanwhile reached a certain level of maturity.

B. Basic gatewaying approach

Figure 1 shows the the basic concept of interworking between H.323 and SIP using our H.323/SIP-Gateway [20] as well as the interaction of the components.

The gateway component handles the mapping, forwarding and execution of the appropriate signaling messages, whereas the RTP media streams are exchanged directly between the caller and callee.

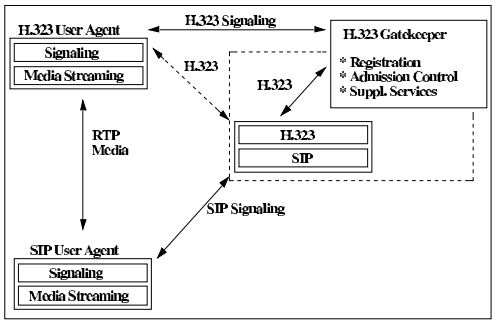


Fig. 1. Interworking between H.323 and SIP

The gateway is first of all supposed to work in a simple scenario just using directly connected H.323 terminals and SIP User Agents. To be really useful in a "real-world environment" it has to support clients attached via fully deployed "protocol clouds" (H.323 terminals using the Gatekeeper Mediated Call Model, SIP User Agents communicating via "chains" of Proxy and Redirect Servers) as well.

We now provide basic information on the building blocks of our implementation.

C. Vovida SIP Architecture

Implementations using the Vovida SIP stack (such as the SIP User Agent sua) are usually built using an internal *Finite State Machine* (FSM) that is driven by events.

The stack originates and receives messages which are forwarded via an internal FIFO. All the events driving the internal FSM are encapsulated as objects. Existing events that also carry parameter information and can thus be extended and new events can easily be added. The same applies to the additional transitions which can be notated in a standardized way. The implementations multi-threaded asynchronous message passing

approach without the necessity to explicitly call functions directly makes the approach very suitable for loosely coupled interaction and enhancement with other implementations.

D. OpenH323 Architecture

OpenH323 applications are built on top of the powerful network, interprocess communication and threading library *pwlib*. The source package also provides an ASN.1 parser, that automatically generates code for parsing and generating H.323 PDUs from their standardized ASN.1 description. The handling and processing of signaling messages is encapsulated in an object oriented way which provides simple but powerful means for extensions. The stack is organized using method calls and callbacks that are associated with the protocol and state transitions.

"Primitives" like *OnIncomingCall* or *OnOpenLogicalChannel* can easily be enhanced to add additional functionality such as the extraction and modification of source or target addresses and ports when processing the RTP endpoint descriptions in the gateway case. Enhancements benefit from the clear object oriented structure of the software as well as from the availability of not just an OpenH323 based H.323 terminal but also other components like a H.323/PSTN gateway (pstngw as part of the base package) that shows a possible interworking functionality. As well as an OpenH323 based gatekeeper [21] as a representation how to deal with requests from multiple communication partners.

E. Straight-forward Integration

The OpenH323 stack uses unique call identifiers referring to a particular call and can be enhanced in a more "close coupling" way by directly calling its methods or registering new methods to be called. We have used that approach for our very first gatewaying experiments, thus integrating the SIP stack as well as the OpenH323 part in just one common executable holding all the functionality.

F. More generalized Gateway Architecture

While the direct integration of two protocol stacks is a valuable approach for creating a first prototype, a more general design must be used for implementing a system that can fulfill its task using a stable and more or less persistent core logic, while combining it with alternative protocol stacks or newer versions of the existing ones. We had to face a rather rapid development of both OpenH323 and Vovida SIP with even changing and non backward compatible classes and interfaces during our project period. Thus a redesign of the gateway has been done, leading to a highly modular system architecture.

Alternative protocol stack implementations can now be integrated as "plug-ins", by implementing the "glueing" code to connect a new component with the well-designed interfaces of the system. Figure 2 shows the structure of the recent system, with the protocol stacks actually used in grey color and alternatives indicated by dashed lines and boxes.

In this architecture, a *Configuration Manager* provides the means to connect the gateway with the SIP and H.323 world. The *Connection Manager* handles the set of the active connections between two end-points that belong to the different signaling worlds. For each new session a *Connection* object is created,

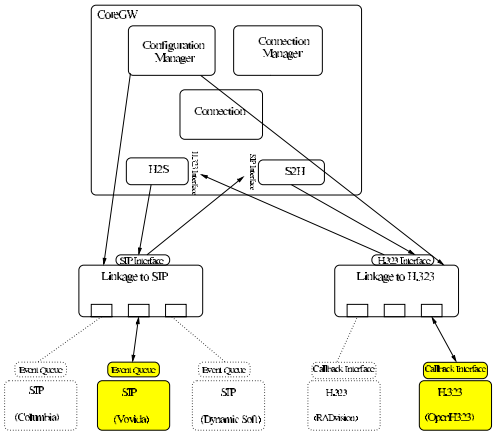


Fig. 2. SIP/H.323 Gateway Architecture

that encapsulates the objects which perform the "translation" of the parameters between the two protocols. The *H2S component* is responsible for the mapping of the H.323 messages and state transitions to the their SIP counterparts, while the *S2H component* is responsible for the reverse translation.

Two interfaces have been defined to increase the modularity of the system. The design of the *SIPInterface* and the *H323Interface* is based on the set of supported messages of each protocol. Each different message is mapped to a specific method. For example, in the SIP protocol the INVITE message is mapped in the Invite() command. Each parameter of the INVITE message is an argument in the Invite() method. The interface for the H.323 protocol has been designed similarly. Since H.323 is a more complex protocol stack, sub-interfaces can be defined for each protocol included in the suite (e.g. H.245, H.225, etc.).

The integration of the core gateway components with the implementation of the protocol stacks is supported by the abstraction of "Linkage" components. Implementation specific parts for each supported stack and their adaption are placed inside those. In our implementation, there are two specific components to integrate the system with the Vovida SIP protocol stack and the OpenH323 protocol stack. For the SIP part, the *LinkageToVovida* component inside the *LinkageToSIP* component communicates with the core Vovida SIP stack through the asynchronous FIFO that processes every event in the protocol stack.

Similarly, in order to integrate the OpenH323 protocol stack, a new module has been realized, the *LinkageToOpenH323*, that implements the callback interface of the OpenH323, as it is required from the specific implementation. All the modification and the implementation details regarding the integration with the specific protocol stacks have been kept in the "Linkage" components, leaving the rest of the system independent.

The arrows in Figure 2 visualize the dynamic behavior of the components. A new SIP message that arrives in the gateway is received from the Vovida component and through the LinkageToSIP layer, it is passed to the S2H component. S2H translates

the parameters to the H.323 equivalents and it calls the LinkageToH323 component, which implements the H323Interface. The later component is passing the information to the OpenH323 protocol stack and the new message is transmitted to addressed receiver.

G. Flexibility by means of Scripting

Within the gateway development and customization there are a number of situations where flexible means of describing the parsing, modification or generation of signaling PDUs or events and states within the system are very valuable. While this can be done using compiled code it restricts the flexibility and opportunities for rapid prototyping of new mechanisms.

Scripting languages can solve some of these tasks very well. Their drawbacks such as slower execution as well as just limited "compile-time"/static type and correctness checking can be accepted within the environment we have.

Figure 3 shows the usage for address mapping but the the approach can be used for the notation of dynamic protocol sequences as well.

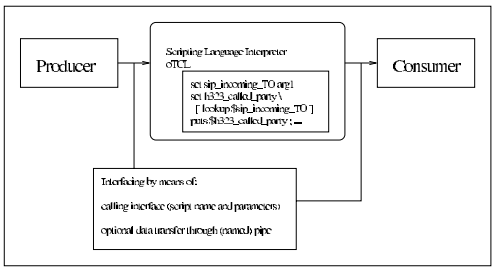


Fig. 3. Integrating scripting functionality

We have chosen to integrate oTcl, an object oriented variant of Tcl, because of its flexibility, easy integration with Unix IPC mechanisms and even direct C or C++ linkage. Its mechanisms have shown to be well suited for the notation and computation of even complex algorithms [22]. During the development and test process of the gateway a certain functional block can be coded as a call to an oTcl function. This one itself does either use generic Tcl code or may wrap native code that is loaded as a shared library at runtime.

This allows to build and dynamically modify data and signaling paths by passing information through (named) pipe chains and via command line arguments.

H. Gateway feature set and interoperability

The gateway is able to handle multiple connections that are identified by the persistent callReferenceValue (CRV) on the H.323 side as well as by the unique SIP call identifiers on the SIP side. Since just the control data but no RTP audio packets have to be exchanged via the gateway it does not form a performance bottleneck.

The implementation has been tested with different infrastructure and end system components both on the H.323 and SIP side. Its current features are listed in Table II. Naturally it interoperates with the SIP User Agent sua that is part of the Vovida protocol stack.

