AN IPTEL ARCHITECTURE BASED ON THE SIP PROTOCOL

João Paulo Sousa
Instituto Politécnico de Bragança
R. João Maria Sarmento Pimentel,
5370-326 Mirandela,
Portugal + 351 27 820 13 40
jpaulo@ipb.pt
Eurico Carrapatoso
FEUP/INESC Porto
R. Dr. Roberto Frias,
4200-465 Porto,
Portugal + 351 22 209 42 98
emc@fe.up.pt

ABSTRACT

More and better accesses to the Internet increase the interest in using it to carry not only data but also voice and video. The IP Telephony (IPTel) was born in this context and offers a framework to create multimedia communication systems. The *Session Initiation Protocol*, used in the IPTel architecture, is a protocol for signaling and call control between two or more participants. This paper presents telephony over IP service. Different protocols typically used in IPtel are analyzed and the architecture and functionality of SIP protocol are explained. Different mobility modes provided by SIP through the application layer are described. Finally we present the sIPtel, a Java application that supports real audio and video communications and uses the SIP protocol for call signaling.

KEYWORDS

IP Telephony, IPtel, SIP, Protocols, Signaling, Audio and Video Codecs.

1. INTRODUCTION

For many years companies and organizations have used a limited set of communication services, such as phone and fax, supported by the traditional telephone network. For Internet services they also have used the data switched network. Currently, they have the opportunity to use the IP network as a framework for communications, thus the integration of new services provides new communication forms and new business models are permitted, making the enterprises more efficient.

Currently two protocols exist for real time communications: the *Session Initiation Protocol* (SIP), from the *Internet Engineering Task Force* (IETF), and the H.323, from the *International Telecommunications Union* (ITU). These two protocols are used for call routing, signaling and call control, as well as other supplementary services.

Developed by the MMUSIC group of IETF, SIP was initially published in RFC 2543 in 1996. This RFC was made obsolete by RFC 3261 in June 2002 [1]. SIP is a control protocol that works at the application layer to create, modify and finalize sessions between one or more parties. These sessions, over IP, may include telephony calls, as well as multimedia distribution and conferences. The SIP or SIP extensions may also be used for instant messaging, presence and distributed games.

This paper starts with the IPtel architecture, followed by a brief presentation of the SIP protocol and the specification of an IPtel service. Three modes of mobility are described: terminal, session and personal motilities. Then sIPtel is presented, an IPtel service developed that provides real time communications with audio and video between two participants and uses the SIP protocol to establish sessions. This section also

describes the features of sIPtel and the architecture implemented. Finally some conclusions are drawn and some suggestions for future work presented.

2. REVIEW OF SIP PROTOCOL

SIP is a simple protocol that uses a set of protocols that provide different services and allow the implementation of a multimedia architecture. The *Real Time Transport Protocol* (RTP) [2] assures media transport and the *Real Time Control Protocol* (RTCP) [2] provides useful information for QoS management. The *Session Description Protocol* (SDP) [3] describes the multimedia sessions. The DNS discovers the address of the destination of a call. The *Lightweight Directory Access Protocol* (LDAP) accesses the database of a location server. The *Telephony Routing over IP protocol* (TRIP) [4] promotes the exchanges of routing information between administrative telephony domains. Finally the *Reservation Protocol* (RSVP) establishes resources reservation.

SIP is a text protocol that is based on a client/server model (the client makes requests and the server returns responses to the client's requests) and uses a semantic and syntax similar to HTTP, as well as its authentication methods. SIP may run over the *Transmission Control Protocol* (TCP) and the *Stream Control Transmission Protocol* (SCTP), but is more used over the *User Datagram Protocol* (UDP).

At the service level, in its initial recommendation SIP includes, for the establishment and finalization of calls, the following services: user location, user resources, session negotiation and session management.

2.1 Using SIP for signaling

It is during the signaling phase that the caller and the called party define parameters to establish the call and exchange data, such as transport address, media type and codecs.

