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.

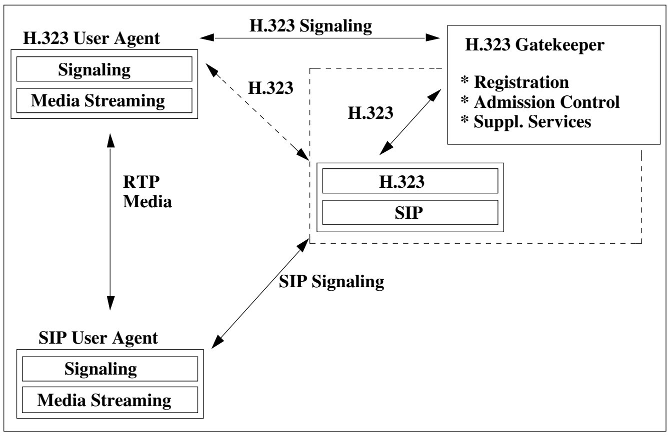

Fig. 1. Interworking between H.323 and SIP

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.

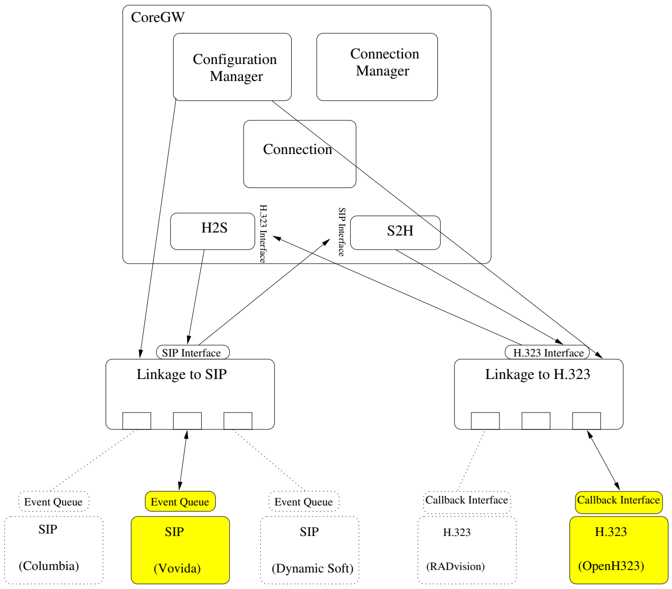

Fig. 2. SIP/H.323 Gateway Architecture

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, 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.

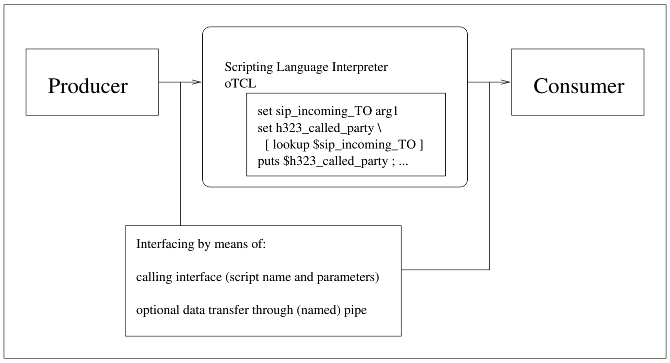

Fig. 3. Integrating scripting functionality

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-*

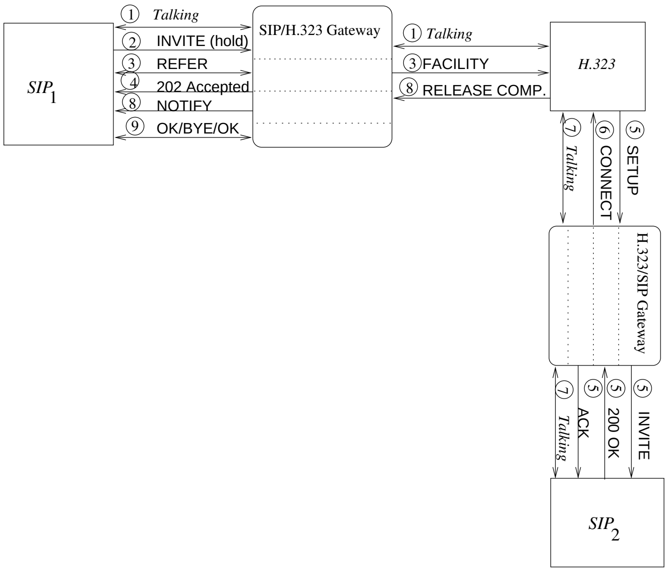

Fig. 4. Schematic scenario for Call Transfer

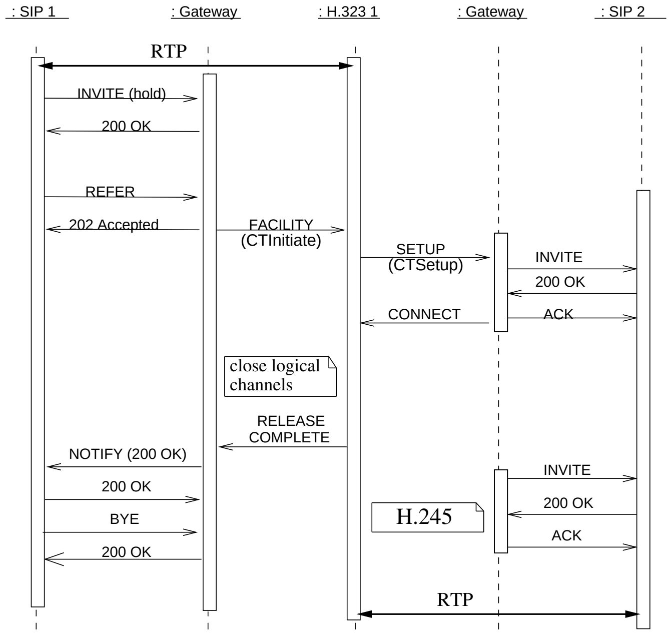

Fig. 5. Call Transfer signaling with gateways

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.