Using DMIF for abstracting from IP-Telephony Signaling Protocols

R. Ackermann[1], V. Darlagiannis[1], U. Roedig[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)

Abstract. IP Telephony recently finds a lot of attention and will be used in IP based networks and in combination with the existing conventional telephone system. There is a multitude of competing signaling protocol standards, interfaces and implementation approaches. A number of basic functions can be found throughout all of those, though. This includes the addressing of participants using symbolic names, the negotiation of connections and their parameters as well as the enforcement of a dedicated handling of data streams by means of QoS signaling activities. Thus, a generic abstraction hiding underlying protocol specifics is very desirable and useful. The Delivery Multimedia Integration Framework DMIF - as part of the MPEG approach towards distributed multimedia systems - forms a general and comprehensive framework that is applicable to a wide variety of multimedia scenarios.
In this paper we describe a more generalized and abstract view to basic IP Telephony signaling functions and show how these can be hidden below a common DMIF interface. This will allow for the implementation of inter-operable applications and a concentration on communication functionality rather than protocol details. We expect that this will also allow for better exchangeability, interoperability and deployability of emerging signaling extensions.

Keywords: Internet Telephony, Signaling, SIP, H.323, MPEG-4, DMIF

1 Introduction

IP-Telephony applications are considered to have a huge economic potential in the near future. Because companies and service providers start to consider it to be getting ready for carrier-grade usage, it may also speed up the deployment of state-of-the art QoS, security and billing components in local as well as in the backbone networks. Though IP-Telephony might be seen as (just) a specific application today, it is part of an ever emerging scene of more general multimedia applications. Considering the high dynamics and multitude of concurrent approaches in signaling protocols, interfaces and implementations, a consistent and comprehensive framework can speed up development and allows a faster, better and more generic implementation as well as the interoperability, exchangeability and re-use of modular components.

2 IP-Telephony Signaling Protocols and APIs

IP-Telephony signaling protocols are used to establish a conversation comparable to a classic telephone call using an IP infrastructure. Typical applications and scenarios are recently based on different protocol suites. Mainly, there are two major approaches - the H.323 [6][7] protocol family and the Session Initiation Protocol SIP [4] with a changing distribution and relevance. Though today, a high percentage of applications and scenarios is still H.323 based (and we will therefore initially focus on it), it is supposed that in the near future the use of the SIP protocol may increase [3][13]. Both protocol types will even be usable together with appropriate gateways [?][14]. Additionally, the close interaction with the existing Public Switched Telephone Network (PSTN) on the basis of interacting media gateways plays a very important role. For that domain, protocols like MGCP [2] / H.248 [5] that describe the interaction towards the PSTN SS#7 [12] and may use both H.323 and SIP for signaling within the IP telephony world are under recent development and standardization.

2.1 Signaling using the H.323 Protocol Family

The H.323 protocol suite compromises a variety of communication relationships, which are handled via dynamically negotiated channels for a number of H.323 protocol components such as RAS, Q.931, H.245. These use Protocol Data Units that are encoded as described in ASN.1 specifications. Though the H.323 protocol suite has proven to provide the intended communication services especially for usage in LANs, it is considered to be complex, not easy to extend and having a considerable signaling overhead that can not be neglected in a global environment.

2.2 Signaling using the Session Initiation Protocol SIP

The Session Initiation Protocol SIP has initially been used as a protocol for multicast applications and provides generic control functionality. Its basic operations which are directly related to call setup are registration of participants and redirection or proxying of control data traffic. This allows to access telephony services through single points of contact, may hide infrastructure aspects and is also applicable for building hierarchies. Additionally to its primary function, SIP allows to control call proceeding and additional services in a very generic, efficient and extensible manner, e.g. by means of Call Processing Language (CPL) scripts. SIP protocol functionality can either be provided by centralized components but also at "smart" end system nodes. Over the last period of intensive work, SIP has emerged towards the core protocol of a comprehensive framework, addressing additional features such as QoS support, firewall interaction and call routing as well.