2.1.1 SIP Components

To implement the functionalities that SIP provides, various components are needed: *User Agent*, *Proxy Server*, *Registrar Server* and *Redirect Server*. Some of them are illustrated in Figure 1.

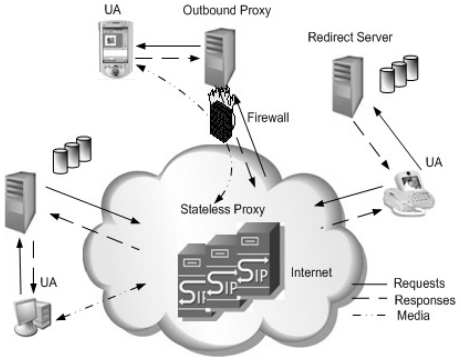

Figure 1. SIP Architecture

- *User Agent:* it is composed by two different parts, the *User Agent Client* (UAC) and the *User Agent Server* (UAS). The UAC is a logic entity that generates SIP requests and receives responses to those requests. The UAS is a logic entity that sends responses to SIP requests.
- *Proxy Server*: it is an intermediary entity that acts like client and server, with the purpose of establishing calls between users. There are two types of *Proxy Servers*: the *Stateful Proxy* and the *Stateless Proxy*:
  - *Stateful Proxy*: it keeps the transactions state during the processing of requests. It permits the forking of requests, in the attempt to find multiple locations. There is also an *Outbound Proxy*, which is *Stateful*, that receives requests from the users, even though it may not be the server resolved by the Request-URI. This configuration is more used and suitable when there are firewalls.

- *Stateless Proxy*: it does not keep the transactions state during the processing of requests. It is more used on SIP *backbones*.

- *Registrar:* it is a server that accepts the registration of users and saves the information of these requests to provide a location service and address translation in its own domain;
- *Redirect Server*: it is a UAS that generates 3xx responses to requests it receives, directing the client contact to an alternative set of URIs.

2.1.2 Starting and ending call

Figure 2 shows a call set up between two users through two *SIP Proxies*. *User1@ipb.pt* intends, with his IPtel software, to make a call to *user2@utad.pt*, which is registered in the *utad.pt* domain. The two *SIP Proxies* have the responsibility of controlling the domains and the registration of the users that belong to those domains.

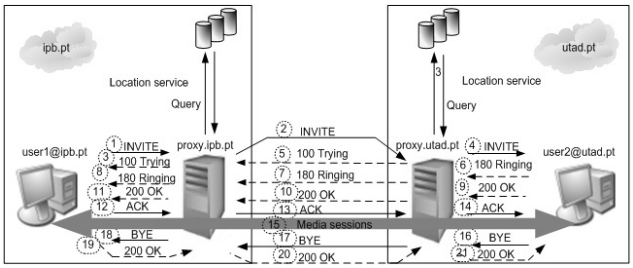

Figure 2. Call example between two users

To initiate the call, *user1* sends an INVITE to *user2*, that transports in the message body the session parameters that he intends to set up, using the SDP protocol. Because the IPtel software does not know the *user2* location or have in its configuration the proxy *ipb.pt* as *Outbound Proxy*, the INVITE is sent to the proxy *ipb.pt*, which forwards the request to the server closest to *user2*. The INVITE has a routing header with the proxy *ipb.pt* address to guarantee that all messages exchanged between *user1* and *user2* go through this proxy. The *proxy ipb.pt* receives the message and, by means of a location service, verifies that it does not know *user2*. Then, using a DNS query, it finds the *proxy utad.pt* that controls the *utad.pt* domain and forwards the INVITE to it. The *proxy utad.pt* receives the request and verifies if the *user2* is registered, and if that is the case the *proxy* forwards the INVITE to the user's IPtel software.

