IP Telephony Cookbook

Deliverable 2

Saverio Niccolini, University of Pisa
Dr. Rosario Giuseppe Garroppo, University of Pisa
Dr. Jörg Ott, Universität Bremen TZI
Stefan Prelle, Universität Bremen TZI
Jiri Kuthan, FhG Fokus
Jan Janak, FhG Fokus
Dr. Sven Ubik, CESNET
Dr. Margit Brandl, Karl-Franzens-Uni Graz
Dimitris Daskopoulos, GRNET
Egon Verharen, SURFnet
Erik Dobbelsteijn, SURFnet

IP Telephony Cookbook: Deliverable 2

by Saverio Niccolini, Dr. Rosario Giuseppe Garroppo, Dr. Jörg Ott, Stefan Prelle, Jiri Kuthan, Jan Janak, Dr. Sven Ubik, Dr. Margit Brandl, Dimitris Daskopoulos, Egon Verharen, and Erik Dobbelsteijn

Table of Contents

List of Figures

List of Examples

Chapter 1. Introduction

1.1. Goal

The IP telephony Cookbook is a reference document addressing technical issues for the set-up of IP telephony solutions. Its goal is to provide the TERENA user community with guidelines and information about the IP telephony world and everything related with it. Since the Cookbook is intended to be a technical document, the main target audience addressed here are the network engineers and system administrations at universities and National REsearch Networks (NRENs); however, university students and researchers may found it useful both for having a technology background and for finding in it information about advanced research topics and projects in the European community.

1.2. Reasons for writing this document

TERENA was requested by some members to start an investigation into IP telephony in September 2001. The response was very positive and suggestions were made to coordinate the creation of a cookbook with recommendations for setting up IP telephony solutions at university and national level, providing information about protocols and interoperability of equipment, as well as integration with the existing international hierarchies for IP videoconferencing. For this reason, a number of people in the TERENA community with significant expertise in the area of IP telephony decided to undertake this task and to write this document hereinafter referred to as IP telephony Cookbook.

1.3. Contents

The IP telephony Cookbook is divided in chapters which guide the reader through an increasing knowledge of the IP telephony. This first chapter contains introductory information and details the contents of Cookbook, moreover useful tips on how to read this document and techno-economic considerations are provided. The Chapter 2 explains the technology background needed in order to understand the topics addressed in the all Cookbook; in this chapter the basic IP telephony components are described and an overview about the IP telephony protocols is given. Chapter 2 ends with additional considerations on call routing and perspectives about the future. Chapter 3 gives an high level overview of scenarios a user may face when building an IP telephony environment, details are given in order to explain what a particular scenario is, what is needed in order to deploy it and why users should go for it. The next three chapters (Chapter 4, Chapter 5 and Chapter 6) detail how to set up IP telephony services; those chapters give the reader the chance of learn how to set up basic services, advanced services (even if still telephony centric) and value added services (where value added is intended with respect to the classic telephony service). Chapter 7 is about Global telephony integration, this chapter details the technology solutions in order to have a global IP telephony integration and a successful replacement of the classic telephony, moreover reports on the today's situation as well as migration and future trends are given. The last chapter contains regulatory/legal considerations users have to be aware when moving from classic telephony to IP telephony, the topics here described are relative to the regulation of IP telephony in Europe and in other countries out of the EU. A large number of classic telephony legal issues will be detailed (from Licensing to Unbundling) mapping them to the IP world. Moreover, the IP telephony Cookbook contains two annex: Annex A and Annex B. While the Annex A is focused to list and describe (briefly) the current and future IP telephony Projects in Europe to learn from, the Annex B gives the reader useful information about IP telephony hardware and software reporting "hands on" experience (i.e., reports on how the devices either did or did not work, how was tech-support, what were workarounds for the problems, etc).