2.3 Application Interfaces - TAPI and JTAPI

The Microsoft Telephony API (TAPI) [11] has been developed in a joined effort by Microsoft and Intel and is provided as part of Win9x and WinNT. Its targets are to isolate features of the underlying hardware from the applications by means of a standard API as well as to specify a Telephony Service Provider Interface that the underlying services have to meet.
It supports basic Computer Telephony Integration (CTI) applications like automated dialing but starting with the TAPI version 3.0 under Windows 2000 involves sophisticated features such as IP multicast conferencing, a H.323 stack and Interactive Voice Response (IVR) functionality as well. Inherently it is limited to the Windows platform though and does not up to now cover SIP, though protocol descriptions state, that it provides powerful means to incorporate new protocols or protocol extensions as so called Third Party Service Providers.
The Java Telephony Application Programming Interface (JTAPI) [15] is an object-oriented interface that allows the development of portable telephony applications in Java. It uses a modular approach that places additional functionality, so-called extensions, on top of a common JTAPI core. The API itself just describes interfaces which have to be implemented for the underlying hardware or protocol infrastructure. As a current drawback it must be stated that there is still only a small though rising number of JTAPI peer class implementations.
Both APIs have current limitations and are inherently targeted at telephony services. We do not consider JTAPI or TAPI as comprehensive alternatives or competitors to our approach - they can even be combined with it or provide services.

3 The abstraction Framework

3.1 MPEG and DMIF

MPEG-4 [10] is a new multimedia standard that is much more powerful and comprehensive than the previous MPEG standards. To begin with, it provides an object-based description of content, which can be both naturally captured and computer generated. Though the term MPEG-4 is often associated with the specification of a set of video codecs working with individual visual objects, the standard is much more comprehensive. Among others, MPEG-4 defines the Delivery Multimedia Integration Framework (DMIF) [8][9]. DMIF is a framework that abstracts and thereby encapsulates the delivery mechanisms from the applications.
The frameworks API, called DMIF Application Interface (DAI), works with Universal Resource Locators (URLs), which specify appropriate delivery mechanisms for specific scenarios. URLs can also specify the required network protocol, which provides a protocol abstraction for the applications. Additional parameters of a connection such as e.g. Quality of Service (QoS) requirements can be passed as arguments through this generic interface as well. DAI is language and platform independent. A basic description of its primitives is given in Table 1.

Primitive	Description
DAI_ServiceAttach	Allows the initialization of a service session with a remote peer, specified with a URL.
DAI_ServiceDetach	Allows the termination of a service session.
DAI_AddChannel	Allows the establishment of end-to-end transport channels in the context of a particular service session.
DAI_RemoveChannel	Allows the replacement of existing transport channels.
DAI_UserCommand	Allows the application-to-application exchange of messages.
DAI_SendData	Allows the transmission of media in the established channels.

Table 1. DMIF primitives and functionality

Additionally, DMIF defines an informative DMIF-Network Interface (DNI) for the network related scenarios. DNI allows the convenient development of components that can easily adapt their signaling mapping to different protocols.

3.2 Framework Architecture

Before defining the basic objects of our architecture, it is important to identify main use cases, which are typically required by IP Telephony applications. Basically a user should be able to register himself in the "IP-networked world", to enable his locating and identification for other participants. After that, it is possible to receive calls or to originate them to remote users. So, in a necessarily limited scenario there are three main use cases, which are shown in Figure 1, using a UML use case diagram.

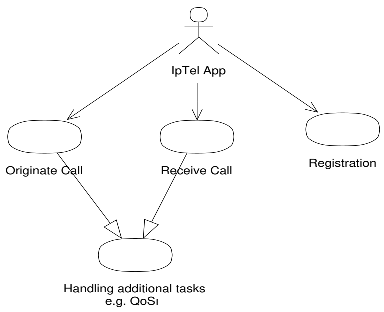

Fig. 1. *Basic Use Cases*

As an example for an additional service, we refer to the signaling for ensuring the desired QoS.

From the analysis of the use cases we derive the architecture of the proposed framework. It is shown in Figure 2.

