YESSIR: A Simple Reservation Mechanism for the Internet

Ping Pan and Henning Schulzrinne

Abstract—
RSVP has been designed to support resource reservation in the Internet. However, it has two major problems: complexity and scalability. The former results in large message processing overhead at end systems and routers, and inefficient firewall processing at the edge of the network. The latter implies that in a backbone environment, the amount of bandwidth consumed by refresh messages and the storage space that is needed to support a large number of flows at a router are too large. We have developed a new reservation mechanism that simplifies the process of establishing reserved flows while preserving many unique features introduced by RSVP. Simplicity is measured in terms of control message processing, data packet processing, and user-level flexibility. Features such as robustness, advertising network service availability and resource sharing among multiple senders are also supported in the proposal.

The proposed mechanism, YESSIR (YEt another Sender Session Internet Reservations) generates reservation requests by senders to reduce the processing overhead, builds on top of RTCP, uses *soft state* to maintain reservation states, supports shared reservation and associated flow merging and is compatible with the IETF Integrated Services models.

YESSIR extends the all-or-nothing reservation model to support partial reservations that improve over the duration of the session.

To address the scalability issue, we investigate the possibility of using YESSIR for per-stream reservation and RSVP for aggregate reservation.

I. Introduction

A. Background

Continuous media applications such as Internet telephony, teleconferencing, interactive multimedia games and media-on-demand have become increasingly popular in the Internet. There are several driving forces behind the growth of continuous media applications: The deployment of IP multicast in the Internet via the MBONE overlay network has provided an important platform for testing and deploying various audio and video applications. Also, the Real-Time Transport Protocol (RTP) [1] facilitated the development of interoperable applications, which have become available across a wide range of platforms. Finally, end systems have become capable of generating and rendering highly-compressed multimedia content.

However, as the usage of the Internet has grown, packet loss, delay variations and lack of bandwidth [2] have made the current Internet unsuitable for widespread delivery of *predictably* high-quality continuous media services. While only sufficient network capacity can correct these problems, it is often desirable to give improved service to certain classes of applications[1].

We can distinguish per-packet and per-flow approaches to providing differentiated QOS:

Per-packet: Information necessary to obtain differentiated service is carried inside each data packet. In IPv4, for example, the type-of-service (TOS) byte may be used or priority may be accorded to certain port numbers. Routers do not need to maintain state beyond a rough classifier list that can be considered static; there is no control protocol overhead.
This approach cannot guarantee a certain QOS, since the amount of traffic injected with a particular TOS value, for example, cannot be bounded. Thus, the per-packet approach is subject to intentional and unintentional denial-of-service attacks.
Per-flow: Resources for a set of packets distinguished by source or destination addresses and port numbers ("flows") are reserved ahead of time, with reservations established and torn down dynamically.

Both approaches can be combined; for example, a resource reservation protocol can limit the amount of traffic with a particular TOS value.

We will focus primarily on per-flow resource reservation. Currently, RSVP [3], [4] is the reservation protocol of record in the Internet. Unfortunately, its generality implies a cost in complexity, as discussed in Section I-C. We set out to address some of these issues.

B. Overview

This paper describes and evaluates YESSIR (YEt another Sender Session Internet Reservation[2]), an in-band, sender-based resource-reservation protocol based on RTP that offers significantly lower code and run-time complexity than RSVP.

YESSIR is motivated by the observation that a large fraction of the applications that require guaranteed quality-of-service are continuous media applications and that a substantial fraction of these either use or will use the Real-Time Transport protocol (RTP) to deliver their data. YESSIR and RSVP can operate side-by-side in the same network, without affecting the certainty of guarantees offered to applications.

In this section, we will briefly explain some of the design decisions that introduce complexity into RSVP and describe features of RTP relevant to YESSIR. The rest of paper is organized as follows: In Section II, we will outline the design goals of YESSIR. The YESSIR reservation mechanism will be detailed in Section III. Section IV reports on our experimental implementation and its performance. Several open issues (including a possible solution to the scaling problem) are discussed in Section V. We summarize in Section VII.

C. RSVP Complexity Issues

Initially, RSVP was perceived as a light-weight reservation protocol, in comparison, for example, with ATM signaling pro-

P. Pan is with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598. pan@watson.ibm.com.

H. Schulzrinne is with the Department of Computer Science and the Department of Electrical Engineering, Columbia University, New York, NY 10027. schulzrinne@cs.columbia.edu.

[1] Particularly, if the users of these applications may be willing to pay for improved predictability in quality-of-service. Note that even with network bandwidth that is statistically sufficient, many continuous-media applications demand *consistent* service throughout the lifetime, for example, of a phone call or video-on-demand session, unaffected by traffic bursts of other applications.

[2] The name reflects the proper attitude of a resource reservation protocol in a well-designed network.

tocols such as Q.2931. However, as implementations are weighing in at between 10,000 and 30,000 lines of code, it seems appropriate to review some of the design features that contribute to the complexity:

Receiver orientation: In RSVP, receivers make reservations, based on information provided by senders. This allows individual receivers within a single multicast group to request different levels of service guarantees, including none. It seems likely, however, that receivers will simply request whatever traffic bandwidth the sender has indicated, either through RSVP PATH messages or some other session initiation protocol [5].

The separation of reservation and path-finding messages for receiver-oriented reservation mechanisms imposes additional processing and protocol complexity.

Receiver diversity: At least for bandwidth diversity, reservations are an inappropriate means to distinguish classes of receivers. Bandwidth diversity could only be accomplished by "thinning" flows, i.e., dropping packets, as flows reach parts of the network endowed with less bandwidth. However, random packet dropping will quickly degrade most audio and video encodings due to their use of prediction across packet boundaries. Other mechanisms, such as layered multicast [6], were found to be superior to support diverse receiver populations.

RSVP receivers can request different values of queuing delay as part of their resource reservation. However, in high-speed wide-area networks, queuing delay is dominated by propagation delays; also, in popular scheduling disciplines such as weighted fair queuing, queuing delay for an individual flow can only be improved by allocating more than its "true" bandwidth, thus incurring allocation inefficiency, particularly if reserved flows constitute a large fraction of the total link capacity.

Receiver diversity and receiver orientation require that nodes merge incoming reservations into a single reservation setting aside the least upper bound of the requests of all downstream receivers. Flow merging also introduces the need for "blockade" state to prevent so-called killer reservation (see Section III-E).

