Service Architectures in H.323 and SIP – A Comparison

Josef Glasmann, Wolfgang Kellerer, Munich University of Technology (TUM)
Harald Müller, Siemens AG, Germany

Abstract

One of the major challenges for next generation IP networks is to provide multimedia teleconferencing services. Moreover, the introduction of new services includes the take over of traditional telephony. Besides the general problems with supporting realtime services in the IP network (e.g. QoS), this puts a special interest on the control of supplementary services and on the mechanisms for their fast and efficient development and deployment. The two most promising approaches in the multimedia over IP area are the protocol suites H.323 (ITU-T) and SIP (IETF). Several comparisons of these two protocols have been published already, but their service architectures have been rarely addressed. This survey provides a comparison of H.323 and SIP, focussing on their service architectures. The basic protocol architectures are reviewed, followed by an in-depth evaluation of their service implementation mechanisms. The studies are backed up by detailed examples for the control of supplementary services in H.323 and SIP. Although the two protocol architectures are quite similar, it is shown that there are considerable differences regarding their supplementary service architectures. H.323/H.450, which is still the more mature standard, has been focussed on supplementary services, smooth interworking with the PSTN and interoperability between different implementations. It has clear advantages for IP telephony applications. SIP has been designed with a broader scope, providing more generic syntax and semantics regarding feature definition and session description. Since the standards do not describe all details for broad range of possible applications and services, this bears the danger of interoperability problems. SIP has advantages for non-VoIP services. A coexistence of both protocols can be foreseen, stressing the importance of interworking between them.

1 Introduction

Fast and efficient development and deployment of new services are important drivers for advanced multimedia business. In the Internet world the standards of the ITU-T (H.323) and of the IETF (SIP) are of most importance for advanced telephony services. Although labeled with "Voice over IP" these architectures provide far more services than just setting up voice calls. The expectations and requirements are: Providing those services that are well known from traditional telephony and offering mechanisms to support the implementation and integration of new features. The comparison of the two standards in this paper focuses on their service implementation concepts.

Some studies have already been published comparing H.323 and SIP (e.g. [26]). They mostly refer to the basic call architectures and focus on issues like complexity and scalability. Paper [27] focuses on service aspects and investigates the usage of the two standards for an overall service architecture according to criteria derived from TINA. Up to now, concrete service implementation issues have not been discussed in detail.

In this paper, H.323 and SIP are compared according to the following criteria: standardization status, supported services, supplementary service architecture, proprietary extension and negotiation mechanisms, interoperability of services and features, interworking with GSTN, and service creation issues. Basic call features like call setup and session modification are distinguished from supplementary services. It is important to understand, that extensibility not only means being open to all kinds of extensions but also means implementation and interoperability issues. Quality of Service issues, network services like conferencing and addressing as well as aspects regarding the complexity and performance are not in the scope of this paper.

The remainder of the paper is structured as follows. In Section 2, a brief overview over the protocols' architecture and their standardization status is given, concluded by a discussion of the general similarities and differences of the two approaches. In Section 3, H.323 and SIP are investigated with respect to their principle methods for service implementation. A comparison highlights the significant differences that show up. Our statements are illustrated in Section 4 by means of concrete service examples. Finally, summarizing the results concludes this survey.

2 The Basic Protocol Architectures

2.1 H.323 Basic Protocol

The ITU-T started work on defining VoIP signaling protocols in May 1995. In December 1996, Study Group 16 passed the H.323 v.1, referred to as a "standard for realtime videoconferencing over non-guaranteed quality of service LANs" [1]. This recommendation describes terminals and other entities (Gatekeepers, Gateways, Multipoint Control Units) that provide multimedia communication over packet based networks. Support for audio is mandatory, while data and video are optional. A rapid standardization process and straightforward interworking with the PSTN have been the main goals from the very beginning. Some existing protocols could be reused directly (RTP, RTCP) others were derived from the ITU-T H.320 [2] protocol suite (H.245, H.225.0-CC) while the RAS

(Registration, Admission and Status) protocol had to be designed from scratch. H.323 v.1 defines the basic call control and signaling for setting up multipoint multimedia conferences. The basic call procedure comprises RAS signaling functions and call signaling functions. RAS signaling functions are required for endpoint registration, admission control and address resolution. Call signaling function includes connection setup, capability exchange and open logical channel procedures. Conferences in H.323 are normally tightly coupled and can be established out of a point-to-point connection via a multipoint controller which performs conference and floor control. To provide scalability an extension for loosely coupled conferences has been defined in H.332 [15].

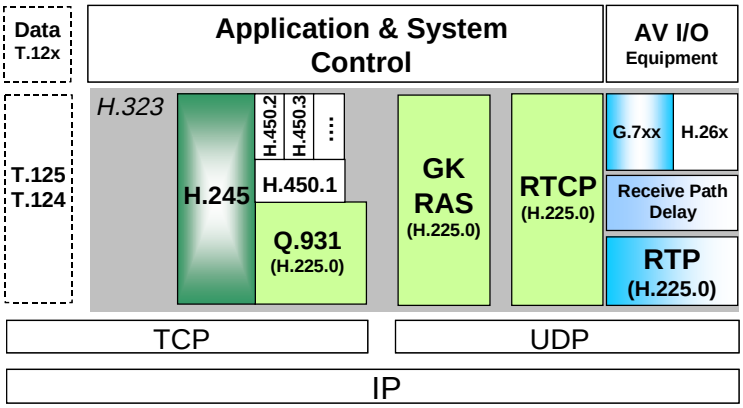

Figure 1: H.323 Protocol Suite

Among other enhancements, the version 2 of H.323 enables enhanced services on top of H.323. ITU-T SG16 evolved the H.450 series recommendations in order to support supplementary services over IP-networks. The scope of H.323 v.2 is depicted in Figure 1 and shows the optional (white) as well as the mandatory components (gray). H.450.1 defines a generic functional protocol on top of H.225.0-CC for all supplementary services [3]. It also defines the control procedures for the terminal equipment involved in handling the protocol messages. The most important features have been standardized already and new features are being added in an ongoing process. Table 1 shows the list of currently standardized and drafted features:

Standard	Feature Name	Subfunctions	Status
H.450.1	„Generic Functions for Supplementary Services in H.323“		Approved
H.450.2	„Call Transfer for H.323“	Single step transfer Transfer with consultation	Approved
H.450.3	„Call Diversion for H.323“	Call Forwarding Unconditional Call Forwarding on Busy Call Forwarding on No Reply Call Deflection	Approved
H.450.4	„Call Hold for H.323“	Near End Hold Remote End Hold	Approved
H.450.5	„Call Park and Call Pickup in H.323“	Directed Park and Pickup Group Park and Pickup Pickup of Alerting Call	Approved
H.450.6	„Call Waiting in H.323“		Approved
H.450.7	„Message Waiting Indication for H.323”	Unified Messaging System Message Waiting Call Back	Approved
H.450.8	„Names Identification for H.323“		Approved
H.450.9	„Call Completion for H.323“	Call Completion on Busy Call Completion on No Reply	Approved
H.450.10	„Call Offering for H.323“		Approval Pending
H.450.11	„Call Intrusion for H.323“	Conference Type of Connection Held Type of Connection Silent Monitoring Forced Release Wait on Busy	Approval Pending
H.450.12	„Common Information for H.323“		Draft