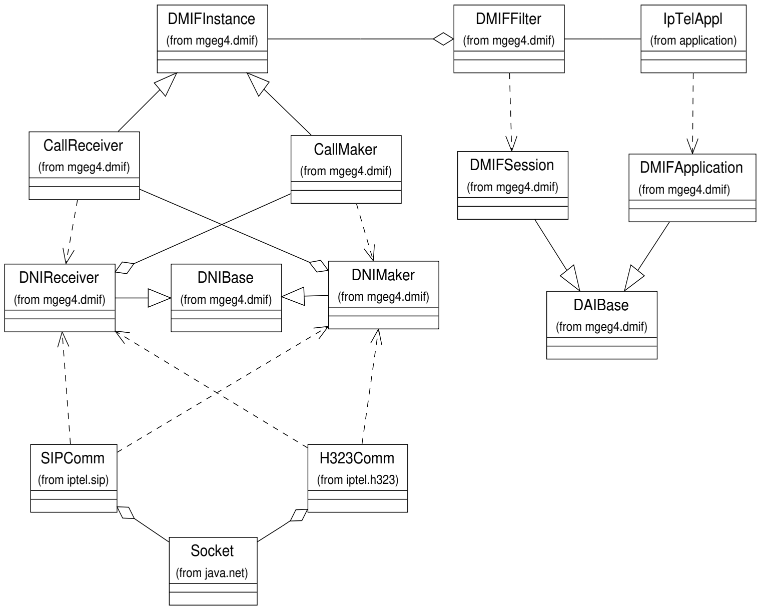

Fig. 2. *Framework Architecture*

In our framework the DMIF Application Interface is implemented with two interfaces, DMIFSession and DMIFApplication. The first provides the set of methods that are offered to the application from the DMIF layer, while the former is the set of callback functions for the DMIF layer to inform the application about events and messages. The DMIF layer is provided to the applications through the DMIFFilter. Its responsibility is to parse application requests and to activate the appropriate DMIFInstances to handle them.
The DMIFInstances are formed of two different objects: the CallReceiver and the CallMaker. A CallReceiver object initially allows the user to register himself. It is then responsible for the acceptance of incoming calls as well as for their handling. A CallMaker executes the requests for outgoing connections. Both CallReceiver and CallMaker behavior is independent of the used signaling protocol. They communicate with the appropriate signaling object using the DNI interface. Specific implementations of signaling protocols are the SIPComm object, which uses the SIP protocol and the H323Comm object, which uses the H.323 protocol suite.

4 Usage Scenarios and Protocol Mappings

After having identified basic functions we now show how calls can either be received or originated using both H.323 or SIP as the underlying signaling protocol. The protocol details are hidden from the applications which use the same interface and primitives in any of the cases. They - using the appropriate URL identifier - just have to specify which of the available protocols should be used .

4.1 Registering and receiving calls using H.323

In this scenario, the IP Telephony Application (IpTelAppl) uses the H.323 protocol to register the participant with a Gatekeeper and enables him to accept calls. The sequence of protocol steps is shown in Figure 3.