Flow management: Because reservation requests are generated from downstream, keeping track of next-hops can become difficult and CPU intensive, particularly in multicast-capable non-broadcast multiple access (NBMA) networks such as ATM subnets.

Application modification: Since RSVP is an out-of-band protocol, applications need to be modified, either to take advantage of kernel-level support for RSVP or to convey their resource requirements to some external agent that makes reservations on their behalf. Both solutions incur complexity.

Firewall complexity: Firewalls are also complicated by RSVP's out-of-band nature. First, they need to parse and pass RSVP requests; they also need to correlate between the flows (session-sender pairs) described in RSVP messages and the actual data streams (Fig. 1).

The reservation mechanism introduced here avoids these problems.

D. RTP Features Useful for Resource Reservation

RTP [1] has been designed to provide end-to-end delivery services for data with real-time characteristics. Although protocol-independent, applications normally run RTP on top of UDP to make the use of its multiplexing and checksum services. It has been widely implemented on multimedia systems across all operating systems and is part of the ITU H.323 recommendation for conferencing and Internet telephony. Examples include *vic* [7], *vat* [8], *rat* [9] and *NeVoT* [10] for teleconferencing over the MBONE, *NetMeeting* from Microsoft and conferencing tools from Netscape.

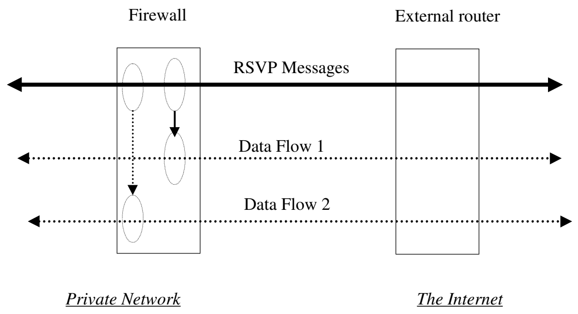


Fig. 1. RSVP overhead at a firewall

Although RTP was not intended as a resource reservation protocol, resource reservation can benefit from the following RTP features:

In-band signaling: RTP defines two types of packets: RTP for data transport and RTCP for control. Each RTP session consists of one RTP data stream and one corresponding RTCP stream, originated by one or more participants. When carried over UDP, data and control packets use adjacent port numbers, so that a router or firewall can easily map from a control stream to the corresponding data stream.

Periodic sender/receiver notification: Senders and receivers periodically send RTCP packets containing reports to the multicast group. Data senders distribute sender reports (SRs) that indicate, inter alia, the number of bytes and packets transmitted since the last report and information allowing the estimation of round-trip times. Data receivers include receiver reports (RRs) that indicate packet loss and delay statistics, among others.

By evaluating these reports, all participating members have knowledge of traffic characteristics, network congestion and session membership. Routers can deduce the resource requirements of a session from these reports, as will be discussed below.

The period between reports has a minimum of 5 seconds and scales with the number of participants, keeping the RTCP session control overhead limited to no more than 5% of the data bandwidth. Senders are allocated at least 25% of the session control bandwidth.

Embedded in applications: RTP is typically implemented as part of the application. As will be shown in Section III-B, even an RTP application that runs the current version of RTCP can be used to initiate resource requests. No kernel modifications, beyond the support of IP router alert options (Section I-E), are needed.

E. IP Router Alert Option

The IP router alert option [11], [12] alerts transit routers to more closely examine the contents of an IP packet. In other words, routers can intercept packets not addressed to them directly, with little performance impact. For example, RSVP PATH messages are carried in IP packets that include the router alert option. Thus, even though RSVP PATH messages are addressed to end systems, PATH messages are intercepted and processed by all transit routers. We make use of router alert options to mark RTCP sender report for YESSIR processing.

II. Design Objectives

YESSIR offers an alternative, light-weight approach to resource reservation in the Internet, using RTCP sender reports to reserve resources in the network.

It has the following properties:

Sender-initiated reservation: As motivated earlier, we anticipate that many applications cannot make full use of the benefits of receiver-initiated reservations. Sender-initiated reservation may also fit better with policy and billing, as the number of senders making reservations is likely to be much smaller than the number of receivers. In many existing systems, such as cable television, the cost of "resource reservation" is bundled with the cost of content, simplifying billing. (Also, a provider of pay-per-view services would likely want to avoid the case where subscribers pay, fail to reserve resources and then ask for their money back since the quality was unacceptable.) In the absence of an Internet-wide authentication and cross-provider billing service, it is far easier for the relatively small number of large-scale content providers, residing at known network addresses, to arrange for payment with major backbone providers than individual subscribers.

Note that RSVP could also be modified to have PATH messages initiate reservations, so that the benefits of sender orientation for some applications do not depend on the use of YESSIR.

Robustness and soft-state: Similar to RSVP and PIM [13], routers maintain reservation states as *soft state*, i.e., reservations disappear by themselves if not refreshed periodically. In YESSIR, this avoids orphan reservations and adapts quickly to routing changes. As in RSVP, an explicit teardown mechanism using RTCP BYE packet avoids holding reservations for a number of soft-state refresh intervals after the requesting application has terminated.

Allow partial reservations: The function of resource reservations is to protect existing streams against disruption by other streams that arrive later.

In "classical" reservation systems, reservations are either made or denied end-to-end. Depending on the system, the requestor can always either ask again, at some cost to the network if done too often ("redialing"). Some systems, including RSVP, also allow to specify a range of resource requests to increase the likelihood of success, however, this can cause low-bandwidth regions to experience high packet losses despite reservations.

We propose an additional reservation model, that of a partial reservation, where some fraction of the links have resource protection for a particular flow, others may not. On links without reservation, traffic is carried on a best-effort basis and the resource reservation request continues downstream towards the receivers. Since YESSIR is a soft-state protocol that resends reservation requests periodically, a flow can acquire a reservation on a link when another flow terminates, without having to retry at the application layer. The user can decide whether to put up with a partially successful reservation and hope that more links will be added as the session continues or cancel the session. For a live presentation, where inserting an end-to-end reservation means missing the event, a user may well decide that the prospect of improving reservation fortunes may be better than not listening at all or foregoing all resource reservations[3]. YESSIR supports both end-to-end and partial reservations.

Provide different reservation styles: YESSIR supports *individual* and *shared* reservation styles. Individual reservations are made separately for each sender, whereas shared reservations allocate resources that can be used by all senders in an RTP session.