Table 1: List of currently standardized features in H.450

2.2 SIP Basic Protocol

The IETF Multiparty Multimedia Session Control Working Group (MMUSIC WG) has defined a control architecture for telephony features over wide area Internet that integrates stored and conference multimedia. The main component is the Session Initiation Protocol (SIP), which was specified by the MMUSIC WG as a proposed standard in 1999 (IETF RFC 2543, [17]). SIP provides advanced signaling and control functionality for a large range of multimedia communications. The main functions are: location of resources/parties, invitation to service sessions, and negotiation of session parameters. To fulfill this functionality, SIP provides a small number of text-based messages to be exchanged between the SIP peer entities (SIP user agent in a user terminal). Network entities like proxy servers or redirect servers that can be traversed by the messages, can be used for support, e.g. for address resolution. In this way, baseline SIP according to RFC 2543 includes all basic call control functionality in one signaling transaction using the INVITE request message.

The session itself is described in two levels. Whereas the SIP protocol contains the parties' addresses and protocol processing features, the description of the media streams that are exchanged between the parties of a multimedia session are defined by another protocol. The IETF suggests the Session Description Protocol (SDP, IETF RFC 2327, [18]), which in fact is not a protocol, but a structured, text-based media description format that can be carried in the SIP message body. Since the message body is transparent to SIP any session description can be transferred, including e.g. a weblink. In this way SIP sessions are not restricted to telephony style calls or conferences but can also include information retrieval or broadcast like sessions depending on the session description. SDP also allows to schedule session start and stop time or to describe recurrent sessions. In the following, we refer to SDP in the context of SIP unless stated otherwise.

In addition to the baseline SIP RFC, several IETF drafts complete the architecture regarding e.g. call control supplementary services, service and feature programming, conferences, preference management and interworking issues. In this way the IETF SIP IP telephony standardization is still work in progress and has not reached a final state yet – the RFCs and drafts are under continuous refinement by the separately set up IETF SIP WG.

In addition to the basic session setup that is specified in the SIP baseline RFC there is no standard of supplementary call control services other than some proposals in IETF drafts. The SIP baseline protocol provides some limited support for call control like call hold, media stream modification, or call termination, but the use of these features cannot explicitly be signaled as supplementary services.

The IETF has generally recognized the importance of advanced call control supplementary services. In July 2000 the SIP WG issued a Draft describing a framework for SIP call control extensions [19]. Up to now only a few supplementary services have been described based on this proposal [20],[21],[22].

Conferences in SIP are normally lightweight multicast conferences, to which a user can be invited. Some SIP extensions for the management of distributed multipoint conferences have been drafted, but have expired [25]. Advanced conference control like floor control or the support of roles is not in the scope of SIP.

In addition to the session processing using SIP, the IETF IPTEL WG proposes several possibilities for the programming of services either for administrators or for the users themselves [23]. This will be discussed in detail in one of the following sections. Figure 2 depicts the overall IETF SIP protocol suite. Work in progress extensions are included and the independence of SIP session signaling from audio/video media processing is shown.

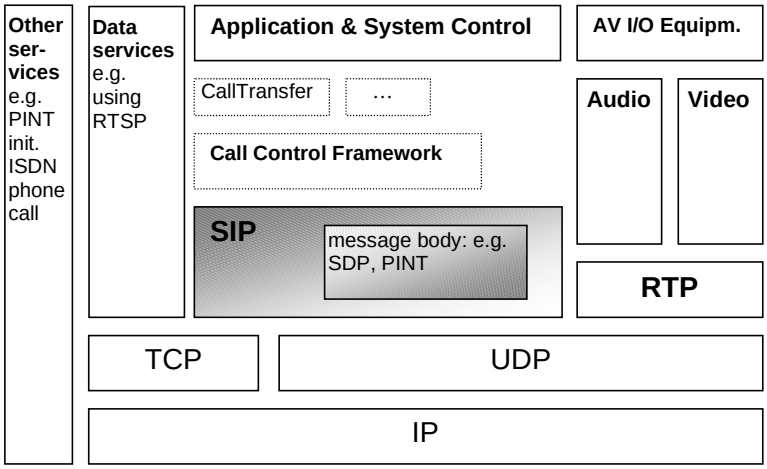

Figure 2: IETF SIP Protocol Suite

2.3 Comparing the Basic Architectures

The two VoIP architectures SIP and H.323 are based on similar concepts. The tables below show the components and protocols of the two approaches. H.323 and SIP comprise a lot of analogies with respect to function split and

service location. In SIP as well as in H.323, basic call and feature control are performed mainly in the terminals. For features requiring network support, servers (gatekeeper resp. proxy server,…) are foreseen in the networks.

It can be assumed, that H.323 and SIP will further converge in the near future, especially when SIP will obtain an adequate implementation status. When looking at the supplementary services, standardization maturity differs notably – with H.323 taking the clear lead. This makes it difficult to compare SIP with H.323. However, it can be anticipated that SIP takes a similar direction as H.323. Both define a general framework for call control features that defines a standardization process and rules for the implementation of new features.

	Client	**Servers in the Network**		
SIP (IETF)	Terminal	Proxy Server, Registrar	Conference Server*	Gateway*
H.323 (ITU-T)	Terminal	Gatekeeper	MCU	Gateway

*not standardized up to now

Table 2: SIP and H.323 Components

	Realtime Data Transmission	**Call Control**	**Feature Control**		**Signaling Procedure Variants**	
			Framework	**Extensions**		
SIP (IETF)	RTP/RTCP	SIP (SDP)	call control framework*	Transfer* Diversion* Message Waiting*	-	SIP-INVITE Transaction
H.323 (ITU-T)	RTP/RTCP	H.225.0, H.245	H.450.1	H.450.2 – H.450.11, H.450.12*	Basic Call Setup	Fast Connect

*Draft

Table 3: Protocols running on the Terminal

On the other side, H.323 becomes more lightweight like SIP. When looking at the H.323 call setup procedure, the ITU-T has additionally introduced the optional fast connect procedure and signaling via UDP. The fast connect is similar to the lightweight SIP session setup and combines the opening of a call control channel, the capability exchange and the open logical channel procedure in one single signaling transaction.