1.4. How to read this document

Since the IP telephony Cookbook is a technical reference document it must include guidelines for users who do not want to read the whole document in order to find the information they need. In this section we give the reader tips on how to read the document in order to retrieve the information needed as fast as possible; for a detailed overview of the contents of the Cookbook please refer to the previous section. To speed up the information retrieval process each readers of this document should identify himself in one of the three main groups:

- users who have no knowledge of IP telephony

- users who do have basic knowledge of IP telephony
- user who have an advanced knowledge of IP telephony

The users belonging to the first group should, first of all, refer to Chapter 2 to acquire the necessary background to understand the rest of the Cookbook, moreover those kind of users who want to set up an IP telephony service should read the Chapter 3 to have a clear picture of the possible scenarios offered by the IP telephony and which is best suited to their environment needs. The second group of users should skip the previous mentioned chapters even if Chapter 3 may be of some interest to them; the main focus of this group of users is more likely to be in the Chapter 4 and Chapter 5 in order to have tips and help in setting up an operative service. The third group of users is likely to be more interested in the "value added" services available nowadays with IP telephony (Chapter 6) or in the integration problem of a IP telephony architecture widely distributed across multiple site and organizations (Chapter 7). Moreover, all three groups of users may find useful regulatory information inserted in Chapter 8 and european project information in Annex A. Last but not least, the product list testing experience reported in Annex A is a must for all users who do not want to risk to make wrong choices in buying decision.

1.5. Techno-economic aspect of moving from classic telephony to VoIP

Many research institutions are facing investment decisions with respect to replacing or expanding their existing telephony infrastructure, which currently mainly consists of large PBXes with proprietary phones and interfaces. As there is a clear industry trend of replacing old-style (TDM) PBXes by IP telephony, it is important that there is a guide on how to attach such an IP telephony solution to the research network. The existing IP connectivity can be used as the basis for establishing good communication between scientists that might not use traditional, still relatively expensive long distance calls as extensively as they could use IP telephony. In addition, the potential for enhancing IP telephony by additional services that support scientific cooperation makes IP telephony an attractive solution even where financial constraints are not as tight.

IP telephony can provide a number of benefits beyond replacing existing PBX/PSTN telephony:

- *Enhanced speech quality:* - The PSTN (and most PBXes) are limited to 3.1 kHz, 8-bit/sample audio. It is likely that future IP phones can provide CD quality, possibly even stereo audio. Even where the additional bandwidth required for this level of quality cannot be provided, codecs such as G.722 (7 kHz speech bandwidth) can be used.
- *Improved availability:* - There are many aspects of availability: Lowering the cost can make telephony more available to low-budget activities. Redundancy can provide as good as (or even better) reliability as traditional telephony. Integrating telephony with location-based computing and group awareness systems can make the communication partners much more "available" - or provide means to move communication to a point in time where it is more appropriate than the usual interrupt-driven telephone call.
- *Improved coverage:* - In a similar argument, IP telephony can be made available in places where traditional phones are often not available in a university, e.g., lab settings (in particular, student labs). Also, many universities still disregard their employees sufficiently to have them share phones in a common office - again, not necessary when workstation-based IP telephony is used.
- *Improved media integration* - IP phones can be enabled to add media to an ongoing call as required, e.g., viewing a picture or drawing on a whiteboard. Using workstations themselves as IP phones can facilitate providing this function while the standards are not yet there for coupling phones and workstations.
- *New services* - As IP telephony evolves it can be used to provide new services (like user defined call processing) or integrate existing concepts, e.g. Presence, Location Awareness or Instant Messaging. Because of the open standards this services must not be limited to a specific vendor.
- *Research* - As mentioned before the protocols and standards used for IP telephony are open and public available. This allows research institutions to work on their own services and solutions.

As regards as the economic aspects, the packetization of voice using Voice over IP has given rise to new interna-

tional telecommunications carriers. These carriers have distributed network architectures using the Internet as a platform. VoIP networks have an architecture offering the most efficient way to implement multilateral telecommunications agreements, thus eliminating the need for carriers to engage in hundreds of bilateral traffic agreements as is required between traditional circuit switched PSTN carriers. Moreover, since packet networks are software driven, they can be configured more dynamically than traditional PSTN can. For example, with a global voice over packet network, new destinations are available to all users on the network, without the need for additional investment every day.