Feature	Conformance
Media capabilities	Support for participants using G.711 (no further codec negotiation even if both participants can support those, yet)
Protocol support	SIP and H.323 according to the features of the building stacks, support for both H.323v1 style and Fast Connect call setup
Environment Integration and Address Mapping	H.323 Gatekeeper-less or Gatekeeper-attached mode using configurable address mapping within the gateway
Scalability	Support for multiple parallel calls

TABLE II
CURRENT GATEWAY FEATURES

On the H.323 side we were able to successfully test the gateway with a number of H.323 gatekeepers from different vendors (Tenovis, Siemens) as well as with the Open Source gatekeeper opengate [21] and the clients NetMeeting, OpenH323 ohphone (formerly voxilla) and the LP5100 IP phone.

I. Availability of other implementations

By the time we finished the first gateway implementation for our research contractor another comparable implementation [23] combining the protocol stack used within the Columbia University sipd and OpenH323 became available. It is distributed in a binary evaluation version and source code can be obtained and licensed for evaluation and further use.

We see our implementation as to a certain extent similar to this one, while our intended licensing policy will differ. The gateway can now (after we reached an agreement with the research contractor) released fully Open Source. Additionally we will use it as an research and implementation basis for the integration of Supplementary Services.

IV. CHALLENGES FOR IP TELEPHONY SERVICE ENHANCEMENTS

In general we can distinguish between *Supplementary Services* (which implies a well defined meaning within the H.450.*n* protocol framework in the H.323 world) and a more general set of additional functionality that we can call *Value Added Services* (such as e.g. Presence or Instant Messaging Services as well as the description and customization of complex *Communication Workflows*). They differ in both where and by which basic mechanisms they are provided, composed and furnished.

A. Service Description and Parameterization

The *Call Processing Language* (CPL) [24] approach primarily addresses service parameterization by untrusted developers

or users within both end as well as infrastructure systems. It uses XML for notating the processing of parameters and the intended functionality. Following the IN-Service model CPL is strictly formalized and uses a decision graph technique, hence a *Directed Acyclic Graph* (DAG), to describe the flow of the program. This allows methods for analyzing worst-case paths and a guarantee for well-defined termination.

CPL scripts can be transported and placed either by out-of-band means but preferably using core protocol mechanisms such as the transport as payload of a SIP REGISTER message. The usage of CPL for creating and describing service logic for H.323 telephony systems is currently under evaluation and ongoing discussion with participation of experts from both the ITU as well as the IETF groups.

B. Integration Mechanisms

Whereas the CPL approach assumes a runtime-environment that has been pre-established within the infrastructure and end system components (and primarily needs to be parameterized), services can also be provided at a lower (more implementation centric) level.

At the moment two programming language and implementation mechanisms for SIP are proposed. For trusted developers an extended CGI (Common Gateway Interface) called *SIP CGI* [25] is considered. The CGI technique is well known for creating dynamic content for web pages or triggering external interactions in the http environment. It is well-suited for developing applications using any programming language and having access to any resource.

Another approach is the use of the Servlet technology. A *SIP Servlet* [26] is a specialized Servlet which executes telephony service logic. It is written in Java code and interacts with a SIP server or a "Servlet-Engine". The standardized Server API [27] allows to run the code on all Servlet-enabled servers. Because it defines a possibly standardized framework and JAVA compile- and runtime checkings as well as security precautions, we conceptually see it between the stringent restrictions of a CPL and the complete (but often also undesired and dangerous) openness of SIP CGI.

C. Relevance for Gateway Design

The approaches build a solid base for writting and installing in particular high level Supplementary Services, whereas our initial focus lies more on *Call Control* services. The gateway must handle the mappings between the protocols H.450 and SIP first of all to ensure an seamless and transparent interworking. Design and implementation should consider the above mentioned approaches though.

V. Providing Gateway Functionality for Supplementary Services

Existing H.323/SIP gateway implementations only support interworking for basic call functionality while interworking for Supplementary Services is (as far as public information is available) not provided yet. Since exactly those services are considered crucial for widespread user acceptance we expect this to change rapidly within the near future.

At the moment the definition of the feature set and appropriate protocol mappings [13], [11] are under development and standardization. The ITU-T Recommendations H.450 specify several ISDN-like services for H.323, whereas Supplementary Services for SIP are specified within the Working Groups of the IETF.

A. Signaling Protocol Extensions

Describing and configuring a service can only be based on the functionality that the underlying protocol provides. This involves both the specification and standardization of new protocol sequences using existing PDUs and methods as well as the definition of new ones, if this is necessary.

Call Transfer [28], *Call Park and Pickup* [29] and a comprehensive set of features comparable to those well-known for the classical ISDN [30], [31], [32], [33] are defined as Supplementary Services for the H.323 environment within the H.450 protocol series. The ITU Recommendations basically define a detailed, stringent and complete set of new (A)PDUs needed for a service, whereas the SIP approach appears to be more "functionality and mechanism centric".

Consequently a conceptional framework [34] for enhancing SIP with Supplementary Services has been derived meanwhile. On this basis two more drafts, that address dedicated Call Control services – Call Transfer [35] and Call Diversion Indication [36] – are available.

Once protocol primitives are established, they have to be integrated and parameterized using higher layer mechanisms. Defining a comprehensive operational system does not need a "part by part" but an integrated system approach. Design and implementation decisions definitely have a general effect and often alternatives can be chosen for where to place a certain functionality. Therefore we consider these higher layer mechanisms within our interworking scenario as well.

B. Analysis of Use Cases and protocol mappings

Whereas the definition and deployment of services within one protocol world is already a challenge, their interoperability in hybrid signaling scenarios is a task, that protocol gateways will have to deal with in the future. Therefore we will show an analysis of possible scenarios and their implications for enabling appropriate mechanisms within our gateway. We have chosen the following example, because it involves general basic functionality that can be refound and adapted in other scenarios.

The ITU-T Recommendation H.450.2 (Call Transfer) describes the service which enables the served user A to transform an established call (user A - user B) into a new call between user B and a user C selected by user A. The involved H.323 and SIP parties are shown in Figure 4.

Participants may be located in three different "zones", each with an own Gatekeeper or Registrar / Proxy server.

A SIP client (SIP_1) is talking to an H.323 client ($H.323$) through a gateway. This call should be transfered into a call between this H.323 and another SIP client (SIP_2). For simplification we do not explicitly show Gatekeeper or Registrar interaction for resolving symbolic names or addresses.

To enable Call Transfer between a H.323 and a SIP client, the SIP side should use a modified stack supporting the SIP *RE-*

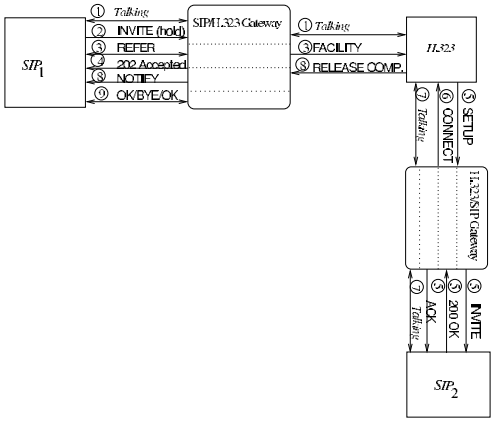


Fig. 4. Schematic scenario for Call Transfer

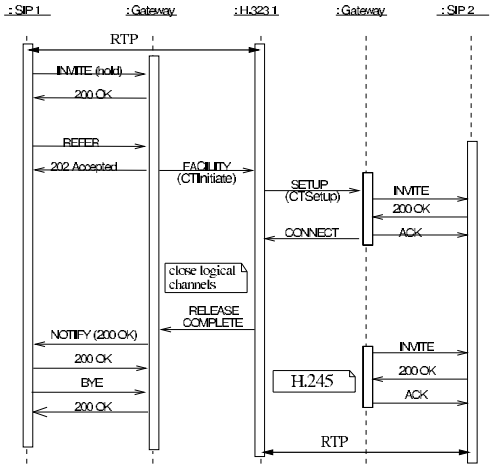


Fig. 5. Call Transfer signaling with gateways

FER [35] call control extension and must interact with an H.323 stack supporting H.450.2. The sequence of messages between the gateways and the three entities (SIP_1, $H.323$, SIP_2) is described schematically within the following paragraph. Additionally signaling details are shown in Figure 5.

In SIP an unattended transfer (the counterpart to the blind transfer in H.323) of a call is instantiated by laying the connection with the endpoint, which should be transfered, on hold. This can be achieved by sending an INVITE (hold) message (2). When this is confirmed, a REFER message (3) from the transferor to the transferee is sent. If the transferee is willing to accept the call transfer, it signals this by a 202 ACCEPTED message.

The REFER method indicates, that the recipient should contact a third party using the contact information provided in the method. The REFER message is "translated" by the gateway into its H.323 equivalent, a FACILITY request (3). The REFER method indicates, that the recipient should contact a third party using the contact information provided in the method.