The substantial difference between the two protocols lies in their targeted range of application. SIP has been designed as a general transaction protocol for setup and tear down of generic sessions. Voice and multimedia is only one possible application of SIP. When not supporting voice or multimedia, a core SIP user agent can be very slim (e.g. only containing the SIP protocol and a generic session description). H.323 on the other hand, has been designed as a control protocol suite with the focus on multimedia applications including telephony. Naturally, because the scope is more restricted as compared to SIP, the range of application for H.323 is not as wide as for SIP. A couple of simple endpoint types (SETs) comprising only a well-defined subset of the full H.323 functionality have already been specified. But even the SETs are more complex as compared to SIP user agents. On the other hand, H.323 provides a more precise and detailed specification of the voice and multimedia functionality.

The focus of this paper is on comparing the service architectures of H.323 and SIP. The following chapters provide a deeper look into the service implementation process and compare the two approaches. This is backed up by explicit service examples.

3 The Service Architectures

3.1 H.323 Service Architecture

In this section, the three main models for supplementary service control currently existing in H.323 are described. They are: Distributed feature control (H.450), Stimulus feature control (H.323 Annex L) and Application layer feature control (H.323 Annex K).

3.1.1 Distributed feature control using H.450

Classification of features in H.323

Features in H.323 can be categorized into three classes: local features, network-based features and supplementary services.

Local features can be implemented in the endpoints without requiring specific signaling to other network entities. Examples for local features are: repeat a call, call history and call lists, local address book, speed dialing, privacy functions like do not disturb and mute, etc.

There are a couple of features that require centralized control. These network-based features are implemented in a centralized fashion in the gatekeeper (GK) or as a backend service behind the GK. Examples are authorization, address resolution, call admission, call detail recording, name/number suppression, etc.

The third category of features is the set of supplementary services (H.450). These are features that require special signaling between the corresponding entities. Examples for supplementary services are: call forwarding, call transfer, call completion, call hold, etc.

H.450 design philosophy

The H.450 supplementary service architecture uses a decentralized function split as far as possible. The peer entities (servers, clients, MCUs, GWs) communicate directly using H.450 signaling without involving centralized network control. For those features that require centralized control, a feature server is used, which is a special form of an H.323/H.450 endpoint. Examples where a feature server is required are: user not available proxy (acts on behalf of unavailable endpoints), messaging server, ACD (Automatic Call Distribution) server or Group-Server for group features.

Besides the fully distributed feature control, H.450 also describes a model where parts of the H.450 functionality can be carried out in H.450 proxies on behalf of the endpoints. The H.450 proxy can e.g. be collocated with the gatekeeper (GK).

One of the most important requirements for the design of H.450 was to simplify feature interworking with switched private (QSIG) and public networks (ISDN). Furthermore, H.450 was designed to be a highly extensible protocol as described in the next subsection. This includes also several mechanisms to ensure interoperability between endpoints with differing feature sets, which is a precondition for multivendor interoperability and smooth deployment of new features.

Extension of H.450

H.450 not only defines a set of supplementary services, it also provides several mechanisms that allow easy extension of this feature set. This subsection gives an overview of the H.450 architecture and its extension mechanisms.

H.450 supplementary service information is sent in H.450 APDUs (application protocol data units) that may be contained in any H.225.0-CC message. H.450 APDUs are exchanged between supplementary service entities and do not influence the underlying H.225.0 call state. Among other information, H.450 APDUs contain ROS (remote operations service) operations that define the semantics of the supplementary services. As with the other H.323 protocol components, H.450 APDUs are specified and coded using ASN.1 notation.

As in many parts of the H.323 protocols, the H.450 APDUs can be extended by manufacturer specific information (NonStandardData). This can either be additional information elements or even new operations. Using this extension mechanism, new supplementary services can easily be defined.

The standard H.450.1 (“Generic Functions”, GF) provides generic services for feature control that are common for all standardized and manufacturer specific supplementary services. H.450.1 provides call-related and call-independent transport of H.450 APDUs. Further, H.450.1 defines in a generic way how to proceed with H.450 APDUs that are not supported/known. This enables interoperability between endpoints with differing feature sets and a stepwise deployment of new supplementary services without having to support them in all endpoints at the same time.

Another standard that facilitates interoperability between heterogeneous endpoints is the emerging H.450.12 (“Common Information”). H.450.12 can be used to exchange the endpoints’ feature capabilities. This can be used e.g. to react in advance on these capabilities. One application for this is e.g. to not present to the user the opportunity to make a transfer if the other endpoint does not support it.

As shown in Figure 3, the architecture of H.323/H.450 enables a separation between basic call and the supplementary services. H.225.0 messages are routed to the basic call entity, where they trigger the respective actions and state changes. H.450 information is passed on to H.450.1. The generic services are carried out and the

ROS operations are passed on to the respective supplementary service entity. The GF is also the place where simultaneously activated features can be coordinated or even blocked in case of feature interactions. Each supplementary service entity has its own state machine that defines the semantics of the supplementary service. The supplementary service state machine is only invoked when a supplementary service is requested. The H.450.x standards specify these state machines using SDL diagrams.

This modular architecture is based on an object-oriented approach/principle and provides scalability with respect to features as it enables easy addition of new supplementary services. When creating a new feature, the new state machine is defined without the need for changing the state machines of the basic call and of other supplementary services. This even allows concepts for dynamic introduction of new features into running systems ("Feature pluggability"). In contrast, having only a single state machine including the supplementary services can grow very complex if many features are added.

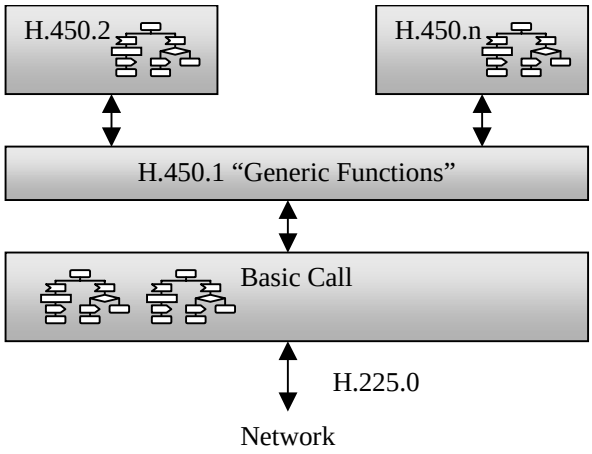

Figure 3: H.323/H.450 Architecture in the Endpoints

Building Feature Combinations for 1st Party Applications

One of the basic ideas of the H.450 features is to define them in such a way that they can be used together with the basic call as building blocks. By combining the atomic feature blocks, more complex features and services can be built. This allows to create a huge set of features by using only a small set of carefully designed building blocks. Applications and further features are built by using telephony APIs on the local machine (e.g. TAPI, JTAPI). Some examples of such combined features are:

- Consultation transfer = call hold + basic call + single step call transfer
- Messaging = basic call + call transfer to announcement server + DTMF control (basic call)
- Attendant console = basic call (multiple line) + call hold + call transfer

Building Feature Combinations for 3rd Party Applications

By introducing a CTI interface (Computer Telephony Integration) that hooks to the H.323/H.450 functionality in a remote endpoint, the H.323/H.450 building blocks may be remote controlled. This allows to create 3rd party call control applications. The functionality provided via the CTI interface may also include functions like monitoring endpoints (e.g. detect when a user is available, ...). The CTI interface is accessible via an API, a common protocol that may be used is CSTA (Computer Supported Telephony Applications). An example for a 3rd party application is ACD (Automatic Call Distribution). It can be built up by combining basic call, call transfer to a music/video server, monitoring an ACD agent using the CTI interface and CTI-initiated call transfer to an agent. Note that the CTI interface and protocol is an add on and is not specified in H.323.