Individual reservations are called for when all senders are active simultaneously, e.g., for distribution of participant video in a conference, while shared reservations are appropriate where several senders alternate, e.g., for audio in a conference. (Shared reservations also avoid the problem that a new speaker may not be able to acquire a reservation; they can re-use the existing reservation of the previous speaker.)

These styles are simplified versions of the fixed filter and wildcard filter reservations in RSVP. Note that the shared reservation styles, one of the distinguishing features of RSVP, does not depend on receiver orientation. YESSIR handles the shared reservation style from the sender's direction, while RSVP supports shared reservation (shared-explicit and wildcard-filter styles) from the receiver's direction.

Low protocol and processing overhead: Rather than defining another signaling protocol, YESSIR messages are transported by RTCP. Given that RTP is in-band signaling and its data and control packets are tightly coupled, updating packet classifiers and firewalls can be simplified. YESSIR uses one message to set up a reservation. Its processing algorithm is very simple, as we will illustrate in Section III-B.

Interoperable with RTP and the IntServ models: YESSIR messages are piggybacked in RTCP. The operation of existing RTP functions at end systems is not affected at all. YESSIR can describe the traffic flows in terms of the service models [14], [15] that have been specified in the IETF IntServ working group.

Provide link resource advertising function: The purpose of making link-level resource reservation is to meet end-to-end application requirements. To that end, YESSIR is able to query and carry collected network resource information to the end systems.

[3] Partial reservations can lead to fragmentation, where a large number of flows all have partial reservations, with unacceptable quality. This aspect is the subject of current work. The soft-state mechanisms also gives a slight advantage to high-bandwidth flows or flows with few senders, as they may get to send RTCP requests more frequently.

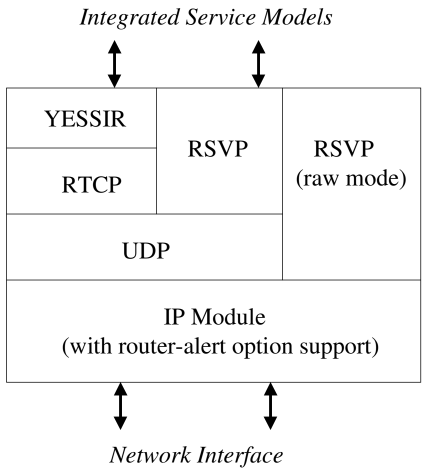


Fig. 2. Protocol relationships

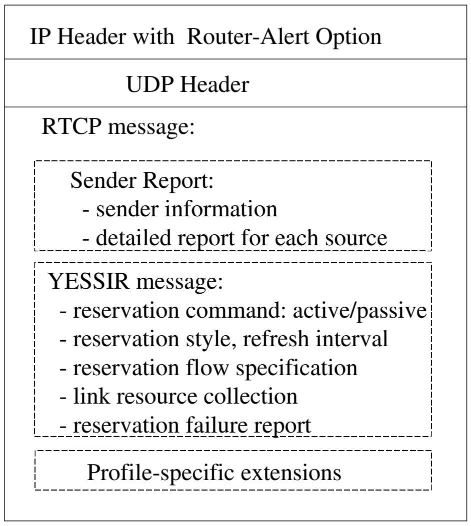


Fig. 3. YESSIR message format

III. YESSIR Operation

A. Protocol Overview

YESSIR reservation messages consist of RTCP sender-report messages, possibly enhanced by additional YESSIR-specific data, carried in IP packets with router-alert options. The placement and relationships to other protocols are shown in Figure 2.

Reservation requests generated by senders are intercepted and processed by those routers that support the router-alert IP option. Routers that do not support the option or YESSIR forward the RTCP message unaltered to the next hop. End systems ignore the router alert option. Thus, YESSIR can be deployed incrementally and without affecting the behavior of end systems.

An optional reservation extension for RTCP is defined. It is piggybacked at the end of an RTCP report (SR or RR), as shown in Figure 3. The YESSIR extension consists of a generic fragment, a flowspec fragment, an optional network monitoring fragment and an optional reservation error fragment. The generic fragment instructs the router as to the desired reservation style (individual or shared), the soft-state refresh interval and whether to make partial reservations. The flowspec fragment provides the router with the necessary information as to whether to admit the flow and what resources to set aside. The flow specification can be in several formats. The optional network monitoring fragment stores link resource information. If it is present in a request, every router along the path updates the link information in the fragment. Currently, we use the ADSPEC format [16] defined by the IETF IntServ working group. Routers where reservation requests fail indicate the reason for failure in the optional reservation error fragment. The fragment is used to collect error information that will allow end systems and network administrators to diagnose reservation failure inside the network.

B. Outline of Operation

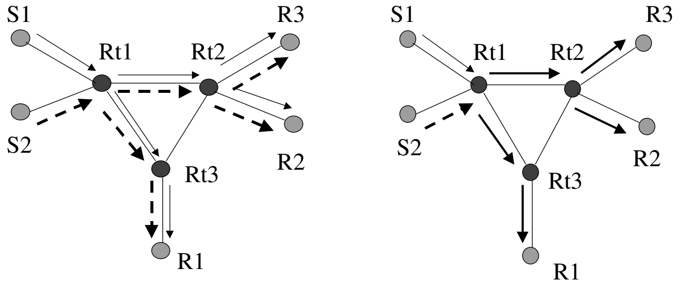
Senders periodically multicast RTCP sender reports (SRs) to all members of the multicast group (or the other party, if unicast). Sender reports contain transmission and reception statistics. Routers may either use the transmission statistics or additional YESSIR flowspecs and other elements.

As shown in Figure 3, YESSIR may insert reservation information into SR, however, YESSIR can also operate without any additional information beyond what is already contained in RTCP sender reports. When an RTCP SR is received by a router, the router will attempt to make a resource reservation according to the information specified in the message.

If a reservation request cannot be granted at a router, the SR packet will continue to be forwarded to the next hop(s). The router has the option of inserting reservation failure information into the SR. As a part of RTCP receiver reports (RR), the receivers will provide failure information to the senders. Based on RRs received, senders can either drop the session, or lower the reservation request and transmitted bandwidth.

If a reservation request is accepted by a router, the corresponding RTP data stream information is added into the packet classifier, and the router's scheduler is updated to support the new stream.