To transfer the call the REFER-headers *Refer-To* and *Referred-By* are used to convey the necessary information. On the H.323 side the FACILITY message is a Q.931 message type defined within the H.225 protocol [37]. Both messages contain (among other information) the address of the new endpoint. With this information a H.225 SETUP sequence (5) to the new endpoint is initialized, which is turned into an INVITE message from the H.323/SIP-Gateway.

After a successful negotiation between the H.323 client and SIP_2 (messages sequence 5) and the receipt of a CONNECT (6) messages at $H.323$ the initial connection is released (8) using H.225.0 RELEASE COMPLETE message. On the SIP side the connection is torn down by the gateway, using a sequence of NOTIFY (8) and 200 OK, BYE and 200 OK (9).

The generally known and implemented interworking techniques can be used to perform the appropriate the H.323/SIP and H.245/SDP (capability handling) mappings.

The description (with its several mappings and transitions) should show that the (prototyping) test and deployment of Supplementary Services can benefit from the availability of a flexible notation and runtime support for protocol sequences and payloads within the gateway.

C. Integrating Supplementary Services into our Gateway

To enable Supplementary Services over a gateway, several of the components shown in Figure 2 are involved and have to be changed or enhanced.

First of all, signaling stacks that provide the low-level functionality for the services (being able to generate and interpret appropriate PDUs such as FACILITY and REFER in our example) have to be chosen and integrated. By interfacing them via Linkage modules this is possible either by enhancing the existing stacks or replacing them with new ones without breaking the core functionality.

The protocol message translation and their semantic interpretation will be done within the core components H2S and S2H. Through the usage of scripting for notating both static message mappings as well as dynamic protocol state transitions our implementation is well suited for doing that in a fast rapid prototyping way.

We consider this integrated scripting functionality as one of its main benefits, because it enables us to quickly react on changes in the specifications and do selective tests. Additionally the question of whether unexpected and undesired feature interactions may occur can be practically monitored.

VI. Conclusions and Future Work

Within this paper we have described the design and implementation of an extendable full Open Source H.323/SIP gateway that will form the core of our ongoing work on providing Supplementary Services for both IP telephony protocol worlds. We consider the use of scripting within the gateway logic as promising approach that should be discussed as work in progress.

The source code of the gateway (in its version without H.450 support for Supplementary Services) will be available as Open Source for further evaluation and enhancement.

REFERENCES

[1] International Telecommunication Union, "Visual Telephone Systems and Equipment for Local Area Networks which provide a non-guaranteed Quality of Service," *Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, May 1996.

[2] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: Session Initiation Protocol," *RFC 2543*, March 1999.

[3] Ismail Dalgic and Hanlin Fang, "Comparison of H.323 and SIP for IP Telephony Signaling," *Proc. of Photonics East, Boston, Massachusetts*, September 1999.

[4] H. Schulzrinne and J. Rosenberg, "A Comparison of SIP and H.323 for Internet Telephony," *The 8th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 98), Cambridge, England*, July 1998.

[5] Charles Agboh, "A study of two main IP telephony signaling protocols: H.323 signaling and SIP; a comparison and a signaling gateway specification," M.S. thesis, Universite Libre de Bruxelles (ULB), 1999.

[6] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTSP: A Transport Protocol for Real-Time Applications," *RFC 1889*, January 1996.

[7] International Telecommunication Union, "G.711: Pulse code modulation (pcm) of voice frequencies," *Series G: Transmission Sytems and Media. Digital Sytems and Networks. Standardization Sector of ITU, Geneva, Switzerland*, November 1988.

[8] International Telecommunication Union, "G.723.1: Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s," *Series G: Transmission Sytems and Media. Digital Sytems and Networks. Standardization Sector of ITU, Geneva, Switzerland*, March 1996.

[9] International Telecommunication Union, "G.729: Coding of speech at 8 kbit/s using conjugate-structured algebraic-code-excited linear prediction (cs-acelp)," *Series G: Transmission Sytems and Media, Digital Sytems and Networks. Standardization Sector of ITU, Geneva, Switzerland*, March 1996.

[10] K. Singh and H. Schulzrinne, "Interworking between SIP/SDP and H.323," *Internet Draft draft-singh-sip-h323-01.txt*, January 2000, Work in progress.

[11] K. Singh and H. Schulzrinne, "Interworking between SIP/SDP and H.323," *1st IP Telephony Workshop 2000, Berlin*, April 2000.

[12] SIP-H.323 Interworking Team, "SIP-H.323 Interworking Team," *http://www.softarmor.com/sipwg/teams/sip-h323/index.html*, 2000.

[13] Radhika R. Roy, Vipin Palawat, Alan Johnston, Charles Agboh, David Wang, Kundan Singh, and Henning Schulzrinne, "SIP-H.323 Interworking Requirements," *Internet Draft draft-agrawal-sip-h323-interworking-reqs-00.txt*, January 2000, Work in progress.

[14] H. Agrawal, R. Roy, V. Palawat, A. Johnston, C. Agboh, D. Wang, K. Singh, and H. Schulzrinne, "SIP-H.323 interworking requirements," Internet Draft, Internet Engineering Task Force, Feb. 2001, Work in progress.

[15] "DynamicSoft SIP Server," http://www.dynamicsoft.com/solutions/sipproxy.html.

[16] "Columbia University sipd," http://www.cs.columbia.edu/~hgs/sipd/.

[17] "kphone and libdissipate," http://www.div8.net/dissipate/.

[18] "Vovida SIP stack," http://www.vovida.org.

[19] "OpenH323, Part of the Linux VOXILLA Telecom Project," http://www.openh323.org/.

[20] Ralf Ackermann, Vasillios Darlagiannis, and Ralf Steinmetz, "Implementation of a H.323/SIP Gateway," *Technical Report TR-2000-02, Darmstadt University of Technology, Industrial Process and System Communications (KOM)*, July 2000.

[21] "Openh323 Gatekeeper," http://www.willamowius.de/openh323gk.html.

[22] USC Information Sciences Institute, *The Network Simulator - ns-2*, http://www.isi.edu/nsnam/ns/.

[23] "Columbia University sip323," http://www.cs.columbia.edu/~kns10/software/gw/.

[24] J. Lennox and H. Schulzrinne, "CPL: A language for user control of internet telephony services," *Internet Draft draft-ietf-iptel-cpl-02.txt*, January 2000, Work in progress.

[25] J. Lennox, J. Rosenberg, and H. Schulzrinne, "Common gateway interface for sip," *RFC 3050*, January 2001.

[26] A. Kristensen and A. Byttner, "The SIP Servlet API," *Internet Draft draft-kristensen-sip-servlet-00.txt*, Sep 1999, Work in progress.

[27] Ajay P. Deo, Kelvin R. Porter, and Mark X. Johnson, "The SIP Servlet API," *Java SIP Servlet API Specification*, April 2000.

[28] International Telecommunication Union, "H.450.2: Call transfer supplementary service for H.323," *Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, February 1998.

[29] International Telecommunication Union, "H.450.5: Call park and call pickup supplementary services for H.323," *Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, May 1999.

[30] International Telecommunication Union, "H.450.3: Call diversion supplementary service for H.323," *Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, February 1998.

[31] International Telecommunication Union, "H.450.4: Call hold supplementary service for H.323," *Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, May 1999.

[32] International Telecommunication Union, "H.450.6: Call waiting supplementary service for H.323," *Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, May 1999.

[33] International Telecommunication Union, "H.450.7: Message waiting indication supplementary service for H.323," *Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, May 1999.

[34] B. Campbell, "Framework for SIP Call Control Extensions," *Internet Draft draft-campbell-sip-cc-framework-02.txt*, March 2001, Work in progress.

[35] Robert Sparks, "SIP Call Control – Transfer," *Internet Draft draft-ietf-sip-cc-transfer-04.txt*, February 2001, Work in progress.

[36] S. Levy, B. Byerly, and J. Yang, "Diversion indication in SIP," *Internet Draft draft-levy-sip-diversion-01.txt*, November 2000, Work in progress.

[37] International Telecommunication Union, "H.225.0: Call signalling protocols and media stream packetization for packet-based multimedia communication systems," *Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, September 1999.

[38] Olivier Hersent, David Gurle, and Jean-Pierre Petit, *IP Telephony: Packet-Based Multimedia Communications Systems*, Addison-Wesley, 2000.

[39] Bill Douskalis, *IP Telephony: The Integration of Robust VoIP Services*, Prentice-Hall, 2000.

An Architecture for Three Challenging Features

Pamela Zave
AT&T Laboratories—Research
Florham Park, New Jersey, USA
pamela@research.att.com

Abstract—**The ECLIPSE project shows how concerns related to the complex behavior of telecommunication systems can be separated from concerns related to the IP implementation of that behavior. Within this context, the behavioral problems of Location and Identification, Switching and Spontaneous Conferencing, and Mail are addressed. Each of these features is challenging because it is not clear which functions should be included within its boundary, because it encompasses many possible behavioral variations and conflicting requirements, because it interacts with many other features, and because (like all features) it must be extensible. For each feature, the paper proposes a boundary and presents an architecture for performing all the functions within the boundary, with all their behavioral variations. The paper also shows how the composition of these feature architectures is extensible and reveals potential feature interactions. Bad feature interactions can be prevented, and good feature interactions preserved, by minor adjustments to the feature composition.**

Keywords— **multimedia telecommunication services, requirements, feature interaction, user interfaces, personal mobility, conferencing, mail**

I. Introduction

TELECOMMUNICATION services are now migrating to IP networks. The good news is that designers can offer customers whatever they might need or want. In this new context, services are unconstrained by the PSTN legacy or by inflexible technology.