3.1.2 Stimulus feature control using H.323 Annex L

Stimulus feature control is the opposite approach as compared to the decentralized H.450 architecture. A centralized feature server in the network is used to control the features in the endpoints. This approach is defined in H.323 Annex L. Endpoints conforming to H.323 Annex L still use functional signaling (H.225.0) for controlling the basic call. This yields basic call interoperability with fully functional H.323/H.450 endpoints. For centralized feature control, a so-called feature key management protocol is defined. It contains basic procedures to control user interface functions such as key presses, display messages and feature indicators (e.g. LED's). There is little intelligence in the endpoints, the feature logic and the semantic procedures are defined in the centralized feature server. This allows an easier deployment of features: only the feature server has to be updated, the endpoints do not

have to be changed. On the other hand, there are no standardized semantics for the features. The semantics are implementation dependent. Applications that want to use the feature functionality for 1st party call control need a functional interface (API), which can not easily be provided using the stimulus approach. A further downside of stimulus feature control is the scalability problem due to feature processing in centralized network components.

3.1.3 Application layer feature control using H.323 Annex K

H.323 Annex K defines an optional way for controlling services in H.323. It allows developing and deploying of new services without updating the H.323 protocol and endpoints. Annex K introduces a service plane above the H.323 call control plane. A service control session can be established after exchanging the relevant information (e.g. sessionID, URL for service control, ...) in RAS or H.225-CC messages. This mechanism allows for call- related and call-independent service control. Service control sessions can be maintained between endpoints or between endpoints and the network (e.g. GK).

The HTTP protocol is used in the service control channel to actually offer, select and activate the services. The service logic is described in HTML pages, scripts, etc. that are transferred via the HTTP protocol. Thus, features can be controlled from any device running a conventional Web browser (including e.g. PDAs).

Example applications may include transferring XML pages possibly including Java and scripts, download of tones and announcements or the upload of call processing scripts from a client to the GK. A concrete example scenario is explained in Section 4.3.2.

3.2 SIP Service Architecture

Like H.323/H.450, SIP feature control bases on a distributed feature control model. As SIP relies on true intelligent terminals, stimulus or transaction protocol based remote control is not addressed so far. For SIP the same feature categories can be applied as in H.323. Local features, network based features like authorization and address resolution in an outbound SIP proxy, and supplementary services. But, in defining supplementary services, SIP started with a different approach as compared to H.323 and standard telephony. Whereas traditional supplementary service implementation is explicitly standardized, SIP provides several elements (methods, headers) to allow the construction of services. Methods define SIP request messages like INVITE, whereas headers are identifiers for message parameters like receiver (*To*:) or route discriminators (*Route*:).

Recently, with the proposal of the call control framework [19] of the SIP WG this has changed as rules are defined how to explicitly identify SIP extensions for supplementary services in order to allow a standardized way of feature invocation and feature negotiation. In addition to that, SIP offers several different possibilities for programming of services with dedicated languages. Implementing services in SIP can in general be done by

- using SIP baseline protocol mechanisms,
- defining extensions (headers, methods - SIP call control framework), or by
- dedicated languages (SIP-CPL, SIP-CGI, SIP Servlets) for network server based service programming.

3.2.1 Supplementary services: baseline SIP and protocol extensions

Referring to the proposed standard (RFC 2543) there are no explicitly standardized supplementary services in SIP. RFC 2543 only describes a possible realization of the Call Hold Feature based on SDP parameters. In addition there are drafts showing sample sequence diagrams for certain features [24] like Call Hold, etc. Some of these supplementary services can be realized by the baseline SIP protocol functionalities i.e. by SIP requests and the transported session description, but they could not be explicitly signaled to the corresponding party.

For other supplementary services the definition of new headers and new methods has been done in several draft proposals. Whereas the definition of new protocol elements is in general applicable for any new feature, a recently issued draft proposes a special framework for call control extensions regarding features like Call Transfer, Conferences, Call Park/Pickup, and Call Monitoring [19]. The authors want to make sure that supplementary services are modularly defined and are separated to each other. This allows a standardized support of call control supplementary services negotiation. Until the writing of this survey the SIP WG has only drafted Call Transfer [20], Call Diversion Indication [21] and Message Waiting [22] based on this framework. This shows the evolution of SIP towards the H.323/H.450 supplementary services definitions.

For feature negotiation SIP provides are three possibilities. To make sure that new SIP headers indicating a new feature are known by the peer server in advance, SIP provides the *Require* header. Features identified by new headers are referenced by their name that has to be registered with IANA or by their reverse name of location (e.g. de.tum.ei.lkn.server) to guarantee correct functionality. For the negotiation of features based on new methods (=request messages) the *Allow* header can be inserted in a SIP request. The *Supported* header indicates to the server

which features are supported by the client. Unless the support of a feature is not explicitly requested (e.g. by *Require*:), unknown extensions are simply discarded by the receiving server. Otherwise an appropriate error message is sent (e.g. 501: Not Implemented) to allow the initiator to alter its request message.

Conveying a message body other than SDP may also be regarded as an extension. For example the message body may include a web page showing a map with the current location of the callee. Therefore SIP can also be used for other session signaling tasks, e.g. the PINT protocol [28] uses SIP to set up calls between two PSTN phones via an ISDN third party call setup gateway. Since these calls are normally terminated in the PSTN the SIP BYE message only ends the signaling relationship between the PINT entities. Paper [28] provides rules for the application of BYE in this special case.

3.2.2 Service Programming Languages

In general service creation in SIP can be done using any programming language. Service creation means to implement a service logic that either controls a specific message flow (cf. the feature combinations approach of H.450) or that reacts on a message request. Targeted for the implementation in SIP proxy servers the IETF has drafted several programming languages that can be used for the implementation of services started by message requests [23].

The IETF generally distinguishes between trusted and untrusted users. For untrusted users i.e. the end-users, the Call Processing Language (CPL) provides means how to handle SIP INVITE transactions. This means to decide whether an incoming request should be rejected, forwarded, or proxied. Addressing inexperienced users, the CPL-functions are very restrictive to avoid security and performance problems (e.g. loops).