Instead of basing reservations on flowspecs, YESSIR can also operate in a measurement mode. Measurement mode makes use of the fact that RTCP SRs contain a byte count and a timestamp. If the first RTCP packet for a session does not contain a flowspec, the router simply records the timestamp and byte count, but does not make a reservation. If a second packet for the same session comes along, the router computes the difference in time stamps and byte counts and thus computes an estimated rate. It then establishes reservations for this measured bandwidth, updated as new RTCP packets arrive. Compared to other measurement-based admission controls [17], this frees the router from the burden to count packets and estimate rates. Another measurement method, which we have not explored in detail, simply has the end system mark an RTP data packet ev-



(a) *Distinct Reservation style*: Reservations for S1are shown as in solid line; S2, in dotted line.

(b) *Shared Reservation style*: At Rt1, after flow merging between reservation for S1 (solid line) and S2 (dotted line), a single reservation (thicker line) is made to Rt2 and Rt3.

Fig. 4. Different reservation styles (S1 and S2 are senders, R1, R2 and R3 are receivers in a single multicast RTP session; Rt1, Rt2 and Rt3 are routers)

ery so often with an IP router alert option. Each RTP packet contains a payload type indication, which indicates the media encoding (e.g., G.711-encoded voice). For many low bit rate codecs, the payload type is associated with a fixed rate (e.g., 64 kb/s for G.711), so that the router can make reservations based on that information alone. This mode, while less general and flexible than the current YESSIR mode, has the advantage of trivial header parsing and fixed refresh intervals. (It also incurs the danger of increased packet delay variation and packet reordering since some RTP packets would traverse a routers "slow path", while most would not.)

Reservation states in each router are maintained as soft-state. The reservation is automatically removed if no RTCP SR is received within several consecutive refresh intervals. To reduce the processing burden at routers, instead of having routers to initiate refresh messages, RTP senders periodically generate RTCP SR's with IP router-alert option to refresh reservations. Compare with hop-by-hop refresh mechanism that has been used in RSVP, the end-to-end refresh mechanism in YESSIR reduces both routers' processing cost (as shown in Section IV-B) and protocol overhead. For example, for a single-sender session, the sender uses merely 1.25% of the session bandwidth for sending SR's. Assuming that a typical SR is about 100 bytes long, any sender with a session bandwidth above 2100 b/s will send a report at least every 30 seconds.

In addition, an RTCP BYE message, sent when a group member leaves, releases the YESSIR state record and any resource reservations.

C. Reservation Styles

YESSIR defines two reservation styles, individual and shared. In individual reservations, every sender in a RTP session has a resource reservation of its own. As shown in Figure 4 (a),router Rt1 receives reservation requests from both senders S1 and S2. After making a reservation, there are two separate reservations on links between Rt1 to Rt2 and Rt3. Depending on the amount of requested resource, RTP data streams from S1 and S2 may have different levels of reservation.

In a shared reservation, all senders of an RTP session share a single resource reservation in the network. As illustrated in Figure 4 (b), the links Rt1-Rt2 and Rt1-Rt3 have a single shared reservation. The amount of resources reserved on the link is the least upper bound (LUB) of the individual flow requests from S1 and S2. For example, if S1 and S2 request 10 kb/s and 15 kb/s of bandwidth, respectively, the shared bandwidth for link Rt1-Rt2 will be 15 kb/s. If there is a reservation failure, the reservation rejection information and the merged flow specification will be piggybacked in the RTCP sender report. Receivers will feed back the failure information and rejected reservation request to all participating members, including the senders. The senders can use these reports from receivers to adjust their requests. Flow merging issues will be addressed further in section III-E.

D. Flow Specification

A sender can specify the resource it is requesting (the flowspec) in different formats. We have considered three types for YESSIR: IntServ, RTP PT (payload-type), and TOS (type-of-service).

For applications that support the IntServ traffic models, namely the controlled-load [14] and guaranteed service [15], the flowspec format will be the one that has already been designed by the IETF IntServ working group [16]. In the flowspec, the requested bandwidth, the burst size and a service class need to be specified explicitly.

For some well-known and well-understood traffic types such as voice, the flowspec contained in the RTCP SR can simply list the current RTP payload type [18]. Separate ranges of the payload type values have been set aside for audio and video, so that a router can assign RTP flows at different granularity: by session, by payload type value or by media class. To reduce the number of queues, a router may simply assign all voice traffic to

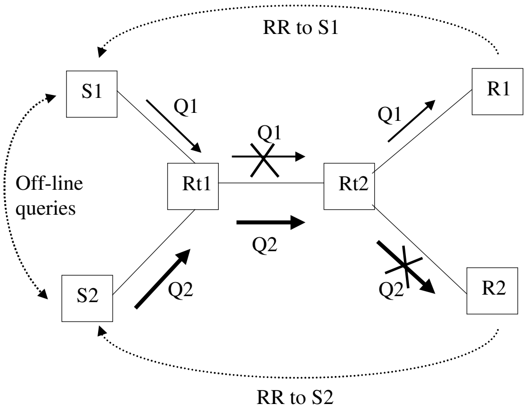


Fig. 5. Problems due to resource contention

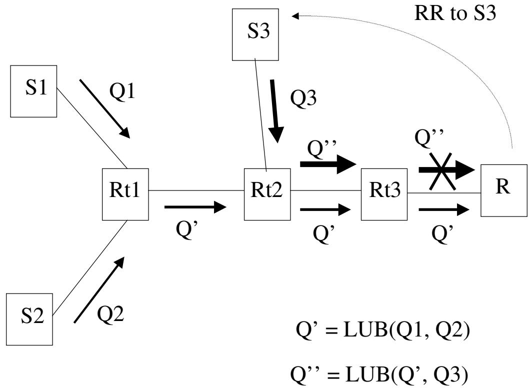


Fig. 6. Flow merging for shared reservation

a single high-priority queue, for example and just track the multicast destination and accumulated bandwidth for each session.

Similar to the RTP PT format, the TOS format allows routers to use the IP type of service information in RTP data packets to map them the appropriate scheduler queue. The YESSIR flowspec contains the TOS value and the allocated bandwidth. This allows the router to keep track of the bandwidth allocated for each TOS value, preventing over-commitment, yet avoids having to look up per-flow state for each packet. To prevent abuse by end applications, gateways rather than end systems would be expected to set the TOS value.