The application, when it receives the request, alerts the user that he has a new call and sends a response to *user1* through the servers. On the other side, *user1*, when it receives the message, alerts the user indicating that the call is ongoing. In this example *user2* accepts the call, so the application sends *user1* a final 200 OK response. The body response transports the parameters of the media session that he intends to use and completes the basic features negotiation provided by SIP. When *user1* receives the 200 OK response, it sends an ACK message and a *Dialog* is created, which is a point to point relationship between two users. After this the media exchange begins with the parameters accorded in the call set up through the SDP protocol. SIP does not control the routing of the media packets between the users and these packets can go through paths different from the one used by the SIP messages. To finish the call, *user2* sends a BYE to *user1*.

3. SIP MOBILITY

SIP allows different mobility mechanisms at the application level. Some of these mechanisms are provided by the protocol in a basic way, while others can be implemented using SIP extensions. The mechanisms presented here only concern signaling mobility and do not offer any support for mobility of the media exchange.

3.1 Terminal mobility

To support terminal mobility at the application level, it is necessary that the network supports mobility (IP Mobility) [5]. The terminal mobility can be divided in three parts: mobility before the call, mobility during the call, and error recovery [6].

- Mobility before the call: it requires that the mobile terminal receives a new address before making or accepting a call [7]. Each time that the terminal detects an IP address change, it creates a new register in the Registrar server and updates the discovered parameters.
- Mobility throughout a call: if a terminal moves during a session and detects a network address change, it must send a new INVITE to the remote user, announcing the session parameters change. At the same time, the process described in the previous section must be initiated.
- Error recovery: when, for any reason, an address break exists, it is convenient to have an automatic mechanism for error recovery. It is the case of two terminals that stay offline for moments and, when they are again online, they both have different IP addresses. A recovery solution is the two terminals registering with the new addresses on the Registrar server [7].

3.2 Session mobility

Session mobility allows session transferences between users and/or terminals. The transference process of sessions or parts of them involves features negotiation for the new session. SIP supports session mobility with two mechanisms. The first mechanism is called *third-party call control* [8] and allows, for example, call transference and conference signaling. The second mechanism is implemented through the REFER method [9], that allows the request for a call between a remote UA and a third entity.

3.3 Personal Mobility

SIP signaling is capable of routing the requests to set up calls to different locations, where the users decided that they could be contacted. The request routing can be made in parallel or sequentially.

In a parallel search (Figure 3a) the proxy hands out the requests to different addresses where it is possible that the user responds to this request. Each of these requests may create several responses which are returned through different proxies. SIP allows the definition of sets of rules for the treatment and return of those responses to the client. This is a native dynamic mechanism provided by SIP, and allows the user to choose where and what resources he wants to use to respond to the request.

In a sequential search (Figure 3b) the proxy server tries to contact each address in a sequential way, making the next attempt only after the previous has generated a final response. The search ends when a 2xx or 6xx class final response is generated [1].

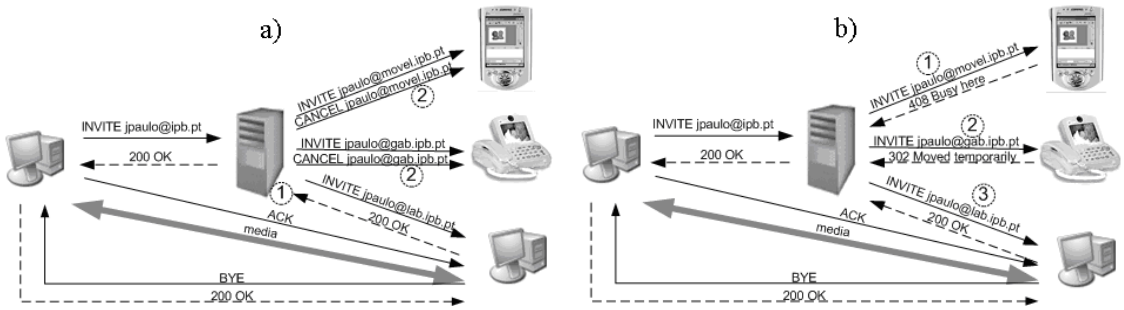

Figure 3. Searches example

3.4 sIPtel implementation

In the development of sIPtel (Figure 4), we have tried to provide the features more used in an IPtel service. The signaling features provided by sIPtel may be summarized as follows: support of one active call at any time, support of multiple calls on hold, possibility of specification by the user of the types of media (audio and/or video) that he wants to receive and/or send), possibility to register users with or without authentication and possibility to authenticate invited calls. sIPtel supports five audio codecs and two video codecs. These codecs and the mechanisms to handle the media were developed with the support of the *Java Media Framework* (JMF) API. To develop the signaling we used the NIST-SIP API of the *National Institute of Standards and Technology* (NIST) [10].