Two approaches for service creation are proposed for trusted users e.g. server administrators: SIP-CGI and SIP Servlets. SIP-CGI Scripts are derived from HTTP-CGI scripts, but have a number of enhancements for supplementary service control like the generation of multiple responses or the ability to handle additional requests. Unlike HTTP-CGI, SIP-CGI Scripts can be reinvoked in order to manage a complete SIP transaction. Whereas SIP-CGI Scripts are independent from any programming language, SIP Servlets require a JAVA environment like Java Servlets. As it is well known from web programming SIP Servlets can be called by incoming messages and instruct SIP Servers how to handle requests.

In this way SIP provides a number of mechanisms for service programming that can all be easily applied since they are based on well-known programming methods. This is a great advantage compared to the proprietary programming methods of the traditional telephony service creation environments e.g. Intelligent Networks. For all service implementations on SIP proxy servers, it has to be made sure that the request traverses the responsible SIP proxy. Since IP networks are not route aware, SIP provides the *Route* header to determine the path a message has to take.

In addition to the programming methods described above that are aimed to program services with dedicated languages on a dedicated SIP server, a new approach suggests the use of java applets to be used in SIP requests [23].

3.3 Comparing the Service Architectures

The comparison of the implementation methods between the SIP and the H.323 service architecture in this survey is based on the following criteria: architecture, protocol extensions, message coding, service programming.

The key characteristic of the H.323 service *architecture* is its explicit definition of separate state machines for each feature independent from the basic call state machine. From the signaling point of view the function split of feature control into framework and extensions is a consequence of this separation. The ITU-T has already standardized the feature state machines of the most important telephony features, with further features being added every few months. The syntax is specified with ASN.1 and the semantics are described using SDL diagrams.

This elaborated and object-oriented approach is based on a long period of experience with the implementation and maintenance of telephony services. As a consequence, important challenges like the subsequent integration of new features into a running system, the interoperability with heterogeneous endpoints and the often neglected but very important field of feature interaction can be dealt with. By reusing the PSTN protocols, ease of interworking with already existing telephony systems (ISDN / Q.SIG) have been taken into consideration, too. Altogether, this approach allows very short product cycles.

The SIP WG on the other side provides only rudimentary implementation instructions. According to the SIP baseline standard [17], SIP features are not explicitly signaled and may even be hidden in the carried session description. However shortcomings in defining features have been recognized and a more stringent standardization process regarding the implementation of features has been postulated. The IETF has to make a strong effort in the near future in order to keep up with H.323.

For the negotiation of proprietary *extensions*, H.450 allows the transmission of individual rules inline with the request messages. These rules instruct the receiver what to do if the extension is unknown. In SIP on the other side only standard processing is foreseen. To identify features H.450 defines a hierarchical name space using vendor

specific extensions. Therefore no central authority is required for changes as soon as the vendor has an official vendor ID. This is quite different to SIP. SIP does not use vendor identifiers and every change has to be registered with IANA to securely avoid interworking problems.

Several *programming* languages are defined in the context of SIP for the programming of SIP servers. Although missing in H.323, they could also be applied for it. Since these programming languages are derived from the HTTP context they are more easily applicable for SIP as SIP is based on HTTP. To activate the appropriate service in the route unaware IP network, SIP can specify the proxy servers that must be traversed by a SIP message; in H.323 this is not possible.

In [26], the use of ASN.1 syntax for *coding* H.323 messages is criticized as an overhead in complexity contrary to the text based SIP-approach. Of course text based coding has advantages in rapid prototyping of individual solutions. A second point is the analysis of signaling messages. For example a network administrator can interpret the content of a message without an interpreter. On the other hand, when using ASN.1, there is a systematic support of the software development process. Tools for syntax checking and automatic code generation speed-up and ease the implementation of message processing functions. Further, the Packed Encoding Rules (PER) used in H.323 compress ASN.1 signaling messages very effectively.

4 Detailed Service Examples

After having discussed the characteristics and the functionality of the two IP telephony standards separately, detailed examples are presented to illustrate some significant differences between H.323 and SIP in the following. Two typical supplementary service examples (Call Hold and Call Transfer) and one non-VoIP service scenario (Click to fax using PINT) have been chosen.

4.1 Call Hold

The feature Call Hold allows the holding party A to interrupt the communication to party B during an active call. The signaling association between the two parties is not terminated. There are two basic variants of Call Hold: Near End Hold and Remote End Hold (in SIP: Far End Hold). With Near End Hold, the bearer channels remain open, but they no longer carry voice/video data from user A. Instead, they can be used to transmit media on hold (MoH), which is for example an announcement, a melody or a video clip. The second form is Remote End Hold, where the communication channels are idle during the hold condition. Endpoint B may play MoH to user B locally. We describe Remote End Hold in our example scenario. Using Call Retrieve procedures, the communication channels can be activated again and the call returns to the active condition.

4.1.1 SIP

The following example shows the SIP realization of a Far End Hold scenario. SIP compliant client/server implementations are assumed in the participating endpoints. Since proxy servers do not play a role in this example, they are not shown in the scenario. Figure 4 shows the Far End Hold Scenario. For completeness, the basic call setup has been included the message flows. SIP [17] defines call states only for call setup explicitly.

As there is no explicit message (SIP method or header) defined for the request of a Call Hold feature one has to rely on the basic SIP transaction mechanism. To set User B on hold, User A according to the SIP standard [17] re-invites User B (second INVITE message) with the connection address parameter of the connection attribute in the session description (SDP) set to zero (e.g. SDP: … c=IN IP4 0.0.0.0 …). This indicates endpoint B that A stops sending media streams. Endpoint B has to detect the request of the Call Hold feature from within the SDP carried as SIP payload. Another re-INVITE from User A with the original address parameter terminates Call Hold.

This ambiguous feature activation may cause severe implementation problems. It remains to the developers of the SIP user agents to recognize the “setting of the destination address for media streams to be put on hold to zero” [17] as a Call Hold request. There is no explicit feature invocation since the method name (INVITE) and the headers are the same as for a basic call. That means, that the same message with identical parameters my cause different reactions by the receiver depending on the actual context of the session. This approach will not scale for a large number of features. The SIP Working Group has already recognized this with the call control framework, which will be illustrated in the next section.

Another problem left open by the standard is the feature awareness. The status of User B being a held party and User A having set somebody on hold has to be remembered to insure robust call processing. Again, it remains to the implementers if it is the users themselves (or a higher layer proprietary application) or the (extended) SIP state machine to remember what feature(s) they are involved. The implementation of e.g. User B playing Music on Hold to her/himself depends on this decision.

The problem of feature (non-) awareness is even worse in combination with other features or in a more complex context, where Call Hold should be not allowed respectively carefully processed. Imagine being involved in a tightly coupled conference, controlled by a conference server. The correct function of the Call Hold service will depend on the feature awareness of the conference server. Since this is not signaled explicitly the whole conference group may be set idle or even worse in case of Near End Hold music is played to all parties. A typical feature interaction situation occurs.