The bad news is the same. Designers of telecommunication features in this new, unconstrained context will be faced with a truly bewildering range of surprisingly difficult choices. They will find that the work required to choose and justify the externally observable behavior of a new feature—in every possible situation—is a large fraction of the total work required to create it.

There are three fundamental reasons why these choices are so difficult. First, telecommunication services are developed incrementally, adding features over time (this has been true of telephony and many other complex application domains, and there is no reason why IP telecommunications should be different). Although decisions should be made with future extensibility in mind, they must be made at a time when future requirements cannot be predicted.

Second, many worthy goals conflict. Everyone expects greatly enhanced functionality, yet enhanced functionality conflicts with ease of use, because complex functionality requires a complex user interface. Enhanced functionality also conflicts with universality of communication, because some functions require the cooperation of all communicating parties. Such functions can only be used by a customer to communicate with other parties whose functions are similarly enhanced.

Third, even when telecommunication features are conceived as being independent, they necessarily interact, which means that they modify or influence one another in determining the system's overall behavior. To manage feature interactions well, designers must predict potential feature interactions, decide which are desirable and which are undesirable, and engineer feature composition so that only the desirable interactions occur. Unfortunately, there is a great deal of experience to prove that people perform these tasks poorly [15], and that they are in fact intrinsically difficult [3], [5], [6], [7], [11].

This paper addresses these problems for three important features. It shows that each of the three features—Location and Identification, Switching and Spontaneous Conferencing, and Mail—is an appropriate unit of development, in the sense that it satisfies a set of closely related requirements, best considered together. A software architecture for each feature shows how the requirements can be satisfied straightforwardly. In addition to shortening the path from requirements to implementation, the architecture alleviates each of the three problems presented above.

First, the architecture guarantees extensibility. It is a specialization or application of the Distributed Feature Composition architecture for describing telecommunication services [9], [16], [17], [19]. DFC has been proven successful at providing feature modularity and feature compositionality, so that the problems of extending a DFC system with new features are minimal.

Second, the architecture helps designers make requirements trade-offs. The architecture for each feature is somewhat general-purpose, and accommodates a range of behavioral variations. Thus designers can experiment with the details of user interfaces and detailed behaviors for

each feature, without altering the overall system organization.

Third, the architecture helps manage feature interaction. DFC is well-suited to predicting potential feature interactions and to engineering feature composition so that all of the desirable interactions can occur, while none of the undesirable ones can. Obviously, the feature architecture here inherits these capabilities from DFC.

II. Basic Service and Baseline Architecture

Figure 1 illustrates the basic multimedia service to which the three features are added. The features are intended to work with all telecommunication devices, from the large (PCs) to the small (cellphones and pagers). A call includes one two-way signaling channel and any number of two-way media channels. For concreteness, this paper focuses on just two communication media: voice and text. For simplicity, it ignores gateways to other networks.

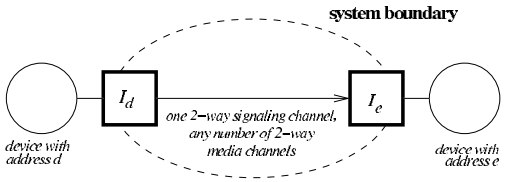


Fig. 1. Basic service.

The baseline architecture is DFC. The following overview of DFC provides just enough information to understand the architecture of the three features.

In Figure 1, I_d and I_e are *interface boxes*. These modules translate between the external protocols of the devices and the internal DFC protocol. An interface box is persistent even if the connection between the device and the network is not, so it is always available to represent the device to the network.

In DFC the term *customer call* is used informally, referring to an attempt by a user to communicate. A customer call generates and is responded to by a *usage*, which is a dynamic assembly of *boxes* and *internal calls*. A *box* is a concurrent process, and is either an interface box or a feature box. An *internal call* is a featureless connection between two ports on two different boxes. In Figure 1, there are no features, so the entire usage consists of the interface boxes and one internal call from I_d to I_e.

All the subsequent figures illustrate usages with *feature boxes* in them. In any snapshot of a DFC system, the usages can be defined formally as connected graphs of boxes and internal calls. Since usages change shape, merge, and split over time, however, the relationship between a usage and a customer call, and for that matter the concept of a customer call, cannot be formally defined.

Each internal call begins with a *setup phase* in which the initiating port sends a setup signal to a DFC router, and the DFC router chooses a box and forwards the signal to it. The receiving port completes the setup phase with a signal back to the initiating port. From that time until the *teardown phase,* the call exists and has a two-way signaling channel. The media channels of the call, which can be initiated from either port, are opened and closed explicitly by signals on the signaling channel.

When a feature box does not need to function, it can behave *transparently*. For a box with two ports, both of which are engaged in calls, transparent behavior is sending any signal received from one port out the other port, and connecting the media channels in both directions. The two calls will behave as one, and the presence of the transparent box will not be observable by any other box in the usage.

Having full control of all the calls it places or receives, a feature box has the autonomy to carry out its function without external assistance. It can re-route or disconnect internal calls, process media streams, and absorb or generate signals.

Figure 2 shows how a well-known desirable feature interaction (see, e.g., [4]) is accomplished in DFC. The user of the device d is attempting to call e, creating a usage with three feature boxes. F_3 has placed an internal call routed to I_e, which failed because I_e's single port is already occupied with another internal call. As a result, F_3 sends the status signal *unavailable* upstream. Any upstream feature box that receives it can treat the *unavailable* condition and absorb the signal, or can ignore the *unavailable* condition and propagate the signal further upstream. Thus downstream *unavailable* treatments have priority over upstream ones.

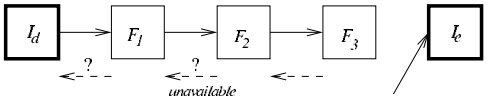


Fig. 2. How *unavailable* treatments interact.

DFC routing is an algorithm that uses provisioned configuration, subscription, and precedence information to determine the boxes in a usage. The setup signal of each internal call is sent separately to a *DFC router* for routing to another box, which may be a feature box (if more features need to be applied) or an interface box (if all relevant fea-

tures have been applied). Necessary information about the history of the usage is carried in the setup signal, so that the routing algorithm itself can be stateless.

The default routing situation is as follows. When the user of a device initiates a customer call, the device interface issues a *new call* whose setup signal contains *source* and *target* addresses. The source address determines a *source zone*, which is a sequence of feature box types subscribed to by the source address. Similarly, the target address determines a *target zone*. The usage unfolds as a linear chain of boxes and calls, eventually containing a box of each type in the source zone, followed by a box of each type in the target zone. When both zones are exhausted, the next internal call is routed to the target interface box.

Most feature boxes are *free*; when the router needs a free box, it simply routes to a new, anonymous instance of the box type. Some feature boxes, however, are *bound*; for each address subscribing to a bound box type, there is exactly one, persistent, addressable instance of the box type. Bound boxes allow joins in usages, as an internal call can be routed to a bound box that is already engaged in other internal calls. In figures, for example 4 and 5, bound boxes and interface boxes are drawn with heavier lines than free boxes.

In the default situation, each feature box makes a *continuation call*, which is an outgoing call using exactly the same setup signal that it received as part of its incoming call. This is the mechanism that causes default routing to unfold. In contrast, feature boxes can also affect routing by making various structured changes to the setup signal of an incoming call before using it to place an outgoing call. Each routing variation is used somewhere in this paper, and is explained where used.

Most features are implemented by one type of feature box. Some features, however, require more than one box type; addresses subscribe to box types depending on which role they are playing with respect to the feature. Boxes *of the same feature* can communicate through, and maintain persisent data in, global *operational data*. Since the features in this paper are complex ones, they use more feature box types than average features.

The DFC architecture is actually a domain-specific adaptation of the pipes-and-filters architecture [13]. Its modularity is exactly the same kind as that claimed for pipes and filters in general.

III. The ECLIPSE Project

The ECLIPSE project has produced an IP implementation of DFC [1].

The philosophy of the ECLIPSE project is to separate the concerns of behavior and optimization. Feature behavior and interactions are described and analyzed in terms of DFC, so that description and analysis can benefit from DFC's modularity, abstraction, and formality. Meanwhile, the ECLIPSE implementation incorporates many optimizations that allow feature boxes to function efficiently in an IP setting. These optimizations are invisible to feature designers, and apply equally to all features developed within the DFC framework.