E. Killer Reservations

In a heterogeneous network, a reservation request may fail for any number of reasons at a router. Unfortunately, such failures may also affect requests from other senders. Figure 5 demonstrates the *killer reservation* effect. Two requests $Q1$ and $Q2$ (where $Q1 < Q2$) arrive at router Rt1. If $Q2$ arrives first and is accepted at Rt1, but rejected at Rt2, it could cause a smaller reservation $Q1$ to be rejected at Rt1 since the resource has been taken by $Q2$. As a result, neither request will enjoy and end-to-end QoS guarantee.

RSVP and ATM each solve this problem differently. RSVP generates RESVERR messages and creates a *blockade state* to allow smaller reservation requests going through while blocking large requests. Unfortunately, blockade states are difficult to manage and incur high implementation complexity. If an ATM reservation cannot be accepted by switch, that switch sends back a resource release message towards the sender to tear down the reserved resource at upstream nodes.

In YESSIR, partial reservations for both $Q1$ and $Q2$ will be made. However, senders receive an indication that the reservation was only partially successful and can then change or drop the reservation, clearing the way for other reservations to succeed.

F. Flow Merging

In YESSIR, flow merging only takes place for shared reservations. As discussed earlier, the merged flowspec is the least upper bound (LUB) value of the flowspecs from all participating senders. Here, we propose a best-effort approach to flow merging: when there is already a reservation in place, this reservation remains if a larger reservation request from another sender cannot be granted. As a result, all senders will have some fraction of their bandwidth reserved, though they may have different reservation requirements.

Figure 6 shows an example. S1 and S2 are the initial senders of a shared-reservation RTP session. The merged flowspec Q' is reserved inside the network, where $Q' = \text{LUB}(Q1, Q2)$. Later, a new sender S3 joins the RTP session and requests $Q3$ worth of resources. Router Rt2 tries to reserve the merged flowspec $Q'' = \text{LUB}(Q', Q3)$. Assume the reservation is successful and the new request Q'' is relayed to router Rt3. If Rt3 cannot reserve Q'', it should continue to use the previous reservation Q'. Sender S3 will be informed about the last workable reservation Q' from receiver R via RTCP and will ultimately decide if it wishes to continue to participate in the session or whether it can lower its sending rate.

G. Error Handling at Routers

In YESSIR, a router does *not* generate error messages to the senders, nor does it try to automatically correct problems such as *killer reservation* that are introduced due to reservation failures at neighboring routers. Instead, it inserts error information into the SR message. It is up to the receivers to inform the senders about reservation failures via RTCP receiver reports. Also, RTCP sender reports containing YESSIR reservation requests are always forwarded, even if unsuccessful.

We chose this approach for several reasons:

1. This behavior is simple to implement. As shown in several RSVP implementations [19], the support for error message handling and associated blockade states are costly in terms of protocol processing, timer management and extra state storage.

2. For links where resources are relatively plenty, such as a gigabit Ethernet, there is no reason to reserve resources for small data streams. In this case, a router should ignore YESSIR messages, and forward the requests downstream.
3. Managing resource over shared-media network such as Ethernet and token-ring networks is difficult. In this case, a router can insert a "reservation-undoable" flag in the error fragment of the RTCP SR message and forward it downstream.
4. More importantly, as described earlier, reservations are soft state. If a resource is not available at the first reservation time, there is always a possibility that reservation can be made during refresh times.

H. Dynamic Reservation Feature

An RTP session may not require a reservation for its whole duration. If reservations cost money, an application may well decide to only reserve network resources if best effort service proves unsatisfactory[4]. RTP-based applications supporting YESSIR can easily operate in this "reserve-when-needed" mode, as YESSIR reservation requests are coupled with RTCP messages. RTCP receiver reports have been designed to monitor traffic statistics. Senders can monitor receiver reports and only include a reservation request if a sufficiently large fraction of receivers indicate reception problems.

I. Network Resource Advertising

In order to satisfy end-to-end service requirements, we adapted the OPWA (*One-Pass With Advertising*) scheme proposed by Shenker and Breslau [20] and described by Wroclawski [16] for YESSIR. Here is how it works in the context of YESSIR: each reservation request message carries a network monitoring fragment that consists of fields for hop counts, propagation delay, aggregated bandwidth and delay bounds. As SR messages traverse routers, this fragment will be updated at every hop. The receivers, upon reception of the SRs, will send the collected path resource information back to the senders in RTCP receiver reports. The senders can refer the path resource information to adjust their reservation levels by sending new requests.

J. Updating the Packet Classifier

As shown in Figure 7, when a YESSIR message is received, the reservation setup agent will query the local traffic control database for resource availability. If the resource is sufficient at the egress interface(s), the agent updates the database and the scheduler.

According to the RTP profile, RTP data uses an even port number and the corresponding RTCP stream uses the next higher (odd) port number. Thus, during the parsing of RTCP messages, RTP data packet information including the IP source and destination addresses, port numbers and protocol type can be learned automatically.

After the router successfully sets up the scheduler, it inserts RTCP's IP source and destination address, protocol type (presumably, UDP), and the corresponding RTP data port numbers

[4] It obviously runs the risk that reservations will fail when the network is sufficiently busy to drop best-effort traffic.

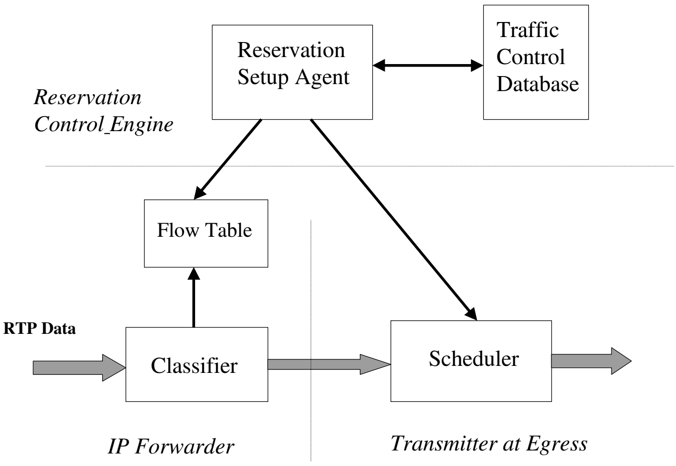


Fig. 7. A router model for reservation support

into the flow table. When RTP data packets are received, the packet classifier filters on the IP and UDP headers and forward the packets to the scheduler.