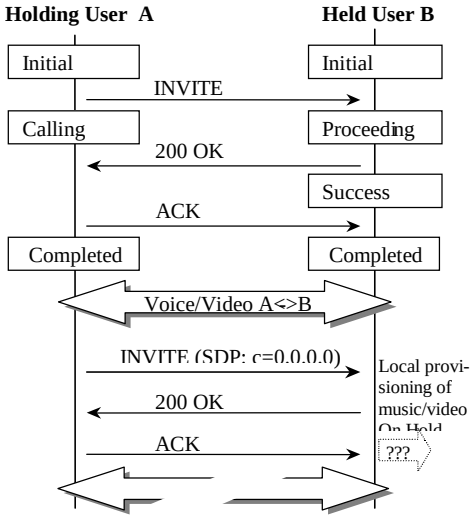

Figure 4: SIP: Far End Hold

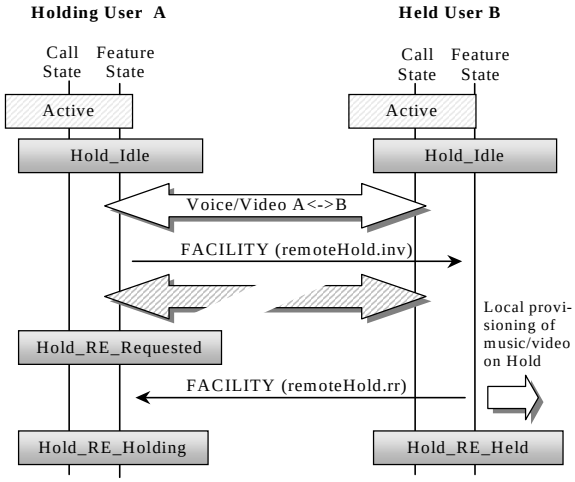

Figure 5: H.323/H.450.4: Remote End Hold

4.1.2 H.323/H.450

The following scenario describes the supplementary service Call Hold in H.323/H.450 as standardized in H.450.4 [6]. The examples assume the fully distributed model with H.450 implementation in the participating endpoints. The GK is transparent in this case, and is therefore not shown in the diagrams.

Figure 5 shows the Remote End Hold scenario in H.323/H.450. The two vertical lines represent the state machines for basic call and the feature state machine of each endpoint for the call hold feature.

In the beginning, the served user A has an active call with user B. User A pushes e.g. a hold button on his endpoint, which results in a FACILITY message containing a remoteHold.inv operation. This clearly identifies a remote end hold. At the same time, endpoint A interrupts the existing media (voice, video, ...) from/to B. No bandwidth is consumed any longer. The feature state of A changes to Hold_RE_Requested to remember locally that a remote hold has been initiated. The basic call state machine does not have to be changed when introducing the supplementary service, since the feature state machine defines the states and actions. Further, feature interaction is facilitated since all involved endpoints can determine by looking at their feature states that they are in a hold condition and whether new events and actions lead to unwanted interactions with other features.

Upon reception of the remoteHold.inv operation, B checks whether he can support remote end hold. In the positive case, endpoint B interrupts the media channels, too and provides local media on hold to the user. The confirmation of this action is signaled to A in a FACILITY message.

If B did not support the operation Remote End Hold, the rejection would be signaled to A. Note that B needs no support of H.450.4 to reject the request. The Generic Functions H.450.1 contain a generic mechanism that indicates the required actions when a requested operation is not supported (e.g. ignore, reject, ...). Even if Remote End Hold is supported by endpoint B, there might be situations where the hold operation should be rejected (e.g. if B is in a conference). This can be signaled including the respective reject cause.

4.2 Call Transfer

The feature Call Transfer allows a *transferring user* A to transfer an active call with *transferred user* B to a *transferred-to user* C. The outcome of the actions is that the previous call between A and B is cleared and a new call between B and C is in the active state. There are two basic forms of Call Transfer: Single Step Transfer (or Unattended Transfer) and Multi Step Transfer (or Consultation Transfer). The Single Step Transfer is described in the example.

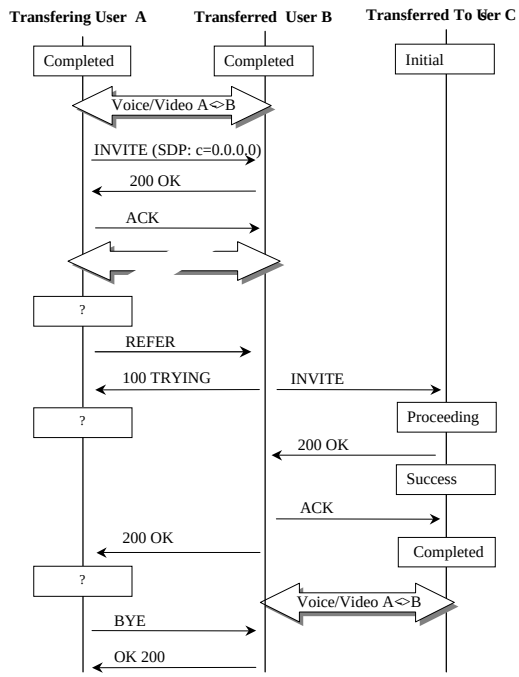

Figure 6: SIP: Single Step Transfer

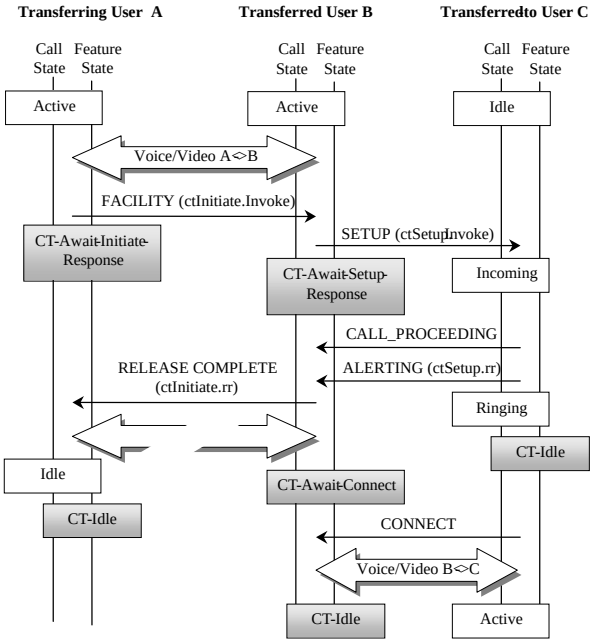

Figure 7: H.323/H.450: Single Step Transfer

4.2.1 SIP

Other than with Call Hold, for Call Transfer the IETF has recently issued a draft that defines a new method to be used explicitly to signal call transfer: the method REFER [20]. Figure 6 shows a sample message flow for an unattended (single step) transfer starting from an active call between A and B. A initiates the Call Transfer with setting the transferred user B on hold. When B accepts Call Hold, A initiates the transfer procedure by sending a REFER request with C's address to B. This indicates that B should invite C and issue a success response ('200 OK') to the originator, A in this case. On reception of a success response, A terminates its signaling relationship with B issuing a BYE request. When the REFER method is not supported by a SIP entity it simply returns an error message ('501 not implemented') to the initiator.