IP telephony telecommunications companies may expand the availability of services to a wider audience. IP telephony technologies can be used to build out voice networks more rapidly and at a lower cost than legacy PSTN systems. Easier deployment of Voice over IP networks can bring the benefits of telecommunications to more people in a much shorter time frame than would be possible with conventional PSTN networks. At the same time, not having to build extensive infrastructure gives many companies a motivation in migrating to IP telephony architectures.

Chapter 2. Technology Background

Abstract

This chapter shall provide technical background information about protocols and components used in IP telephony. It introduces involved component types, gives detailed information about H.323 and SIP and RTP as well as some notices about *Media Gateway Control* and vendor specific protocols.

2.1. Components

An IP telephony infrastructure usually consists of different types of components. This section shall give an overview about typical components without describing them in a protocol specific context.

2.1.1. Terminal

A terminal is an communication endpoint that terminates calls and their media streams. Most commonly this is a hard- or software telephone or videophone, eventually enhanced with data capabilities.

There are terminals that are intended for user interaction and those which are automated - like answering machines.

An IP telephony terminal is located on at least one IP address. There may well be multiple terminals on the same IP address but they are treated independently.

Most of the time a terminal has assigned one or more addresses (see Section 2.1.5). In case that IP telephony servers are used a terminal registers this addresses with its server.

2.1.2. Server

To place an IP telephony call it requires at least two terminals - and the knowledge of the IP address and port number of the terminal to call. Obviously such a situation is not very satisfying for you don't want to remember IP addresses to call persons and you want to be able to call targets on hosts that use DHCP.

As mentioned before terminals usually register their addresses with a server component. The server stores this telephone addresses along with the IP addresses and thus is enabled to map a telephone address to a host.

When a telephone user dials an address the server tries to resolve the given address into a network address. To do so the server may interact with other telephony servers or services. It may also provide further call routing mechanisms like CPL scripts or skill-based routing.

Finally a telephony server is responsible to authenticate registrations, authorize caller and to do the accounting

2.1.3. Gateway

Gateways are telephony endpoints that allow calls between endpoints that usually wouldn't inter operate. Usually this means that a gateway translates one signaling protocol into another (e.g. SIP/ISDN Signaling gateways), but translating between different network addresses (IPv4/IPv6) or codecs (Media Gateways) can be called gatewaying as well. It is of course possible that all functionality exists in a single gateway.

Finding gateways between VoIP and a traditional PBX is usually quite simple. Gateways that translate different VoIP protocols are harder to find. Most of them are limited to basic call functionality.

2.1.4. Conference Bridge

Conference Bridges provide means of having 3- or multi point conferences that can be either ad-hoc or scheduled. Because of the high resource requirements conference bridges are usually dedicated servers with special media hardware.

2.1.5. Addressing

A user willing to use a communication service needs an identifier to describe itself and the called party. Ideally, such an identifier should be independent of the user physical location. The network should be then responsible for finding the current location of the callee. One party may define to be reached by multiple contacts.

Regular telephony systems use E.164 numbers[1] - the international public telecommunication numbering plan. An identifier is composed of up to 15 digits with a leading plus sign, for example +1234565789123. When dialing, the leading plus is normally replaced by the international access code, usually double zero (00). This is followed by a country code and a subscriber number.

First IP telephony systems used IP addresses of end-point devices as user identifiers. Sometimes they are still used now. However, IP addresses are not location independent (even if we use IPv6) and hard to remember (especially if we use IPv6) and therefore they are not suitable for user identifiers.

Current IP telephony systems use two kinds of identifiers:

- *URIs* (RFC2396[2])
- *Numbers* (E.164)

Some systems tried to use names (alpha-numeric strings), but it led to a flat naming space and thus limited zones of applicability.

A Universal Resource Identifier (URI) uses a registered naming space to describe a resource in a location independent way. Resources are available under a variety of naming schemes and access methods including e-mail addresses (mailto), SIP identifiers (sip), H.323 identifiers (h323,RFC3508[3]) or telephone numbers (draft-ietf-iptel-rfc2806bis-02[4]). E-mail like identifiers have several advantages. They are easy to remember, nearly every Internet user already has an e-mail address and a new service can be added using the same identifier. The user location can be find with a Domain Name System (DNS). The disadvantage of URIs is that they are difficult or impossible to dial on some user devices (phones).