The current version of ECLIPSE is fully distributed, so that feature boxes can be located anywhere in a network. It incorporates an optimization that allows media streams to travel end-to-end rather than following the signaling path, and another optimization that supports efficient distribution of all routing data and most operational data. Future optimizations will address signaling latency, efficient maintenance of data consistency, and signaling/media synchronization. Separation of concerns allows us to improve these optimizations incrementally, without disturbing feature development or the execution of existing features.

Because of this separation of concerns, it is meaningful to discuss the behavior of a telecommunication system without constant reference to its implementation. This paper takes advantage of this freedom, deferring the implementation issues to other venues.

IV. The Features

A. Location and Identification

A.1 Requirements

It is attractive for a telecommunication system to offer, in addition to addresses tied to devices, *personal addresses* associated uniquely with people. A personal address can subscribe to feature boxes, own private data, and be the source or target of calls.

The concept of personal addresses leads directly to two requirements:

1. When the target of a call is a personal address, it is necessary to find a device where the person is located, and direct the call to that device [location].
2. When a user is connected to the system through a device, it is necessary to identify that user, and to ensure that the user only has access to his own subscribed feature boxes and personal data [identification].

These two requirements are intimately related and best addressed together. Briefly, the location requirement demands an answer to the question, "Which device is this person using?" The identification requirement demands an answer to the dual question, "Which person is using this device?"

The relationship between location and identification can best be explained in terms of shared customer

data. Location relies primarily on the relation *located_near(person,device)*. Identification relies primarily on the relations *has_access_to(person,device)*, representing long-term possession or authorization rather than current location, and *authenticates(password,person)*.[1] Nevertheless, identification can also use *located_near* when several people have access to a device and it is too inconvenient to demand a password. Also, a location feature might bear the responsibility of guaranteeing that the correct person has been reached, which it can only do by use of *authenticates*. Also, a location feature might use *has_access_to* to create a menu from which a user can choose in updating his location.

The concept of Location and Identification encompasses the IP-oriented definition of personal mobility [12], in which a user becomes available to the network through registration or login. It also encompasses the communication style of traditional (wireline) telephony, in which some devices are connected to the network at all times. A good feature is actually much harder to design in the telephony context, because users do not expect to have to register before they can communicate, and because some devices are shared simultaneously by multiple people.

A.2 Core Functions and Software Architecture

DFC has *mobile addresses*, addresses not permanently associated with any device. DFC mobile addresses can be used (among other things) as personal addresses.

Loc is a free box type of this feature, subscribed to in the target zone by each personal address p (Figure 3). Upon receiving an incoming call, *Loc* uses the operational data to perform the core location function of determining the device d where p is presumably located, and placing a continuation call with $target = d$.

When a feature box in the target zone changes the target in a continuation call, routing is automatically affected. The DFC router discards the remainder of the old target zone (feature box types that have not yet been routed to) and begins routing to the target zone of the new target.

Once a user has been reached through device d, *Loc* behaves transparently. By doing this, it performs the core function of giving the user access to the feature boxes subscribed to by p, and through them to the private data of p (only feature boxes routed to on behalf of p can access p's slice of the feature operational data).

Note that *Loc* must be the *last* feature box in the target zone of p, so that all personal feature boxes are guarded by it, and none are skipped when the usage is retargeted to d.

[1] The type *password* is intended to include fingerprints, voice prints, and other means of bio-identification.

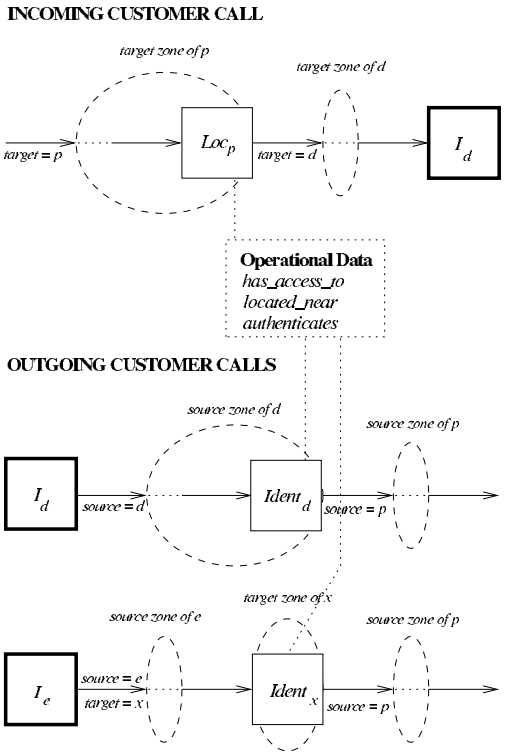


Fig. 3. Architecture of the Location and Identification feature.

Ident is a free box type of this feature, subscribed to in the source zone by each participating device d (Figure 3). Upon receiving an incoming call, *Ident* uses the operational data to perform the core identification function of determining which personal address the user owns, if any. If there is a relevant personal address p, it places a continuation call with $source = p$. Just as in the target zone, this will cause the router to throw away the rest of the current source zone, and to route the continuation call to the first feature box of the source zone of p, giving the user access to the feature boxes and data of p. If there is no relevant personal address, the box places a continuation call with no changes, thus continuing with the source-zone feature boxes of d.

A *participating device* is intended for use by people with personal addresses, and thus subscribes to *Ident*. But the owner of a personal address might be using a public device with no such subscription, such as e in Figure 3. In

such a case the only way to invoke *Ident* is to dial a special address x that subscribes to *Ident* in the target zone. In this context, if identification succeeds *Ident* collects a real target address from the caller and places a *new* outgoing call, so that routing to the source zone can begin again.

The relation *located_near* is highly dynamic. New location information can be entered manually through *Loc* or *Ident*, once the feature box has identified its user.[2]

A.3 Behavioral Variations

A wide variety of identification policies can be built into different versions of this feature. At one end of the spectrum, *located_near* and *has_access_to* might be unambiguous enough, and trusted enough, to identify the user in all cases. For example, if there is a device such as a cellphone to which only one person has access, the feature might assume that any user of the cellphone is that person. At the other end of the spectrum, every new use of a device requires a password. Note that if *Loc* demands a password, it does so after a user has answered the call to the device interface, but before allowing the user to exchange signals with the personal feature boxes of p.

Between the two extremes lies the murky territory of devices shared among several people. For these devices, it might take quite a bit of experimentation to find a policy that balances security and convenience satisfactorily.

Another range of variation concerns the failure behavior of *Loc*. First, is it a failure if no user answers the outgoing call, if the user cannot authenticate himself, or only if *Loc* has no current location for p? Second, does *Loc* attempt to handle a failure itself (perhaps by placing an outgoing call to a different address), or does it send an *unavailable* signal upstream to be treated by another feature?

There is also a range of variation in the user interface of this feature. How is *located_near* modified? How is its current value displayed? Do *Loc* or *Ident* inform the user when they have identified him as owning a particular personal address?

Finally, if a user connected to a *Loc* or *Ident* box through device d uses the box to change his current location from d to e, the box can offer to transfer the ongoing customer call from d to e. For example, d might be a home PC and e a cellphone. If the user is talking through d and needs to leave home, he can transfer his location and the conversation to e, pick up the cellphone, and walk out the door with it. This function is discussed further in Section V-B.

[2]It also makes perfect sense to collect location information automatically. However, the design decisions entailed therein are so different that it is a different feature entirely.

B. Switching and Spontaneous Conferencing

B.1 Requirements

A person can only deal with one voice channel at a time. Even passive listening to two voice channels simultaneously is unlikely to be comfortable or effective.

The concept of a single voice channel leads directly to two requirements:

1. If a user has several customer calls in progress, it is necessary to switch the user's single voice channel among the various calls [switching].
2. A user with several customer calls in progress should be able to conference together the voice channels of some or all of those calls [conferencing].[3]

These two requirements are intimately related and best addressed together. The reason is *completeness*, which means in this case that at any time, a user should be able to group his calls into conferences in any way that he chooses, and to be speaking with whichever conference (counting an unconferenced call as a singleton conference) that he chooses.

If switching and conferencing are separated, then completeness becomes extremely difficult to achieve. To see why, consider Figure 4, which shows a configuration achievable through traditional PSTN features. Subscriber d has used 3-Way Calling to make a conference with e and f, and has used another instance of 3-Way Calling to make a conference with g and h. The subscriber's Call Waiting feature enables him to switch between talking to these two conferences. However, there is no way that he can form an f/g conference. The historical grouping of customer calls into conferences is embedded in the configuration, and cannot be altered without tearing calls down and setting them up again.

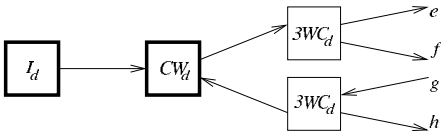


Fig. 4. A completeness problem.

B.2 Core Functions and Software Architecture

This feature is implemented by the bound box type *SSC*, as shown in Figure 5. The instance of *SSC* has one internal

[3]This function is referred to as *spontaneous conferencing* because it operates locally on existing customer calls. Pre-arranged conferences are much more elaborate, requiring configuration functions, participation management, floor control, and security [10].

call on its left, connecting *SSC* to its associated device. This internal call has one voice channel, which is managed by *SSC*.