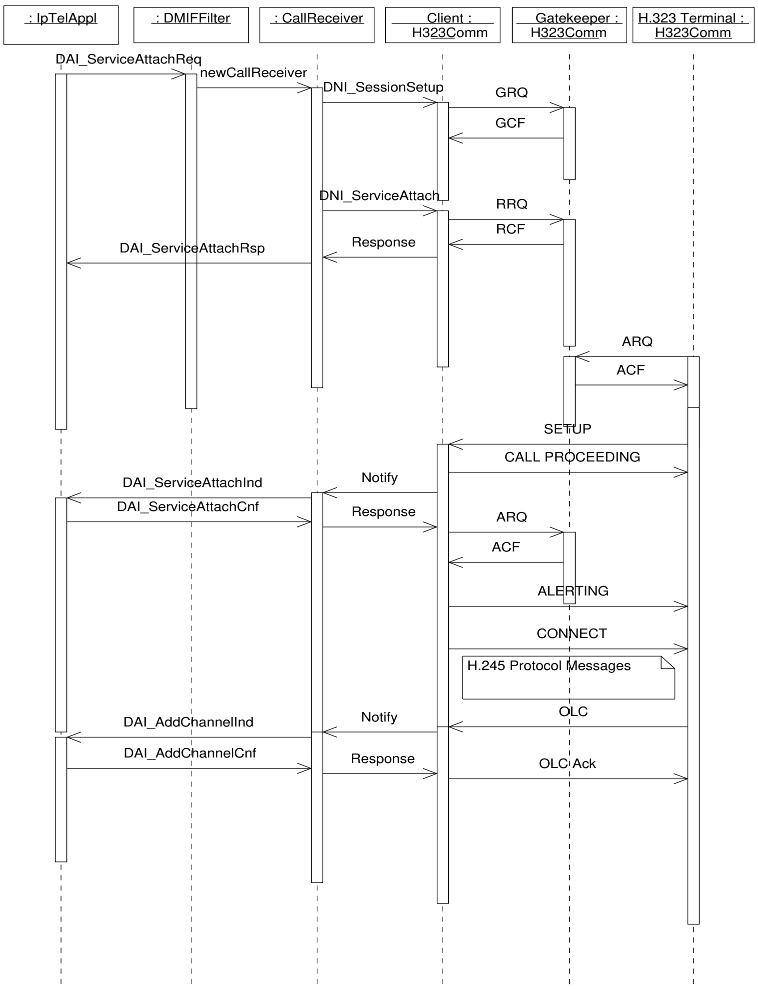

Fig. 3. *Registering and receiving calls - H.323 scenario*

It should be noticed that the DAI interface is used between the IP Telephony application and the DMIFFilter and the DNI between the CallMaker and the H323Comm object. The IpTelAppl attaches to the local Gatekeeper, by using the DAI_ServiceAttachReq primitive, where it can pass the address of it, if it is known. The DMIFFilter, parses the passed URL and activates a CallReceiver object to handle the details of the operation. The CallReceiver may inquire at the local H323Comm object about the location of the Gatekeeper, if its address is not already specified, using the DNI_SessionSetup primitive. In this case the local H323Comm object broadcasts a request to locate the Gatekeeper. After the Gatekeeper has been identified, the CallReceiver object requests the H323Comm object to register itself at the local Gatekeeper. The local H323Comm object is completing the request, by exchanging one more pair of messages with the Gatekeeper (RRQ and RCF). After the successful completion of the registration task, a handler to the CallReceiver is returned to the IpTelAppl for further usage.

Suppose that later another H.323 terminal wants to call the IpTelAppl. It can obtain the address of the local H323Comm object from the Gatekeeper. Then, it submits a Setup message to the local H323Comm object to request the establishment of a new session. The IpTelAppl is informed about this request from the CallReceiver object. The IpTelAppl instance can decide to accept the new call and the local H323Comm object requests admission from the local Gatekeeper (ARQ and ACF messages). After that it replies to the remote H.323 terminal that it accepts the connection (CONNECT message).

A number of H.245 messages follow in order to exchange the capabilities of the two terminals. Then, an Open Logical Channel (OLC) message is sent to request for a media channel. The CallReceiver indicates this to the IpTelAppl (DAI_AddChannelInd), which confirms it. Finally, the local H323Comm object sends an acknowledge to the remote H.323 terminal. The last procedure might be repeated for more media channels.

4.2 Registering and receiving calls using SIP

Figure 4 shows the registration of an IpTelAppl in order to receive calls for the SIP case. The IpTelAppl requests user registration with the DAI_ServiceAttach command. The DMIFFilter therefore creates a new DMIF Instance, the CallReceiver to handle the registration and possible future calls. A CallReceiver requests from the SIPComm object to establish a connection with the SIP location sever. The SIPComm object sends a REGISTER message to the Location Server to store its location information for future incoming calls. The handler returned to the IpTelAppl is used to proceed future interactions.

Later, when a new INVITE message is received from the SIPComm object (Client instance), the IpTelAppl will be informed. It then can either confirm the acceptance of the incoming call or reject the new invitation. In the case of acceptance, a SIP 200 OK response is replied to the caller, and the call is established after the final ACK is received.