Other than within basic call based features (see previous section), the use of REFER clearly determines a Call Transfer supplementary service call. Nevertheless the IETF draft lacks important information for the implementation of this feature. We marked in Figure 3 points where call states have to change in order to provide appropriate message handling. The missing of a finite state machine in the standards causes interoperability problems with different implementations. But even if these new states are added to the basic call state machine the basic call would get more and more complex by adding more features. This causes scalability problems.

4.2.2 H.323/H.450

Figure 7 shows a Single Step Transfer according to H.450.2. Starting from an active call with endpoint B, the endpoint A initiates the transfer by sending a FACILITY message with the ctInitiate.Invoke operation. This operation contains the transferred-to address of party C. In the case of a multistep transfer, it would contain information about the call identity of the consultation call, which is required to associate the two calls at endpoint C.

Upon receiving the ctInitiate.Invoke operation, endpoint B starts a new call with party C using a SETUP message with a ctSetup.Invoke operation. This new call may inherit the media capabilities of the call with A or negotiate the media capabilities from scratch. The ctSetup.Invoke operation contains information about the transferring user A (transferringNumber), which can be displayed to the user or examined by other features and applications. It may

also be required for call admission and/or billing purposes. The H.450.1 GF would indicate to ignore the ctSetup.Invoke operation if it is not supported by endpoint C. This would result in a normal call setup taking place even if endpoint C did not support H.450.2.

After an ALERTING message containing a ctSetup.rr operation is received, endpoint B disconnects the first call with endpoint A. The user B will hear a ringing tone until user C goes off hook. Up to that point in time, the intermediate feature states in A, B and C allow to rollback the call transfer and restore the original call between A and B if anything fails. After the CONNECT message is received from endpoint C, the communication between B and C is established.

4.3 Non-VoIP Feature Example

4.3.1 SIP

In addition to the invitation of parties to participate in a multimedia session, SIP allows to initiate sessions that go beyond VoIP and are not bound to a specific media. Even non IP-based sessions e.g. a PSTN-call may be invoked by SIP. To illustrate this open character of SIP regarding session initiation, the PINT service protocol is explained in the following.

The PINT protocol (PSTN/Internet Interworking, RFC 2848511[28]) uses SIP and SDP for the invocation of telephony services in the GSTN from an IP network. All SIP extensions specified in PINT are in line with the SIP baseline behavior. The SIP INVITE message is used as a transport container for the assured exchange of service control information (e.g. a GSTN service description) between a PINT user (SIP client) and a PINT gateway (SIP server). The PINT gateway relays the request to a specific GSTN network control component and the latter performs the requested GSTN telephony service. Services scenarios are for example "click to dial" or "click to fax back".

Whereas the PINT user applies SIP to invite a remote PINT server into a session, which will be set up in the telephony network, the particular description of the telephone network session is carried as SDP payload in the INVITE message body. SDP has been enhanced with additional parameters for the support of new network types (ISDN, GSM, …), new media types (fax, image, …) and format specific attribute tags. The SDP session description is transparent for the SIP INVITE transaction and only the PINT gateway knows how to process. Figure 8 shows an example message flow for a "request to fax content" service.

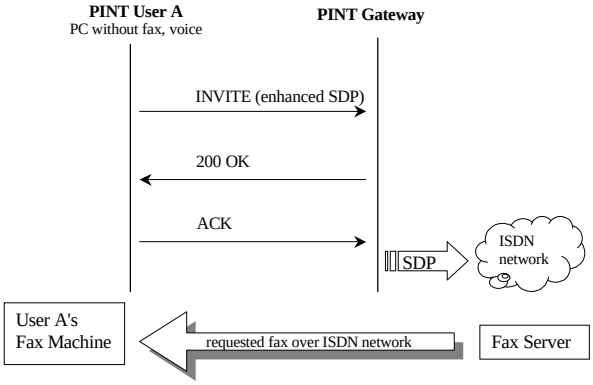

Figure 8. SIP: Click to Fax with PINT

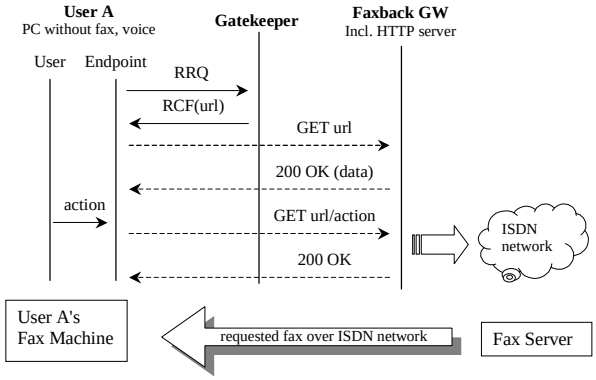

Figure 9. H.323: Click to Fax using H.323 Annex K

4.3.2 H.323

Although H.323 has put its focus on multimedia and voice services, it is also possible to provide non-VoIP services. In the following, an approach using H.323 Annex K is sketched which gives similar functionality as for the SIP/PINT example. With Annex K, it is possible to construct very simple endpoints for service control, containing only parts of the RAS protocol and an HTTP client (e.g. Web browser). Upon RAS registration with the GK, the endpoint receives a URL of the Faxback GW to contact for the service control session. Using HTTP, a selection of service options is presented to the user. W hen the user requests an action (e.g. Click to Faxback), this is again transferred to the Faxback GW using HTTP. The Faxback GW then carries out the respective actions towards the PSTN network (e.g. by using IN call control as in the PINT approach).

Compared to the SIP/PINT example, it becomes clear that the function split is different. Whereas SIP provides a transaction protocol to transmit a session description, in H.323 Annex K the transaction handling has to be programmed in the HTML (or other means) description of the service.

5 Conclusion

This survey has given an overview over the two VoIP standards H.323 and SIP, focussing on the service architecture and the mechanisms to develop and deploy services using the two standards. Although both protocols may be used for VoIP applications, their original focus is very different. While SIP has been designed as a generic transaction protocol for session initiation not bound to any specific media like audio or video, the focus of H.323 has been set to handle voice and multimedia calls including supplementary services.

As far as standardization of call control and supplementary services are concerned, SIP still shows some shortcomings regarding the number of standardized features and their implementation status. Currently SIP is going to be extended, in order to keep up with the respective functionality in H.323.

Comparing the service architectures and the resulting consequences for feature implementations, H.323 describes and enables an object-oriented approach based on QSIG, separating supplementary services from basic call control. The feature descriptions seen in SIP reveal, that SIP tends to extend the basic call to control supplementary services, which leads to migration and interoperability problems. In conclusion, H.323 provides better functionality, interoperability and interworking (PSTN and PBX) with respect to supplementary services.