On its right *SSC* has many internal calls, each corresponding, from the perspective of device *d*, to a customer call. The design of *SSC* gives each of the internal calls on its right a special, formal significance here designated a *switched call*, because it is a call relative to, and preserved by, *SSC*.

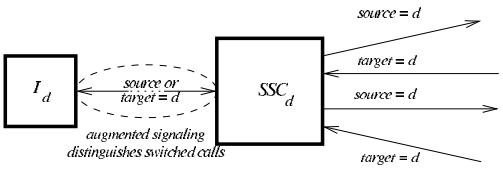


Fig. 5. Architecture of the Switching and Spontaneous Conferencing feature.

Except for its switching and conferencing functions, the behavior of *SSC* with respect to switched calls is intended to be transparent. However, in this case the effect of transparency takes some work to achieve:

1. Each switched call has a locally unique identifier used by *SSC* and *I*. If the switched call is an outgoing call from the device, *I* assigns its identifier. If the switched call is an incoming call to the device, *SSC* assigns its identifier.
2. Except for signals specifically related to switching and conferencing functions, all signals of all switched calls pass transparently through *SSC*. Between *I* and *SSC* signals are labeled with their switched-call identifiers, so both boxes can distinguish which switched call a signal belongs to.
3. From the perspective of *I* and *SSC*, there is a big difference between a switched call that appears when the boxes are idle, and a switched call that appears when the boxes are busy; the former results in setting up a chain of boxes and internal calls between *I* and *SSC*, while the latter piggybacks signaling and media on the existing chain. Nevertheless, the user interface provided by *I* makes both look the same to the user.

Note that the architecture of this feature requires substantial cooperation from the interface box.

Device address *d* subscribes to *SSC* in both the source and target zones. *SSC* is a bound box type. Therefore each switched call to or from *d* is routed through the unique box SSC_d. In Figure 5, the chain of boxes and calls between I_d and SSC_d is double-arrowed because it might have been placed in either direction. It persists as long as the busy episode of the device persists.

There are many ways of providing voice conferencing and switching functions. A minimal, but complete, scheme requires that the user be able to choose individual switched calls and conferences. A voice conference is a set of switched calls. Each switched call belongs to exactly one conference; when it is created (placed or received) it is put in a new singleton conference. Then only two user operations are needed:

1. *select(f: conf)*, which connects the user's voice channel to conference *f*, disconnecting it from any other voice source/sink.
2. *move(c: call, f: conf)*, which moves call *c* (within *SSC*) from whichever conference it is in to conference *f*.

We have to choose between making *SSC* a device feature box, subscribed to by addresses of participating devices, or a personal feature box, subscribed to by personal addresses. This is not an easy choice.

The main advantage of making it a personal feature box is that it will always be available to the person, from any participating or non-participating device.

If subscribers to the system are expected to use a variety of different devices, then making *SSC* a device feature box is even more advantageous. First, to provide a good user interface, it needs to be designed with the device capabilities in mind. For example, on a voice-enabled PC each switched call can have its own window, and there is no real limit on how many of them the user can handle. On a cellphone, on the other hand, it might be wise to prohibit more than two simultaneous switched calls. This customization can be provided by having different versions of *SSC* subscribed to by different device types. If *SSC* were a personal feature box, the same version would be used from many devices, and it could not be customized in this way.

Second, this architecture preserves the integrity of switched calls by means of close cooperation between the interface box and the *SSC* box. If *SSC* were a personal feature box, then *every* device interface would have to be programmed for this particular kind of cooperation, which seems a lot to ask in an allegedly modular and extensible system. If only participating devices are expected to cooperate, then the difficulty of dealing with *SSC* as a personal feature is less, but so are the advantages, since Switching and Spontaneous Conferencing is available from participating devices in either case.

B.3 Behavioral Variations

The main variation in this feature concerns the user interface. Designers have a great deal of latitude in the user-controlled switching and conferencing operations, and in the details of how a switched call is displayed to the user.

Another behavioral variation concerns what happens to other media, such as text. Conferencing of switched calls can apply to all their media channels simultaneously, or to voice alone. Since a user can easily handle multiple simultaneous text conversations, it would make sense to conference voice only, and (on a PC, at least) to display the text channel of each switched call in its own window. Conferencing of text could also be an optional function, or an optional adjunct to voice conferencing.

It is easily possible to add a *transfer* function to this feature. Transfer applies to a conference; after a transfer, the switched calls are still connected to each other, but are no longer connected to the subscriber's device. The definition of a transfer is trickier if some of the media are not included in the conference.

C. Mail

C.1 Requirements

Voice mail originated with answering machines. *Text mail* or E-mail has been gaining momentum since the early days of the Internet. Although the history of *text chat* goes back to the early days of timesharing operating systems, it has had a recent surge of popularity in the form of instant messaging. In this context it is convenient to refer to telephony as *voice chat*.

Despite their disparate histories, these forms of telecommunication are closely related—in function, if not in implementation. The only difference between voice chat and text chat, and between voice mail and text mail, is the medium employed. The primary difference between chat and mail is that the former is real-time communication, while the latter is buffered communication.

Surely one of the best possible uses of IP telecommunications is to unify these functions. By doing so, we should be able to provide a smoother, richer, and more flexible user experience. The unification should satisfy these requirements:

1. Chat between two subscribers is always possible if both people want it.
2. Mail between two subscribers is always possible if either person wants it.
3. If two subscribers have capabilities for at least one common medium, then there is a way for them to communicate.

The basic service (Section II) is chat, so mail must be added as a feature.

C.2 Core Functions and Software Architecture

The Mail feature has two box types, both free: *Read/Send* and *Receive*. Both device and personal addresses can have mail by subscribing to *Read/Send* in the source zone and *Receive* in the target zone.

Mail is stored in the operational data of this feature, as shown in Figure 6. As with the operational data of Location and Identification, the mailbox is partitioned into address slices, and only a box routed to on behalf of address x can access the mail of address x.

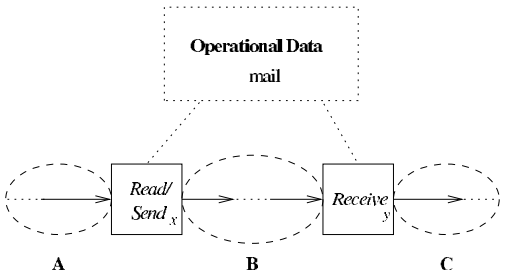


Fig. 6. Architecture of the Mail feature.

The key to understanding Figure 6 is that the three chains of boxes and internal calls marked **A, B,** and **C** may be present simultaneously or at different times. In the default situation (chat), the two feature boxes are transparent and all three chains are present simultaneously.

A *Receive* box must be provisioned (also in the operational data) to accept messages in any medium for which its subscriber has reading capability on some device.

To read mail, a subscriber places a call which will be routed through *Read/Send* in the source zone. The subscriber invokes the *read* function of this box, and opens one or more media channels to this box. The subscriber can then read his messages in all the current media. In this mode of communication, only chain **A** is present.

Alternatively, the subscriber can invoke the *send* function of the *Read/Send* box, and open one or more media channels. In this situation, with only chain **A** present, the subscriber can store in his own mailbox a message to be sent. The subscriber can disconnect **A** as soon as the message is complete.

Once the subscriber has stored a message and ordered it sent, the instance of *Read/Send* that stored the message is responsible for delivering it to the target address. It places an outgoing call to that address with a *mail* flag augmenting the setup signal. If the setup signal travels through the usage to a *Receive* box, the *Receive* box will respond to the flag. Instead of behaving transparently, it will make no outgoing call. It will accept opening of any medium in which it can accept messages, receive the stored message

from *Read/Send*, and store it in the target's mailbox. In this mode of communication only chain **B** is present.

Perhaps the target address does not subscribe to *Receive*. In this case no feature box in the usage will recognize the *mail* flag, and the flag will be ignored. Call chain **B** will be extended with call chain **C**, hopefully all the way to a device, to which the message will be delivered. If the **B** and **C** call chain does not succeed because the device is not available or no one answers it, then the instance of *Read/Send* responsible must retry periodically. Between tries it sleeps, disconnected from all internal calls. The instance cannot die until it has delivered the message for which it is responsible.

If the caller wants to chat, then both feature boxes in Figure 6 are initially transparent. However, an *unavailable* or *unanswered* condition will trigger the *Receive* function, which will accept opening of any media in which it can accept messages, and use any open media channel to offer to store mail for the target address. In this mode of communication chains **A** and **B** are present.

Finally, a caller who originally wished to chat may change his mind and want to send mail instead. He may have chatted with someone already, been disconnected by that person, and wish to send mail as an afterthought. Or the attempt to chat may have failed, he may be currently receiving an offer of mail services from the callee's *Receive* box, and wish to use his own Mail feature instead.