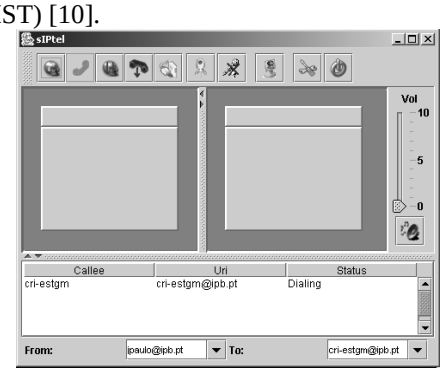

Figure 4. Graphical interface of sIPtel

3.5 sIPtel operation

sIPtel provides most of the features of the IPtel service, which are available through a graphical user interface illustrated in Figure 4.

This main interface is split in four sections: a control area, that includes a toolbar providing fast access to the functions; an area that presents the local video and the remote video; a visualization area, that presents the calls state and the calls in progress; and an area that contains two address bars to insert the SIP URI of caller and destination. The Figure 5 presents the functionalities provided by the sIPtel graphical user interface.

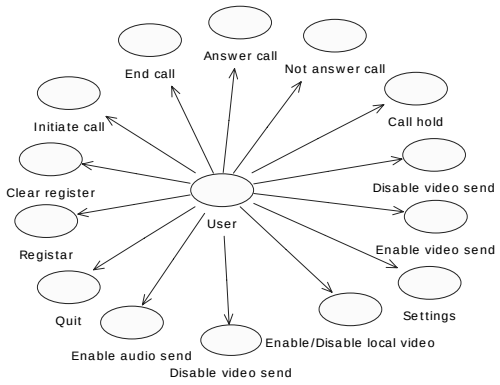

Figure 5. Operation of sIPtel

3.6 sIPtel architecture

The sIPtel implementation is divided in five parts, each corresponding to a package:

- *ipb.iptel.gui*: contains the classes that provide the user interface and the media presentation;

- *ipb.iptel.sip*: contains the classes that make possible call signaling;
- *ipb.iptel.rtp*: contains two classes responsible for sending and receiving the media;
- *ipb.iptel.util*: contains two utilities classes used by various classes that belong to the other packages;
- *ipb.iptel.test*: includes two applications that allow the user to test video and audio reproduction in the system.

SIP is a layered protocol. In the low level message parsing and coding are carried out. The second layer is the transport layer and defines how the client sends requests and receives responses and how a server receives requests and sends responses. The third layer is the transaction layer and deals with application-layer retransmission, matches the responses to the requests, and handles application-layer timeouts. The layer above the transaction layer is called the transaction user. Every SIP entity, except the stateless proxy, is a transaction user. Finally, there is the Dialog whose purpose is to make easier the messages sequencing and the proper routing of requests between UAs.

3.6.1 Transaction layer implementation

A SIP transaction occurs between a client and a server, and includes all the messages since the first request sent by the client to the server until he receives from the server one final response (non-1xx). All transactions have a client side and a server side. The client side is known as client transaction and the server side is known as server transaction. To implement the client transaction two state machines [1] were used:

- *Invite Client Transaction* (Figure 6): processing of the client transaction for INVITE requests;
- *Non Invite Client Transaction*: processing of the client transaction for all requests except INVITE and ACK.

Like for client transaction, there are two states machines for a server transaction [1]:

- *Invite Server Transaction* (Figure 6)*:* if the request is an INVITE;
- *Non Invite Server Transaction:* if the request is any except INVITE or ACK.