K. Security Considerations

RTCP and RTP data are tightly coupled. Thus, at a firewall, when a rule for a particular RTP data stream is defined, it will be automatically applied to the corresponding RTCP messages. Similarly, if a rule has been define to accept certain RTCP messages, the associated RTP data will be accepted as well. Supporting reservations across firewalls is therefore greatly simplified.

Currently, YESSIR relies on security mechanisms at the IP layer to provide authentication. If necessary, it would be easy to add an authentication facility to either RTCP or the YESSIR elements.

IV. Implementation and Measurements

A. YESSIR Processing Algorithm

The processing of YESSIR messages in a router is very similar to that of RSVP PATH messages, except that the router needs to call the local resource managers to make appropriate reservation in the case of YESSIR. Briefly, the following algorithm can be used to make reservation on arrival of a YESSIR message. The router will only "see" RTCP messages where the IP packet header has the router alert option set.
1. Perform a quick sanity checks on the UDP and RTCP headers;
2. locate the flow's reservation state in the router based on the IP source and destination addresses and UDP port numbers; [5]
3. if we cannot locate one, create a new reservation state;
4. query the routing tables to find egress interface(s);
5. make a reservation based the flowspec fragment in YESSIR message or the bandwidth measurement in the router state

[5] Alternatively, we could also hash on the 4-byte RTP synchronization source identifier (SSRC) instead of 12 bytes of source/destination information. Even though the SSRC is only unique within each RTP session, the probability of collision is low.

Code Section	Time (μsec)	% of Total
PATH Processing:		
PATH entry creation	410.51 ± 7.51	37.12%
Route query	40.61 ± 1.84	3.67%
Send PATH downstream	283.16 ± 3.85	25.61%
RESV Processing:		
RESV entry look-up	11.03 ± 0.43	1.00%
Update reservation info	126.35 ± 1.10	11.43%
Flow merging and forward RESV	234.14 ± 3.27	21.17%
Single RSVP flow setup overhead	$1{,}105.80 \pm 9.47$	100%

TABLE I

PROCESSING OVERHEAD FOR RSVP TRIGGER MESSAGE

Code Section	Time (μsec)	% of Total
YESSIR entry creation	41.40 ± 1.24	11.61%
Route query	38.43 ± 2.00	10.77%
Update reservation info	23.33 ± 0.38	6.54%
Send YESSIR msg downstream	253.53 ± 0.74	71.08%
Single YESSIR flow setup overhead	356.68 ± 2.84	100%

TABLE II

PROCESSING OVERHEAD FOR YESSIR TRIGGER MESSAGE

record;
6. store reserved resource information in the reservation state;
7. relay the message to each egress interface.

B. A Direct Measure of Protocol Overhead

We have implemented both RSVP and YESSIR on the IBM Common Router Architecture software platform. Both implementations have the similar data structures and coding style, and share the same set of data processing routines. We measured the various costs associated with RSVP and YESSIR on a router. The measured router was the IBM 2210 Nways Multiprotocol Router, which is based on a Motorola 68040 processor with bus speed of 32MHz. Processing times were measured by reading clock ticks from the timer register of the processor that has a timing resolution of 31.25 ns per tick. We divided the times into categories so that we can have somewhat loose subjective comparison between RSVP and YESSIR.

The underlying routing protocol used in our experiment is OSPF, and it updates the routing table in case of route changes. A route query operation is a straight forward route-table lookup. All RSVP flows are set up as controlled-load, fixed-filter style, and encapsulated in IP with the router-alert option. YESSIR messages use the RTP PT (Payload-Type) format, the individual reservation style, and are encapsulated in RTCP, UDP and IP with the router-alert option. Since the packet classifier and scheduler are implemented differently depending on the physical network interface, but are the same for RSVP and YESSIR, we chose to bypass them in our tests. Data collected here only reflects the RSVP and YESSIR control path behavior.

Tables I and II present the protocol processing overheads of setting a new RSVP or YESSIR flow. Table III and IV show the processing overhead of refreshing a flow.

C. The Processing Overhead Analysis

From the measurement shown in Tables I and II, we observe that a router can set up a new reservation flow with YESSIR three faster than if it uses RSVP. On soft-state refresh, the YESSIR processing overhead is nearly 50% less than that of RSVP.

Comparing all these times, we see that the overhead of constructing and sending a message is about 250 μsec. This includes getting a new buffer [6], copying data and scheduling for transmission. RSVP requires to send two messages to setup a flow, while YESSIR takes only one message.

[6] Alternatively, we can simply modify the received packet instead of getting a new buffer. However, RSVP and YESSIR are designed to support multicast flows, where multiple buffers may be required to forward a single packet at a router.

Code Section	Time (μsec)
On receive:	
PATH entry look-up	30.39 ± 0.76
Route query	37.94 ± 1.99
RESV entry look-up	11.01 ± 0.35
Update reservation info	44.05 ± 1.39
Timer routine:	
Send PATH Refresh	262.02 ± 10.20
Send RESV Refresh	239.06 ± 3.44
Single RSVP flow refresh overhead	624.46 ± 12.26

TABLE III

PROCESSING OVERHEAD FOR RSVP REFRESH MESSAGE

Code Section	Time (μsec)
On Receive:	
YESSIR entry look-up	19.31 ± 0.69
Route query	39.93 ± 0.38
Update reservation info	24.49 ± 0.31
Forward YESSIR downstream	252.56 ± 2.00
Timer routine:	
YESSIR flow checking	8.03 ± 0.73
Single YESSIR flow refresh overhead	344.32 ± 1.88

TABLE IV

PROCESSING OVERHEAD FOR YESSIR REFRESH MESSAGE

The packet transmitting overheads become more critical during soft-state refresh process. YESSIR relies on end-to-end soft-state refresh, that is, end users periodically transmit RTCP SR's with IP-alert option to maintain the flows inside the network. As a result, a YESSIR refresh message takes about 344 μs to process. RSVP uses hop-by-hop soft-state refresh mechanism. A RSVP router in the network is required to send refresh messages both upstream and downstream. As shown in Table III, refreshing a RSVP flow takes about 624 μs. Even worse, transmitting RSVP refresh messages takes place at timer interrupt level, which locks up the memory bus during the processing, thus stalling the packet forwarding loop. If a router maintains a large number of RSVP flows, its packet forward performance can be serious degraded due to long timer interrupts. YESSIR, on the other hand, uses the timer to check the flow lifetime only, and therefore takes far less time in each timer interrupt (approximately 8 μs).