In either case, he can invoke the *send* function of his *Read/Send* box at this later stage. *Read/Send* tears down any parts of **B** that may remain, leaving only **A**, and then acts as it does when *send* is invoked initially.

C.3 Behavioral Variations

During chat, either box type can offer to store (record) the conversation on behalf of its subscriber.

As part of its *unanswered* treatment, *Receive* can offer a *screening* function. Instead of answering an incoming call, the callee invokes this function. The media channel is opened all the way from the caller to the callee. At the same time, *Receive* presents an *unanswered* indication to the caller and offers to store a message. If the caller has a message, *Receive* both stores it and allows it to pass through to the callee. If the callee then requests a full connection, *Receive* stops recording and connects the medium in both directions.

In a multimedia system, more things can happen during a customer call, and old concepts such as *unanswered* must be defined anew. If a caller attempts to open channels for two distinct media, and the callee only accepts one of them, does *Receive* offer to store a message in the rejected medium?

Singh and Schulzrinne suggest some additional variations [14]. A subscriber's instance of a *Receive* box might call him to *notify* him of the arrival of an important message. A call might be *reclaimed* by a user who answers the phone while a message is being recorded (this is a slight variation on screening). It should be possible to record *multimedia messages* with components in several media.

There is a wide range of variation in the user interface for the Mail feature, particularly the user interface for retrieving stored mail. Although DFC includes all the facilities necessary for describing how mail is stored and retrieved, it would also be possible to interface *Read/Send* and *Receive* with an off-the-shelf mail server.

V. Feature Interactions

Since feature interactions are a by-product of feature modularity and feature compositionality, their precise nature depends on the feature-specification language and feature-composition operator. Thus all interactions of our three features are discussed in DFC terms.

DFC was designed specifically to minimize feature interactions that are known to be undesirable, such as logical inconsistencies and implementation conflicts. Composition of features in a less structured framework would result in many more feature interactions than these, most of them bad.

The following discussion is informal, focusing on interactions that are known to occur, and on how they can be managed within DFC. The formal basis for this work, emphasizing detection of potential feature interactions and its relation to verification, is introduced elsewhere [2], [17], [18].

A. Location and Identification

This feature has many interactions. Three of them are straightforward and have already been handled by the feature architecture:

1. Since all outgoing calls placed by *Loc* have device addresses as targets, *Loc* would unconditionally cancel any feature box placed after itself in the target zone of a personal address. This would be bad, and is prevented by placing *Loc* last.
2. *Loc* generates *unavailable* signals upstream, which other personal feature boxes might handle. For example, a *Delegate* box might delegate (forward) the customer call to another person. Or *Receive* might offer to store a message. This beneficial feature interaction is supported by placing *Loc* last in the target zone of a personal address, so signals traveling upstream will pass through them.
3. *Ident* and *Loc* are intended to guard access to personal feature boxes such as *Read/Send* and *Receive*. Their place-

ment in usages puts them between personal feature boxes and users, which is necessary for this desirable guarding interaction to occur.

There is a particularly interesting feature interaction in systems in which some users own personal addresses, while others use device addresses as if they were personal addresses. If e is a device address used in this old-fashioned way, it may subscribe to personal feature boxes such as *Read/Send, Receive,* and *Delegate*. What happens when a personal address is temporarily located at this device?

Consider, for example, a personal address q subscribing to *Receive* and *Loc* in the target zone. Each incoming call to q is retargeted by *Loc* to device e, which also subscribes to *Receive* in the target zone. If no one at e answers the call, then the *unanswered* condition will be handled by the nearest *unanswered* treatment, which in this case will be $Receive_e$. This is definitely not the most desirable behavior, as the mail belongs to q.

The general solution to this interaction problem is shown in Figure 7. Figure 7 depicts a complex usage involving device addresses d and e, and personal addresses p and q. Device d has placed two switched calls, one to q and one to e. The switched call to q is retargeted to e, as q is currently located at e. The switched call to e is identified as being placed by p, who is currently located at d. This usage contains all boxes of all three features. All four addresses use Mail, and therefore subscribe to *Read/Send* in their source zones and *Receive* in their target zones.

On the source side, the feature boxes subscribed to by device address d must be partitioned into device-oriented boxes and personal boxes. The order of boxes in d's source zone must place device-oriented boxes first, followed by *Ident*, followed by personal boxes.

The upper instance of $Ident_d$ in Figure 7 does not identify the user (presumably because the user did not ask to be identified), and does not change the source address it received. It is followed in the usage by $Read/Send_d$. The lower instance of $Ident_d$ identifies the user as p and changes the source address it received. It is followed in the usage by $Read/Send_p$. On either path, there is exactly one instance of the personal box *Read/Send*.

On the target side, the feature boxes subscribed to by device address e must also be partitioned into device-oriented boxes and personal boxes. The order of boxes in e's target zone must place personal boxes first, followed by *L&I Marker*, followed by device-oriented boxes. *L&I Marker* is another free box of the Location and Identification feature. It behaves transparently in all cases, and is present specifically to solve this feature-interaction problem.

In the lower path of Figure 7, there is no retargeting, and there are instances of all the boxes subscribed to by e in the target zone. In the upper path Loc_q retargets to e. Loc_q places its outgoing call as a special *direct call* to address e. Because *Loc* and *L&I Marker* are both boxes of the same feature, and therefore have the privilege of cooperating in this way, a DFC router routes the call directly to $L\&I\ Marker_e$, bypassing the personal feature boxes subscribed to by e. On either path, there is exactly one instance of the personal box *Receive*.

In Figure 7, most of the arrows are dotted, indicating that other feature boxes might be present there if the appropriate addresses subscribe to them. The exception is the direct call, which is intended specifically to avoid the inclusion of other feature boxes.[4]

Finally, Location and Identification alters addresses, and therefore interacts with many address-sensitive features. For example, the Blocking feature consists of a free target-zone feature box *Blocking* that rejects customer calls from certain callers. The Callback Last feature has a free target-zone box *Log* that logs the source of an incoming call in operational data, and a free source-zone box *Callback* that uses the last source as a target, if requested. Both of these features can be affected by the fact that *Ident* can change the source of a customer call from a device address to a personal address. At least in these cases, the feature interaction is a good one, as a personal address is preferable for both purposes.

B. Switching and Spontaneous Conferencing

The most important feature interactions have already been handled by the feature architecture. Although Switching and Spontaneous Conferencing is a powerful feature, other features should not need to know about it to work properly. The design of *SSC*, in preserving the integrity of switched calls, ensures that feature boxes on its right (in the orientation of Figure 5) will work as expected regardless of the presence of *SSC*.

There is still a question, however, about routing. With respect to device d in Figure 5, some switched calls are incoming and some are outgoing. Yet all share the linkage between I_d and SSC_d and whatever feature boxes might be positioned there by routing. So, to preserve completely the integrity of switched calls, any sequence of feature boxes subscribed to by d in the target zone after *SSC* must be the reverse of the sequence of feature boxes subscribed to by d in the source zone before *SSC*.

[4]If e were a device address that did not subscribe to any personal boxes, then it would not subscribe to *L&I Marker*, either. In this case the direct call would be routed to the first box in the target zone of e.

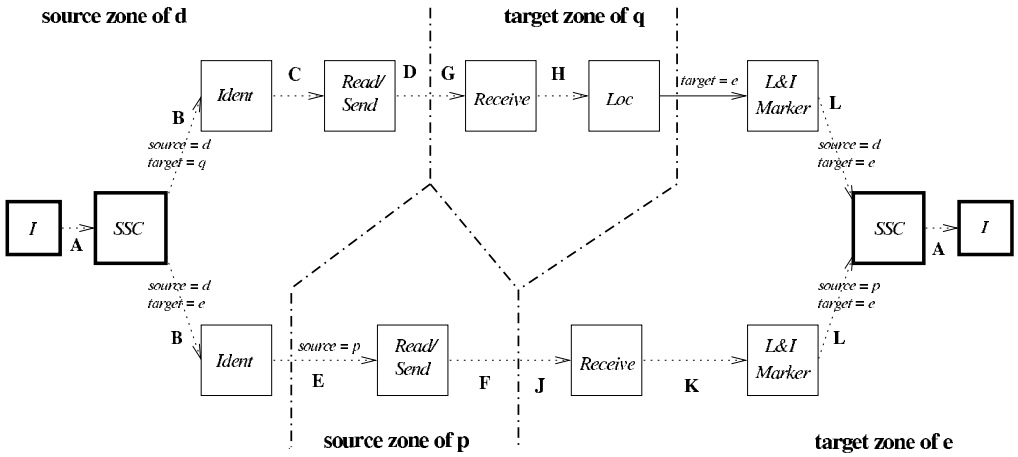


Fig. 7. The three features together.

One feature box that fits perfectly between I_d and SSC_d, and obeys the routing restriction above, is *Break-In*. *Break-In* is part of the Emergency Break-In feature, which allows agencies authorized to handle emergencies to connect immediately to a device at any time, regardless of its state and features.