If we want to integrate a regular telephony system with IP telephony, we must deal with phone number identifiers even on the IP telephony side. The numbers are not well suitable for the Internet world relying on domain names. Therefore, the ENUM system was invented, using adapted phone numbers as domain names. We will describe ENUM in Chapter 7.

2.2. Protocols

2.2.1. H.323

The H.323 Series of Recommendations evolved out of the ITU-T's work on video telephony and multimedia conferencing: after completing standardization on video telephony and video conferencing for ISDN at up to 2 Mbit/s in the H.320 series, the ITU-T took on work on similar multimedia communication over ATM networks (H.310, H.321), over the analog Public Switched Telephone Network (PSTN) using modem technology (H.324), and over the still-born Isochronous Ethernet (H.322). The most widely adopted and hence most promising network infrastructure - and the one bearing the largest difficulties to achieve well-defined Quality of Service - was addressed in the beginning of 1995 in H.323: Local Area Networks, with the focus on IP as network layer protocol. The primary goal was to interface multimedia communication equipment on LANs to the reasonably well-established base on circuit-switched networks.

The initial version of H.323 was approved by the ITU-T about one year later in June 1996, thereby providing a basis on which the industry could converge. The initial focus was clearly on local network environments, as QoS mechanisms for IP-based wide area networks such as the Internet were not well established at this point. In early 1996 Internet-wide deployment of H.323 was already explicitly included in the scope as was the aim to support voice-only applications and, thus, the foundations to use H.323 for IP Telephony were laid. H.323 has continuously evolved towards becoming a technically sound and functionally rich protocol platform for IP telephony ap-

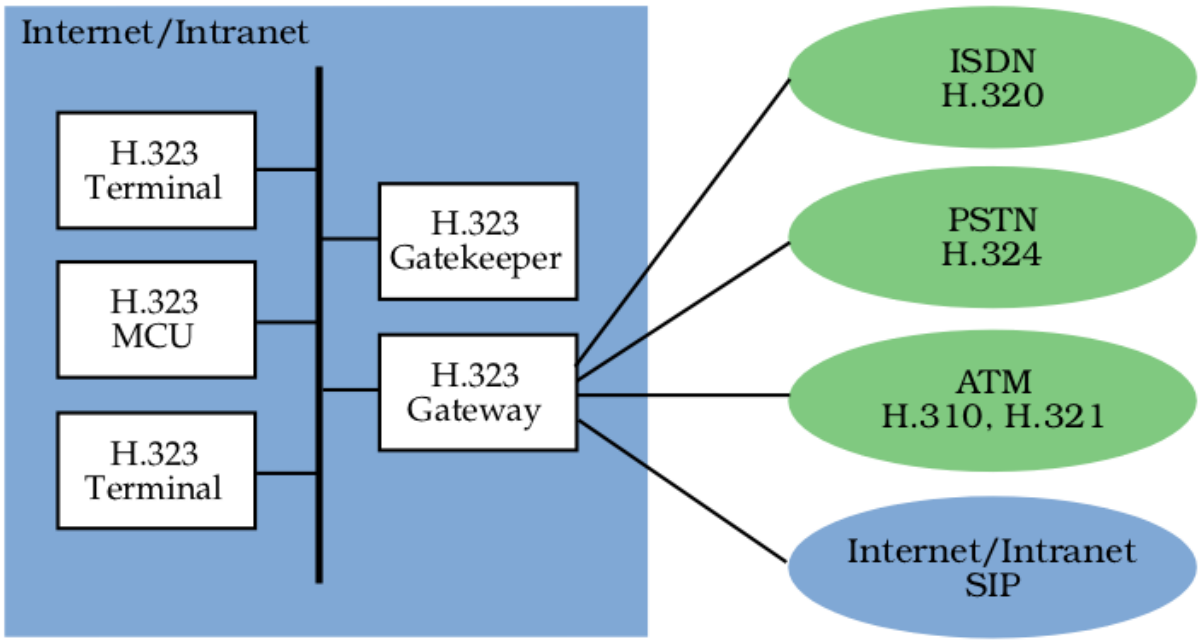
[1] http://www.numberingplans.com
[2] http://www.ietf.org/rfc/rfc2396.txt
[3] http://www.ietf.org/rfc/rfc3508.txt
[4] http://www.ietf.org/internet-drafts/draft-ietf-iptel-rfc2806bis-02.txt