The times reported above suggest accessing packet memory at router can be expensive. Creating new flow entry and updating reservation information require extensive message parsing and copying. Given YESSIR's PT flowspec is far more smaller (one word in the message) than RSVP's IntServ-format Sender-Tspec and Flowspec objects, we observed that the cost for creating a new YESSIR entry is nearly ten times less than that for RSVP, and the cost for updating reservation data in YESSIR is five times less than that for RSVP.

D. State Maintenance Overehad Comparison

YESSIR reduces the message overhead in the network. Figure 8 shows the message overhead for RSVP and YESSIR for various numbers of receivers over a link. The protocol overhead for one RSVP flow is the summation of a PATH and a RESV message. We illustrate the overhead of YESSIR messages with IntServ flowspec format, and RTP payload-type (PT) flowspec.

RSVP has the ability to pack multiple flows inside a single RESV message. The figure shows that, for a MTU of 1500 bytes, the total protocol overhead with the "compressed" RESV format is still higher than what is for YESSIR.

V. OPEN ISSUES AND FUTURE WORK

A. A Possible Solution for Scalability

RSVP has scaling problem due to its inability to aggregate small flows. If every voice flow over an IP backbone has an RSVP connection, a router may have to manage thousands of flows at each link. For example, an OC-3 (155 Mb/s) link can support 2400 64 kb/s voice flows, taking approximately 1.2 MB of storage[7]. At a refresh interval of 30 seconds, this requires about 230 kb/s of bandwidth, based on a size of a PATH message including ADSPEC of 208 bytes and a RESV message for guaranteed service and fixed filter of 148 bytes. The router has to process about 80 refresh messages a second.

In comparison, in a typical router deployed in the backbone, it takes about 0.4 MB to store 50,000 routes[8]. The bandwidth for route updates in a stable network is negligible.

One solution to support a large number of real-time flows is to make reservations in a hierarchical fashion, by using RSVP inside the backbone and establishing a small number of large-bandwidth "virtual paths", while reserving individual flows using YESSIR in local and regional networks.

As illustrated in Figure 9, inside the backbone, routers B1 and B3 operate as RSVP proxy servers, and have established a reserved flow, B1–B2–B3. Senders S1 and S2 use YESSIR to set up reservations to receiver R over the Internet backbone. In a YESSIR message, there is a bit to indicate whether the request is active (that is, every router needs to try to make the reservation) or passive (routers must ignore the request).

When reservation requests from S1 and S2 are received, B1 will first turn the reservation bit to passive mode in the requesting messages, preventing reservations from taking place inside the backbone. At B1, the packet classifier is updated to re-direct RTP data traffic from S1 and S2 to the pre-established RSVP connection. At B3, when requests from S1 and S2 are received,

[7] ISI and IBM have reported that a RSVP flow requires up to 500 bytes for storage in their implementations.

[8] In IBM's NSFnet routers, it takes a total of 8 bytes to store a route, including the support for CIDR.

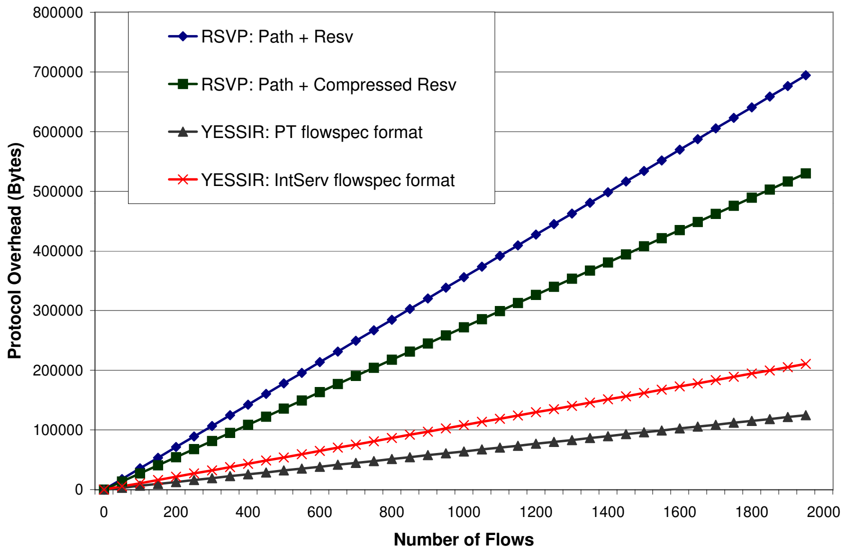


Fig. 8. Protocol overhead comparison between RSVP and YESSIR

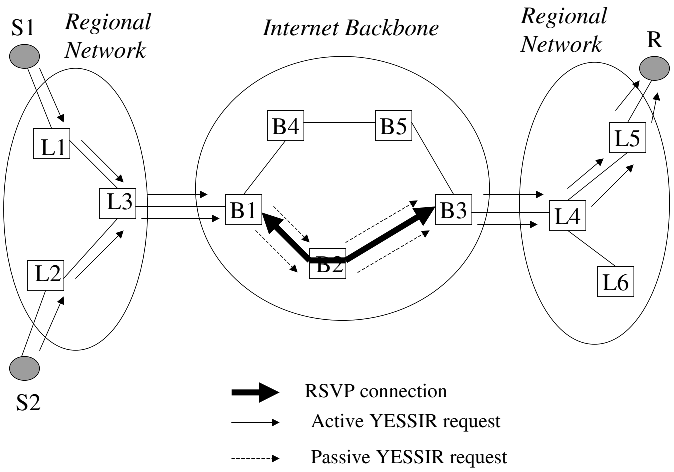


Fig. 9. An example of solving the scaling problem with two levels of reservation

the router will turn the reservation mode back to active. Requests will be routed toward the receiver R and make appropriated reservation along the way. An end-to-end reserved connection is therefore established.

Some of the unsolved issues are:

Selecting RSVP proxy servers: In the example, the RSVP flow is originated from B1 and terminated at B3. The mechanism and criteria to select a proxy server can be tricky: a BGP external speaker [21], a PIM rendezvous point [13], and a router managed by some policy agents are some of the candidates for RSVP proxy servers.