These two levels were implemented according to the RFC 3261 specification and were two of the more difficult tasks in the development of sIPtel.

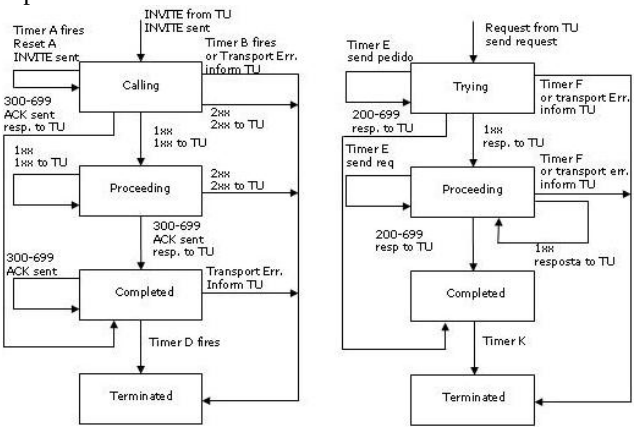

Figure 6. *Invite Client Transaction* and *Invite Server Transaction* state machines.

The following two sections describe two of the most important classes of IPtel. These classes are responsible for the graphical interface and signaling.

3.7 sIPtel class

The sIPtel class implements the Graphical User Interface (GUI). This class accepts in the command line the filename that contains the configuration needed to initiate the NIST-SIP stack. Through the GUI are also provided the functionalities presented in section 3.5, with the exception of stack parameters configuration.

Next the classes shown in Figure 7 are described. The sIPtel class results from the *JFrame* class and allows the creation of a window with title and border. The *VideoScrollPane* allows one or more *VideoPanels*, where the local and/or the remote video are presented. Two *VideoScrollPane* objects are created, one to support the *VideoPanel* object, where the local video is shown, and the other to support the various *VideoPanels* used to present the remote video streams. The *Player* interface is used to present and control the local video and remote audio. The *ServerMain* class initiates the *stack* and acts as the interface between the GUI and the classes that implement the signaling.

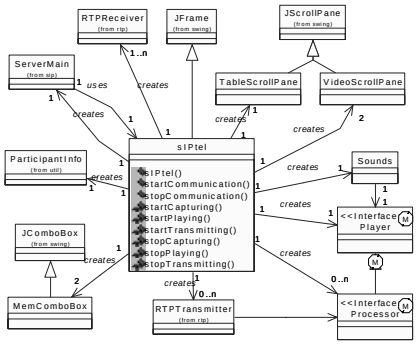

Figure 7. Main component: class diagram

3.8 ServerMain class

The *ServerMain* class, illustrated in Figure 8, implements the main reference point for all signaling classes. The NIST-SIP *stack* is created in this class and is configured with the parameters of the *configuration.properties* file. This *stack* uses the NIST-SIP parser that parses the messages received and calls the *SIPStackMessageFactoryImpl* class. Depending if the message is a request or a response, the *SIPStackMessageFactoryImpl* class creates a *SIPServerRequest* or *SIPServerResponse* and calls the *processRequest* method of the one of those objects. The creation and sending of SIP requests are carried out by the *HandlerRequests* class.

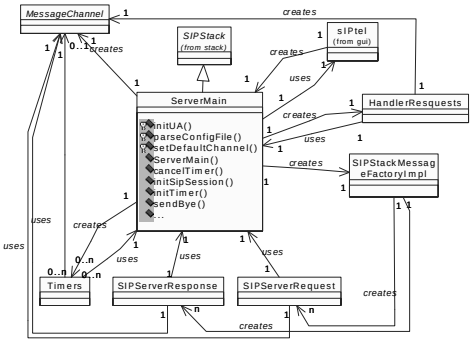

Figure 8. Signaling component: class diagram

The *ServerMain* class registers the states of all signaling transitions, needed to implement the four state machines that were described in section 3.6.1. The transaction level did not exist in the API NIST-SIP at the time of the implementation of this service, therefore it was implemented following the RFC 3261 and is responsible for the requests and responses retransmission, the association of the responses to the requests and timeouts. The transaction state is shared by various sIPtel classes: *ServerMain*, *HandlerRequests*, *SipServerResponse*, *SIPServerRequest* e *Timers*.