Figure 8 shows how this feature works. Normally *Break-In* is completely transparent. In an emergency situation, someone makes a customer call to d with an *Emergency* box in its source zone. The internal call placed by the *Emergency* box is a direct call, so it goes directly to the *Break-In* box. The *Break-In* box receives it and interrupts whatever else is going on to connect the emergency caller to I_d.

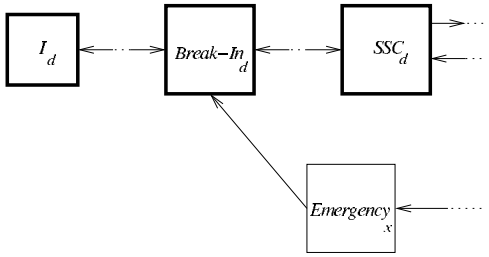


Fig. 8. The Emergency Break-In feature.

Note that the *Break-In* box needs no knowledge of the special signaling arrangement between I_d and SSC_d (see Figure 5). From its perspective, the internal calls on its right and left in Figure 8 are absolutely ordinary. Regardless of how many switched calls they are supporting, these internal calls are interrupted as wholes.

The importance of switched calls is illustrated by adding a transfer function to Location and Identification, as mentioned in Section IV-A.3 and pictured in Figure 9. Originally $Ident_d$ received the call marked **A** (and continued it to the right), and had not yet placed the call marked **B**. Later the owner of personal address p used $Ident_d$ to change his location to e, and $Ident_d$ transferred the ongoing switched call to I_e by placing **B** and tearing down **A**.

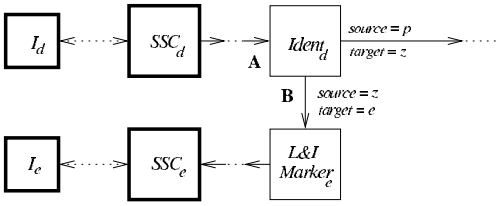


Fig. 9. The transfer function of Location and Identification.

B is a direct call to *L&I Marker*. This creates a switched call at e having two targets and no source. However, provided that the switched call is past the setup phase (in which source and target are asymmetric) and into a com-

munication phase (in which source and target are symmetric), the unusual configuration should cause no problems.

The point of Figure 9 is that the transfer is feasible because it is only operating on one switched call. There is no confusing involvement with the other switched calls of d or e, and there are no signaling limitations to prevent *Ident* from performing this new function. *Ident* should ensure the success of the connection to e before tearing down the connection to d, as SSC_e might already be juggling its maximum number of switched calls.

C. Mail

Many of Mail's potential interactions are exemplified by its interactions with Location and Identification, and have already been discussed in Section V-A.

The Personal Directory feature maintains a list of personalized names and their associated addresses. Its target-zone box, *Personal Caller ID*, uses the list to translate the source address of an incoming customer call to a name whenever possible. Its source-zone box, *Personal Dialing*, uses the same list to provide speed dialing.

A reasonable target-zone sequence for an address subscribing to all the personal feature boxes mentioned would be: *Personal Caller ID, Blocking, Receive, Delegate*. If the *Delegate* function is activated, it should supersede *Receive* as a failure treatment. Blocked customer calls should not reach the *Receive* box, as they should not be allowed to leave messages. All of *Blocking, Receive,* and *Delegate* can use personalized names rather than addresses, if the names are supplied from upstream by *Personal Caller ID*.

The function of Mail could be augmented incrementally with a source-zone *Mailing List* box that is activated when its incoming internal call has the *mail* flag set and the name of a mailing list in the place of an address. *Mailing List* takes a message from *Read/Send* as if it were a *Receive* box, buffers it, and places one outgoing internal call for each entry in the mailing list, delivering the message to that address.

A reasonable source-zone sequence for an address subscribing to all the personal feature boxes mentioned would be: *Read/Send, Mailing List, Personal Dialing. Mailing List* cannot do its job unless it comes after *Read/Send*. If *Personal Dialing* follows *Mailing List*, then mailing lists can have personalized names in them, which *Personal Dialing* will translate to addresses.

The mailing features also illustrate the limits of switched calls. When an instance of *Mailing List* is active, it receives one internal call corresponding to one switched call, and places many internal calls. Further away from a *SSC* box than a *Mailing List* box, switched calls have become meaningless.

Hall describes eight other E-mail features within in a DFC-like architecture, and analyzes their interactions [8]. Because of the close similarities between the two architectures, these results apply to DFC as well.

D. Examples of Extensions

In Figure 7, bold letters mark regions of the usage where additional feature boxes could be placed. To summarize examples used throughout Section V, here is a list of regions and feature boxes that could be placed there:

A *Break-In*
B *Callback*
D *Mailing List, Personal Dialing*
F *Mailing List, Personal Dialing*
G *Personal Caller ID, Blocking*
H *Delegate*
J *Personal Caller ID, Blocking*
K *Delegate*
L *Log*

Callback Last (*Callback, Log*) is a device feature because its semantics depends on the ordering of switched calls at a particular device.

Regions **B** and **L** might also be the home of media-choice features—features that help users negotiate on which medium they will communicate. For example, a box in region **L**, observing that a caller has attempted to open a channel of a medium that device e does not support, could signal to the caller which media e *does* support. Media-choice features must be closely associated with devices because their functions depend entirely on which media their device supports.

VI. Conclusion

This architecture also handles additional complexity that is not discussed here, for lack of space. There are additional behavioral variations and feature interactions, some of them quite interesting.

Nevertheless, there is enough detail to show the potential of the architecture. It reduces the designer's overall burden to a manageable level by minimizing his need to discover new issues and interactions, by separating concerns, and by solving some specific design problems altogether. Furthermore, the ECLIPSE implementation of DFC shows that real telecommunication systems can be built in this way.

Acknowledgments

My colleagues Greg Bond, Eric Cheung, Michael Jackson, Hal Purdy, and Chris Ramming have all contributed greatly to this work.

REFERENCES

[1] Gregory Bond, Eric Cheung, Andrew Forrest, Michael Jackson, Hal Purdy, Chris Ramming, and Pamela Zave. DFC as the basis for ECLIPSE, an IP communications software platform. In *Proceedings of the IP Telecom Services Workshop 2000.*

[2] Gregory Bond, Franjo Ivancic, Nils Klarlund, and Richard Trefler. ECLIPSE feature logic analysis. In this volume.

[3] L. G. Bouma and H. Velthuijsen, editors. *Feature Interactions in Telecommunications Systems.* IOS Press, Amsterdam, 1994.

[4] Kenneth H. Braithwaite and Joanne M. Atlee. Towards automated detection of feature interactions. In [3], pages 36-59.

[5] M. Calder and E. Magill, editors, *Feature Interactions in Telecommunications and Software Systems VI.*, IOS Press, Amsterdam, 2000.

[6] K. E. Cheng and T. Ohta, editors, *Feature Interactions in Telecommunications Systems III*, IOS Press, Amsterdam, 1995.

[7] P. Dini, R. Boutaba, and L. Logrippo, editors. *Feature Interactions in Telecommunication Networks IV.* IOS Press, Amsterdam, 1997.

[8] Robert J. Hall. Feature interactions in electronic mail. In [5], pages 67-82.

[9] Michael Jackson and Pamela Zave. Distributed feature composition: A virtual architecture for telecommunications services. *IEEE Transactions on Software Engineering* XXIV(10):831-847, October 1998.

[10] Nadia Kausar and Jon Crowcroft. An architecture of conference control functions. In *Proceedings of Photonics East,* Boston, Massachusetts, September 1999.

[11] K. Kimbler and L. G. Bouma, editors. *Feature Interactions in Telecommunications and Software Systems V.* IOS Press, Amsterdam, 1998.

[12] Henning Schulzrinne. Personal mobility for multimedia services in the Internet. In *Proceedings of the European Workshop on Interactive Distributed Multimedia Systems and Services,* Berlin, Germany, March 1996.

[13] Mary Shaw and David Garlan. *Software Architecture: Perspectives on an Emerging Discipline.* Prentice-Hall, 1996.

[14] Kundan Singh and Henning Schulzrinne. Unified messaging using SIP and RTSP. In *Proceedings of the IP Telecom Services Workshop 2000,* pages 27-33. Atlanta, Georgia, September 2000.

[15] Hugo Velthuijsen. Issues of non-monotonicity in feature-interaction detection. In [6], pages 31-42.

[16] Pamela Zave. DFC website at `www.research.att.com/info/pamela/dfc`.

[17] Pamela Zave. An experiment in feature engineering. In *Essays by the Members of the IFIP Working Group on Programming Methodology,* Springer-Verlag, to appear.

[18] Pamela Zave. Feature-oriented description, formal methods, and DFC. In *Proceedings of the FIREworks Workshop on Language Constructs for Describing Features*, pages 11-26. Springer-Verlag, London, 2001.

[19] Pamela Zave and Michael Jackson. New feature interactions in mobile and multimedia telecommunication services. In [5], pages 51-66.