plications, the first major additions to this end being included in H.323 version 2 approved by the ITU-T in January 1998. In September 1999, H.323 version 3 was approved by the ITU-T, incorporating numerous further functional and conceptual extensions to enable H.323 to serve as a basis for IP telephony on a global scale and to make it meet requirements in enterprise environments as well.

2.2.1.1. Scope

As stated before, the scope of H.323 encompasses multimedia communication in IP-based networks, with significant consideration given to gatewaying to circuit-switched networks (particular to ISDN-based video telephony and to PSTN/ISDN/GSM for voice communication).

Figure 2.1. Scope and Components defined in H.323



H.323 defines a number of functional / logical components as shown in figure Figure 2.1:

- *Terminal* -- Terminals are H.323-capable endpoints, which may be implemented in software on workstations or as stand-alone devices (such as telephones). They are assigned to one or more aliases (e.g. a user's name / URI) and/or telephone number(s).
- *Gateway* -- Gateways interconnect H.323 entities (such as endpoints, MCUs, or other gateways) to other network/protocol environments (such as the telephone network). They are also assigned one or more aliases and/or telephone number(s). The H.323 series of Recommendations provides detailed specifications for interfacing H.323 to H.320, ISDN/PSTN, and ATM based networks. Recent work also addresses control and media gateway specifications for telephony trunking networks such as SS7/ISUP.
- *Gatekeeper* -- The gatekeeper is the core management entity in an H.323 environment. It is, among other things, responsible for access control, address resolution, and H.323 network (load) management and provides the central hook to implement any kind of utilization / access policies. An H.323 environment is subdivided into zones (which may but need not be congruent with the underlying network topology); each zone is controlled by one primary gatekeeper (with optional backup gatekeepers). Gatekeepers may also provide value-add, e.g. act as conferencing bridge or offer supplementary call services.
- *Multipoint Controller (MC)* -- A multipoint controller is a logical entity that interconnects call signaling and conference control channels of two or more H.323 entities in a star topology. MCs coordinate the (control aspects of) media exchange between all entities involved in a conference; they also provide the endpoints with participant lists, exercise floor control, etc. MCs may be embedded in any H.323 entity (terminals, gateways gatekeepers) or implemented as stand-alone entities. They can be cascaded to allow conferences spanning multiple MCs.
- *Multipoint Processor (MP)* -- For multipoint conferences with H.323, an optional Multipoint Processor may be used that receives media streams from the individual endpoints, combines them through some mixing/