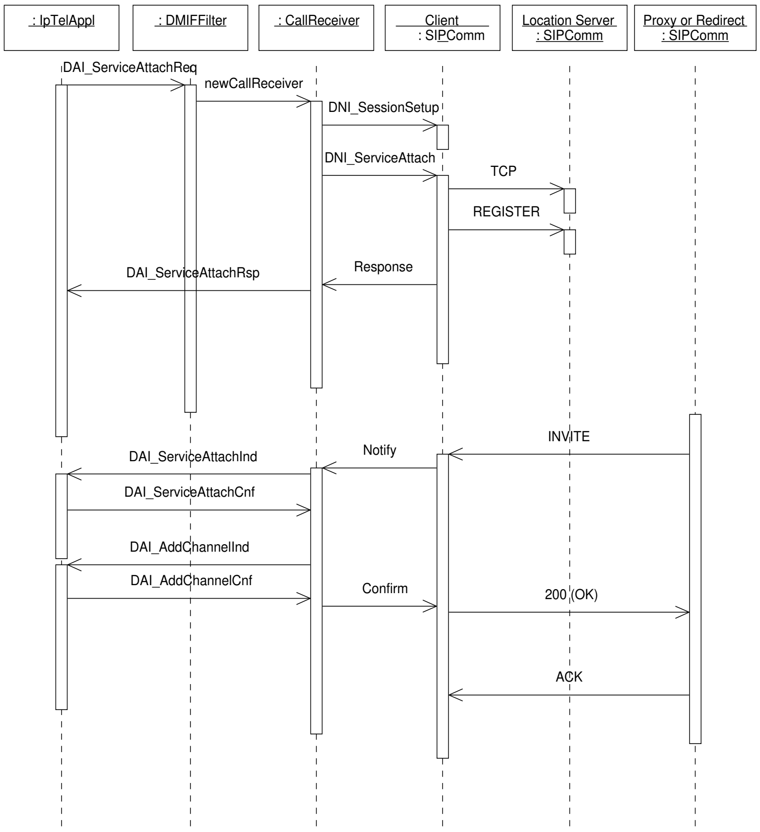

Fig. 4. *Registering and receiving calls - SIP scenario*

4.3 Call Setup using H.323

In this scenario - shown in Figure 5 - the IpTelAppl wants to setup a call to a remote H.323 terminal. Only the most important and relevant (to the DMIF layer) H.323 messages are shown.
The IpTelAppl calls the DAI_ServiceAttachReq to originate a new call. It passes the URL of the remote participant for symbolically addressing the intended communication partner. The DMIFFilter parses the request and creates a new CallMaker object to handle the details of this operation. It requests the session setup from the local H323Comm object, which communicates with the Gatekeeper to request for admission to place the call and to address the remote party (ARQ and ACF messages). Then, the CallMaker calls the DNI_SessionAttach primitive to request the establishment of a new session with the remote terminal from the local H323Comm object. A set of messages is exchanged between the two H.323 peers, compromising both Q.931 and H.245 protocol elements. At the end, the IpTelAppl receives a positive response.
Once the connection is established, the application may initiate the setup