4. CONCLUSION AND FUTURE WORK

This paper described the IPtel architecture and an IPtel service with video support. The architecture is based on a set of independent and modular protocols that were combined to allow the implementation of the basic characteristics that exist on traditional telephony services and the integration of new services, using the IP network as the communications infrastructure.

The sIPtel was developed according to the IPtel architecture. It provides, through a graphical interface, a set of typical operations of an IPtel service, such as calls initiate, control and finalize. This service was developed using Java because of the native features that this language offers. We also used an open source API that simplified the construction of calls signaling and media exchange.

The sIPtel was subjected to two types of interoperability tests. The first one aimed to evaluate the interoperability level through the criterion defined by the *Technical Program Committee*. The sIPtel satisfied 100% of the interoperability basic criteria. The second type consisted in the analysis of interoperability of the sIPtel with various IPtel software packages (e.g., *SCS-Client, Instant xpressa, eStara SoftPhone, Ubiquity's User Agent, NIST-SIP Proxy, Proxy Siemens*), that use the SIP protocol for call signaling. In this test we verified that sIPtel communicated with success in many scenarios. We also detected some incompatibilities with session announcement. These incompatibilities happened because the proprietary software adopted different documents that propose different scenarios for the same operation (e.g., put call on hold).

The choice of the SIP protocol has proved, in our opinion, to be a good choice because in the middle of this work only a few experimental applications were using this protocol, but since then the number has increased significantly. In the future we intend to integrate new services in the sIPtel, such as instant messages, presence, QoS, and also implement a full architecture to support micro and macro mobility.

ACKNOWLEDGMENTS

We would like to thank Mr. José Manuel Oliveira for the enlightening discussions that took place during the work presented in this article.

REFERENCES

[1] J. Rosenberg et al, " SIP: Session Initiation Protocol ", Request For Comments 3261, Internet Engineering Task Force, June 2002 (RFC 2543 obsolete). URL: http://www.ietf.org, July 2002.

[2] H. Schulzrinne et al, "RTP: a transport protocol for real-time applications", Request For Comments 1889, Internet Engineering Task Force, Janeiro 1996. URL: http://www.ietf.org, June 2002.

[3] M. Handley and V. Jacobson, "SDP: Session Description Protocol", Request For Comments (RFC 2327), Internet Engineering Task Force, April 1998. URL: http://www.ietf.org, July 2002.

[4] J. Rosenberg et al, “Telephony Routing over IP (TRIP)”, Request For Comments 3219, Internet Engineering Task Force, January 2002. URL: http://www.ietf.org, July 2002.

[5] C. Perkins, “IP mobility support”, Request for Comments 2002, Internet Engineering Task Force, October 1996.

[6] H. Schulzrinne, E. Wedlund, “Application-layer mobility using SIP”, http://citeseer.nj.nec.com, September 2002.

[7] E. Wedlund, H. Schulzrinne, “Mobility Support using SIP”, *in Second ACM/IEEE International Conference on Wireless and Mobile Multimedia*, Seatle, August 1999. URL: http://citeseer.nj.nec.com, September 2002.

[8] J. Rosenberg et al, “Best Current Practices for Third Party Call Control in the Session Initiation Protocol,” Internet Draft, Internet Engineering Task Force, March 2003. Work in progress.

[9] R. Sparks, A. Johnston, “Session Initiation Protocol Call Control - Transfer”, Internet Draft, Internet Engineering Task Force, February 2003. Work in progress.

[10] M. Ranganathan, “Internet Telephony/VOIP”, January 2001, URL: http://dns.antd.nist.gov/proj/iptel/, November 2002.