RSVP tunnel identification: The combination of source and destination address and port number should not be used to classify packets inside the backbone due to large storage overhead. How to classify packets in backbone RSVP routers needs to be studied. Possible solutions could be encapsulating data packets at the edge of the backbone, or making the use of CIDR [22], or managing IPv6 flow-ids properly.

Join YESSIR/RSVP gateway: In the example above, senders S1 and S2 have to somehow join themselves to the nearest

YESSIR/RSVP gateway, B1, prior to the reservation requesting time. The joining mechanism needs to be designed.

Reduce soft state overhead: Frequent refresh among routers can be costly if the number of flows to be managed is fairly large. On the other hand, infrequent refresh may reduce the system's ability to correct failure in timely fashion. A more efficient soft state management mechanism needs to be in place for YESSIR and RSVP. We will base our design on [23] and [24].

VI. Related Work

A number of protocols have explored sender-based reservations, including ST-II+ [25] and its predecessors, RTIP and RCAP [26] and CBSRP [27]. ST-II+ replaced IP with a new, connection-oriented Internet protocol and integrated resource reservation with establishing connectivity, thus making the smooth transition between reserved and best effort flows more difficult. RTIP and RCAP took a similar approach. All these protocols were out-of-band to the data protocol and used a "hard state" approach to state management, i.e., requiring explicit set-up and tear down of connections.

VII. Summary

Resource reservation is useful for supporting continuous-media services over the Internet. The question at this stage is: at what price? YESSIR provides a way to simplify the reservation processing and therefore reduce associated overhead at routers.

The YESSIR approach (1) is sender-initiated to support of "push" applications and simplify processing; (2) allows partial reservations; (3) supports multiple reservation styles; (4) uses soft state mechanisms to reliably and responsively maintain reservation states; and (5) takes advantage of the close relationship between RTP and RTCP packets for easy packet classification and firewall support.

We are in the process of implementing YESSIR on hosts and routers. The host implementation is based on *NeVoT*, *vic* and *vat*. The router implementation will be further developed and evaluated on experimental router prototypes from IBM Research.

We plan to interface YESSIR and RSVP to develop a hierarchical reservation system to solve the scaling problem.

References

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

[2] V. Paxson, *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California at Berkeley, Berkeley, California, May 1997.

[3] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: a new resource ReSerVation protocol," *IEEE Network*, vol. 7, pp. 8–18, Sept. 1993.

[4] B. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation protocol (RSVP) – version 1 functional specification," RFC 2205, Internet Engineering Task Force, Oct. 1997.

[5] M. Handley, H. Schulzrinne, and E. Schooler, "SIP: Session initiation protocol," Internet Draft, Internet Engineering Task Force, Mar. 1998. Work in progress.

[6] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *SIGCOMM Symposium on Communications Architectures and Protocols*, (Palo Alto, California), pp. 117–130, Aug. 1996.

[7] S. McCanne and V. Jacobson, "vic: A flexible framework for packet video," in *Proc. of ACM Multimedia '95*, Nov. 1995.

[8] V. Jacobson, "Multimedia conferencing on the Internet," in *SIGCOMM Symposium on Communications Architectures and Protocols*, (London, England), Aug. 1994. Tutorial slides.

[9] I. Kouvelas, V. Hardman, and A. Watson, "Lip synchronisation for use over the internet: Analysis and implementation," in *Proceedings of the IEEE Conference on Global Communications (GLOBECOM)*, (London, England), Nov. 1996.

[10] H. Schulzrinne, "Voice communication across the Internet: A network voice terminal," Technical Report TR 92-50, Dept. of Computer Science, University of Massachusetts, Amherst, Massachusetts, July 1992.

[11] D. Katz, "IP router alert option," RFC 2113, Internet Engineering Task Force, Feb. 1997.

[12] D. Katz, R. Atkinson, C. Partridge, and A. Jackson, "IPv6 router alert option," Internet Draft, Internet Engineering Task Force, June 1997. Work in progress.

[13] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei, "An architecture for wide-area multicast routing," in *SIGCOMM Symposium on Communications Architectures and Protocols*, (London, UK), pp. 126–135, Sept. 1994.

[14] J. Wroclawski, "Specification of the controlled-load network element service," RFC 2211, Internet Engineering Task Force, Oct. 1997.

[15] R. Guerin, C. Partridge, and S. Shenker, "Specification of guaranteed quality of service," RFC 2212, Internet Engineering Task Force, Oct. 1997.

[16] J. Wroclawski, "The use of RSVP with IETF integrated services," RFC 2210, Internet Engineering Task Force, Oct. 1997.

[17] S. Jamin, *A measurement-based admission control algorithm for integrated services packet networks*. PhD thesis, Dept. of Computer Science, University of Southern California, Los Angeles, California, Aug. 1996.

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

[19] G. Gaines and L. Salgarelli, "RSVP implementation survey," tech. rep., Institute for Information Technology of the National Research Council of Canada, July 1997.

[20] S. Shenker and L. Breslau, "Two issues in reservation establishment," in *SIGCOMM Symposium on Communications Architectures and Protocols*, (Cambridge, Massachusetts), Sept. 1995.

[21] Y. Rekhter and T. Li, "A border gateway protocol 4 (BGP-4)," RFC 1771, Internet Engineering Task Force, Mar. 1995.

[22] V. Fuller, T. Li, J. Yu, and K. Varadhan, "Classless inter-domain routing (CIDR): an address assignment and aggregation strategy," RFC 1519, Internet Engineering Task Force, Sept. 1993.

[23] Jacobson, "Scalable timers for soft state protocols," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Kobe, Japan), Apr. 1997.

[24] P. Pan and H. Schulzrinne, "Staged refresh timers for RSVP," in *Proceedings of Global Internet*, (Phoenix, Arizona), Nov. 1997. also IBM Research Technical Report TC20966.

[25] L. Delgrossi and L. Berger, "Internet stream protocol version 2 (ST2) protocol specification - version ST2+," RFC 1819, Internet Engineering Task Force, Aug. 1995.

[26] A. Banerjea and B. A. Mah, "The real-time channel administration protocol.," technical report, UC Berkeley, 1991.

[27] S. T.-C. Chou and H. Tokuda, "System support for dynamic QOS control of continuous media communication," in *Third International Workshop on network and operating system support for digital audio and video*, (San Diego, California), pp. 322–327, IEEE Computer and Communications Societies, Nov. 1992.