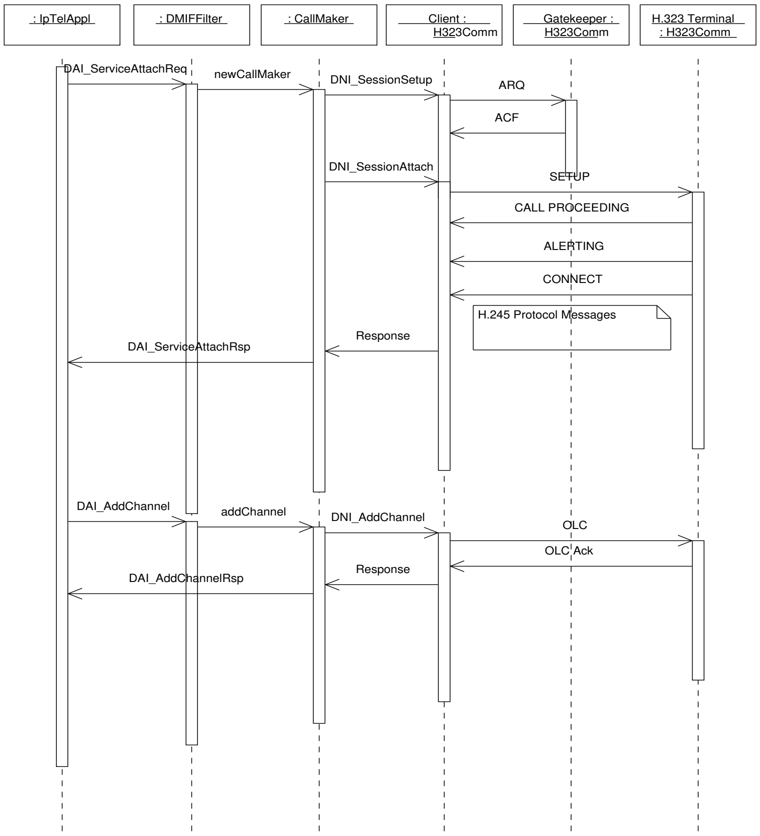

Fig. 5. *Originating Calls - H.323 scenario*

of additional media channels with the DAI_AddChannelReq primitive. The local H323Comm object is negotiating the channels with the remote H.323 terminal (OLC and OLC Ack messages). This procedure is repeated between the two terminals for every media channel.

4.4 Call Setup using SIP

In this scenario - described in Figure 6 - the IP Telephony application is setting up a new call, using the SIP protocol. No specific details of the protocol are required for the application.
It constructs an appropriate URL, which denotes the intended usage of the SIP service. The IpTelAppl requests the DMIFFilter to attach with the requested remote participant. The symbolic address of the remote user is passed as a SIP URL with the appropriate parameters. The DMIFFilter creates a new CallMaker object to handle the signaling task for this new call. The CallMaker interacts with the SIPComm object (Client instance) to establish the call (if TCP is used) with a SIP proxy

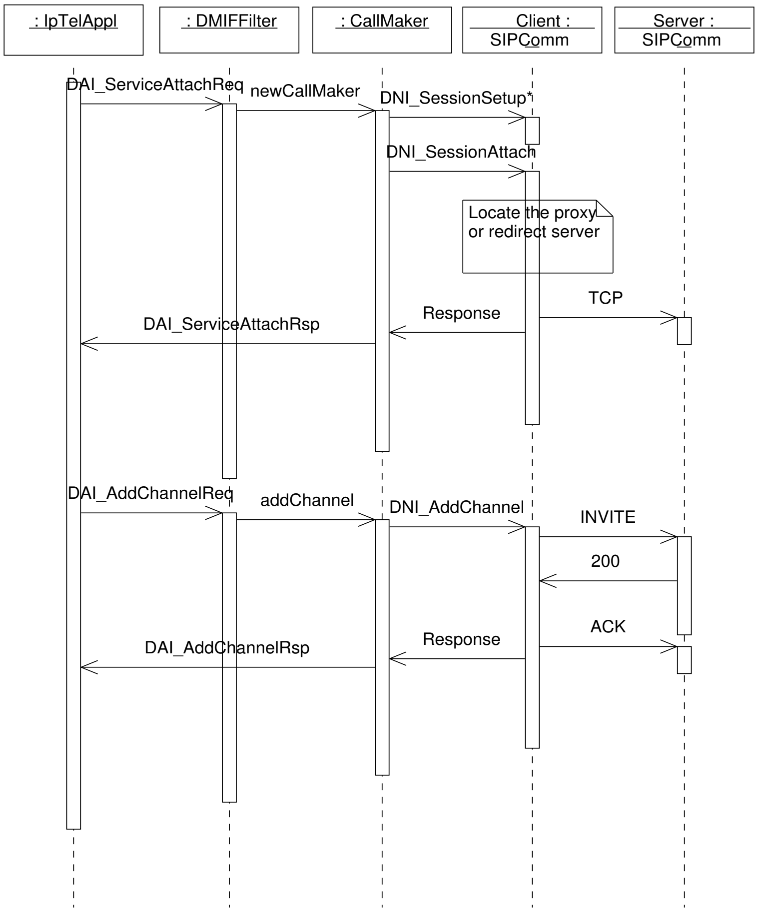

Fig. 6. *Originating Calls - SIP scenario*

or redirect server, using the DNI_SessionAttach primitive. The Client has to locate the appropriate SIP proxy or redirect server first to request the remote user. After the (successful) connection with the server a response is given back to the IpTelAppl, with a handler to refer to the same objects for later requests.