SIP has advantages with respect to the design of low cost non-voice terminals. A SIP client does not have to support e.g. voice, SDP, capability exchange for data, and therefore very lightweight SIP clients can be built to control non-VoIP services. Therefore, SIP has its strengths for lightweight and easy to implement solutions with focus on flexible session initiation. SIP's broader scope forms the basis for a wider range of possible applications. H.323 had not been targeted to non-VoIP services in the beginning, but there are already some extensions to the standard in that direction: H.323 Annex K is applicable in that arena with some limitations. Summarizing, SIP provides better mechanisms for controlling non-VoIP services.

With the focus set on supporting voice and multimedia over IP including supplementary services, H.323 is the better choice for IP telephony applications. This includes replacement scenarios for legacy PBXs, but is especially true when IP telephony supplements and coexists with legacy telephone systems. Although this may sound like a typical application for enterprise scenarios, the trend of outsourcing applications (ASP solutions) can also be observed for IP telephony. Thus, supplementary services and H.323 become more important for carrier implementations, too.

Although the two standards are approaching each other, their focus and applicability is still different. It can be expected that neither of the two protocols will succeed over the other. They will probably coexist in different environments and implementations over a longer time, putting also a strong requirement on interworking between them.

6 References

[1] ITU-T Recommendation H.323, Packet-Based Multimedia Communications Systems. 1998.

[2] ITU-T Recommendation H.320, Narrowband Visual Telephone Systems and Terminal Equipment. 1993.

[3] ITU-T Recommendation H.450.1, Generic Functional Protocol for the Support of Supplementary Services in H.323, 1998.

[4] ITU-T Recommendation H.450.2, Call Transfer Supplementary Service for H.323, 1998.

[5] ITU-T Recommendation H.450.3, Call Diversion Supplementary Service for H.323, 1998.

[6] ITU-T Recommendation H.450.4, Call Hold Supplementary Service for H.323, 1999.

[7] ITU-T Recommendation H.450.5, Call Park and Call Pickup Supplementary Services for H.323, 1999.

[8] ITU-T Recommendation H.450.6, Call Waiting Supplementary Services for H.323, 1999.

[9] ITU-T Recommendation H.450.7, Message Waiting Indication Supplementary Service for H.323, 1999.

[10] ITU-T Recommendation H.450.8, Name Identification Supplementary Service for H.323, 2000.

[11] ITU-T Recommendation H.450.9, Call Completion Supplementary Services for H.323, 2000.

[12] ITU-T Draft Recommendation H.450.10, Call Offer Supplementary Service for H.323, 2000.

[13] ITU-T Draft Recommendation H.450.11, Call Intrusion Supplementary Service for H.323, 2000.

[14] ITU-T Draft Recommendation H.450.12, Common Information Additional Network Feature for H.323, 2000.

[15] ITU-T Recommendation H.332, H.323 Extended for Loosely Coupled Conferences, 1998.

[16] M. Korpi, V. Kumar. Supplementary Services in the H.323 IP Telephony Network. IEEE Communications Magazine, July 1999, p.118-125.

[17] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg. SIP: Session Initiation Protocol. IETF Request For Comments RFC 2543, March 1999.

[18] M. Handley, V. Jacobson SDP: Session Description Protocol, Request for Comments: 2327, April 1998.

[19] B. Campbell. Framework for SIP Call Control Extensions. IETF Internet Draft, draft-ietf-sip-cc-framework-00.txt, March 2000. Work in Progress.

[20] R. Sparks. SIP Call Control - Transfer. IETF Internet Draft, draft-ietf-sip-cc-transfer-03.txt, February 2001. Work in Progress.

[21] S. Levy, B.Byerly, J.R. Yang. Diversion Indication in SIP. IETF Internet Draft, draft-levy-sip-diversion-01.txt, November 2000. Work in Progress.

[22] R. Mahi, I. Slain. SIP Event Package for Message Waiting Indication. IETF Internet Draft, draft-mahy-sip-message-waiting-01.txt, February 2001. Work in Progress.

[23] H. Schulzrinne, J. Rosenberg. The Session Initiation Protocol: Internet-Centric Signaling. IEEE Communications Magazine, October 2000, pp. 134-141.

[24] A. Johnston, R. Sparks, C. Cunningham, S. Donovan, K. Summers. SIP Telephony Service Examples. Internet Draft draft-ietf-sip-service-examples-00.txt, November 2000. Work in progress.

[25] J. Mark, K. Kelly. Distributed Multipoint Conferences using SIP. Internet Draft, draft-mark-sip-dmcs-00.txt, March 2000. Work in progress. expired

[26] H. Schulzrinne, J. Rosenberg. A Comparison of SIP and H.323 for Internet Telephony. In Proceedings of NOSSDAV, Cambridge, U.K., July 1998.

[27] R. Glitho. Advanced Service Architectures for Internet Telephony: A Critical Overview. IEEE NetworkMagazine, July/August 2000, pp. 38-44.

[28] S. Petrack, L. Conroy. The PINT Service Protocol: Extensions to SIP and SDP for IP Access to Telephone Call Service. IETF RFC 2848, June 2000.

Biographies

JOSEF GLASMANN (Glasmann@ei.tum.de) received his diploma degree (Dipl.-Ing. Univ) in electrical engineering and information technology from the Munich University of Technology (TUM), Germany, in 1996. Since October 1996 he is a member of the research and teaching staff at the Institute of Communication Networks (Prof. J. Eberspächer) at TUM. His research topics are QoS in IP networks and multimedia service architectures. Currently he is working on his doctoral thesis, which is about the design and implementation of a resource management architecture for VoIP in corporate networks.

WOLFGANG KELLERER (Kellerer@ei.tum.de) received the Dipl.-Ing. Univ. degree in electrical engineering and information technology from the Munich University of Technology (TUM), Germany, in December 1995. Since January 1996, he has been a member of the research and teaching staff at the Institute of Communication Networks (Prof. J. Eberspächer) at TUM. His research interests are service architectures for multimedia services in the context of telecommunication, information and broadcast convergence. Currently he is working on his PhD/Dr.-Ing. degree with the subject of a network independent server architecture for flexible service control.

HARALD MUELLER (H.Mueller@icn.siemens.de) received his diploma degree (Dipl.-Ing.) in 1990 from the Darmstadt Technical University, Germany. In 1996 he received his doctor's degree (Dr.-Ing.) from the Munich University of Technology, Germany in the area of Multimedia Signaling. He joined the Siemens AG, Germany, where he is currently leading a product and system definition group for IP-based realtime communication systems for enterprise customers. His current areas of interest include multimedia signaling for IP-based systems, QoS in IP networks, mobility aspects in IP networks and convergence of IP-based and TDM-based communication systems.