In a next step the IpTelAppl may add a voice channel. IpTelAppl calls the DAI_AddChannel primitive from the DMIFFilter to request for a channel with specific QoS, if this is supported from the underlying network. The previously created CallMaker object is identified and is requested to handle the new request. The CallMaker maps the Application QoS to the Network QoS and calls the DNI_AddChannel primitive to ask the SIPComm object to request for a new channel with the remote user. SIP-Comm, interacts with the SIP proxy and redirect server to locate and invite the remote user, using the INVITE message. In the basic scenario, it will receive a positive SIP response (OK 200), and will complete the invitation with the SIP ACK message.

5 Conclusion and Future Work

In the paper we have proposed a framework based on DMIF and possible mappings for common IP Telephony signaling operations. We are not intending to develope "yet another implementation" for one of the emerging signaling standards, but try to find a generic approach by identifying basic functionalities.
Based on our experiences with implementing a protocol gateway between H.323 and SIP the "conventional way" [1] we assume, that the approach fits well with the recent functionality of established signaling protocols while being flexible enough to also incorporate changes or cope with even new protocols. Though we have concentrated on describing scenarios involving end systems, it is applicable for the development of infrastructure components using a variety of signaling protocols as well.
The basic motivation for choosing DMIF is its standardization and the experience, that generating software in a standardized instead of a per-application or per-protocol way can speed up development and permits more generalized solutions. We consider our framework feasible and intend to implement it using different underlying protocol stack software thus enabling applications to use the described interface and to have means for the evaluation of its performance and implementation as well as runtime-overhead.

References

1. R. Ackermann, V. Darlagiannis and R. Steinmetz: Implementation of a H.323/SIP Gateway. Technical Report TR-2000-02, Darmstadt University of Technology, Industrial Process and System Communications (KOM), Jul. 2000.
2. M. Arango, A. Dugan, I. Elliott, C. Huitema, and S. Pickett: Media Gateway Control Protocol (MGCP). Internet Draft, Internet Engineering Task Force, Feb. 1999. Work in progress.
3. I. Dalgic and H. Fang: Comparison of H.323 and SIP for IP Telephony Signaling. In Proc. of Photonics East, Boston, Massachusetts, Sept. 1999.
4. M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg: SIP: Session Initiation Protocol. Request for Comments (Proposed Standard) 2543, Internet Engineering Task Force, Mar. 1999.
5. K. Hoffmann, K. Pulverer, P. Blatherwick, B. Bell, M. Korpi, T. Taylor, R. Bach, and C. Ruppelt: Megaco/H.248 Generic Packages. Internet Draft, Internet Engineering Task Force, Sept. 1999. Work in progress.
6. International Telecommunication Union: Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service. Recommendation H.323, Telecom-

munication Standardization Sector of ITU, Geneva, Switzerland, May 1996.
7. International Telecommunication Union: Packet based multimedia communication systems. Recommendation H.323, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Feb. 1998.
8. ISO: Information Technology - Coding of audio-visual objects, Part 6: DMIF. ISO/IEC IS 14496-6, 1999. ISO IEC JTC1/SC29.
9. ISO: Information Technology - Coding of audio-visual objects, Part 6: DMIF version 2. ISO/IEC IS 14496-6v2, 2000. ISO IEC JTC1/SC29.
10. R. Koeman: MPEG-4, Multimedia for our time. IEEE Spectrum, 36(2), pages 26-33, February 1999.
11. Microsoft Corporation: The Microsoft Windows Telephony Platform with TAPI 2.1. White Paper. http://www.microsoft.com/ntserver/commserv/techdetails/prodarch/tapi21wp.asp.
12. T. Russell. Signaling system #7: McGraw-Hill, New York, 1995.
13. H. Schulzrinne and J. Rosenberg: A Comparison of SIP and H.323 for Internet Telephony. In The 8th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 98), pages 83-86, Cambridge, England, July 1998.
14. K. Singh and H. Schulzrinne: Interworking between SIP/SDP and H.323. Internet Draft, Internet Engineering Task Force, Jan. 2000. Work in progress.
15. Sun Microsystems: Java(tm) Telephony API. Product Description. http://java.sun.com/products/jtapi/.