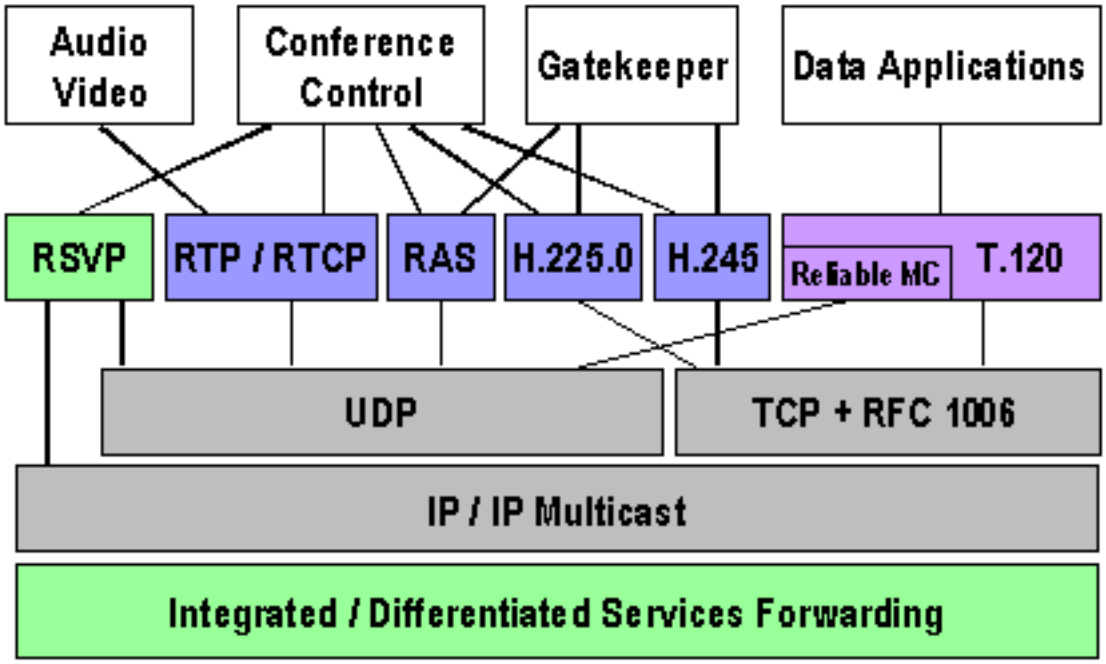
switching technique, and transmits the resulting media streams back to the endpoints.

- *Multipoint Control Unit (MCU)* -- In the H.323 world, an MCU simply is a combination of an MC and an MP in a single device. The term originates in the ISDN videoconferencing world where MCUs were needed to create multipoint conferences out of a set of point-to-point connections.

2.2.1.2. Signaling protocols

H.323 resides - similar to the IETF protocols discussed in the next subsection - on top of the basic Internet Protocols (IP, IP Multicast, TCP, UDP) and makes use of integrated and differentiated services along with resource reservation protocols.

Figure 2.2. H.323 protocol architecture



For basic call signaling and conference control interactions with H.323, the aforementioned components communicate using three control protocols:

- *H.225.0 Registration, Admission, and Status (RAS)* -- The RAS channel is used for communication between H.323 endpoints and their gatekeeper and for some inter-gatekeeper communication. Endpoints use RAS to register with their gatekeeper, to request permission to utilize system resources, to have addresses of remote endpoints resolved, etc. Gatekeepers use RAS to keep track of the status of their associated endpoints and to collect information about actual resource utilization after call termination. RAS provides mechanisms for user / endpoint authentication and call authorization.

- *H.225.0 Call Signaling* -- The call signaling channel is used to signal call setup intention, success, failures, etc. as well as to carry operations for supplementary services (see below). Call signaling messages are derived from Q.931 (ISDN call signaling), as is the protocol; however, simplified procedures and only a subset of the messages are used in H.323. The call signaling channel is used end-to-end between caller and callee and may optionally run through one or more gatekeepers (the call signaling models are later described in the Signaling models section).

  Optimizations: Since version 3 H.225.0 supports the following enhancements:

  - *Multiple Calls* - To prevent using a dedicated TCP connection for each call gateways can be built to handle multiple calls on each connection.

- *Maintain Connection* - Similar to Multiple Calls this enhancement shall reduce the need to open new TCP connections. After the last call has ended the endpoint may decide to maintain the TCP connection to provide a better call setup time for the next call.

Primary use of both enhancements is at the communication between servers (Gatekeeper, MCU) or gateways. While in theory both mechanisms were possible before, beginning with H.323v3 the messages contained fields to indicate support for the mechanisms.

- *H.245 Conference Control* -- The conference control channel is used to establish and control two party calls (as well as multiparty conferences). Its functionality includes determining possible modes for media exchange (e.g. select media encoding formats both parties understand) and configuring actual media streams (including exchanging transport addresses to send media streams to and receive them from). H.245 can be used to carry user input (such as DTMF), it also enables confidential media exchange, defines syntax and semantics for multipoint conference operation (see below). Finally, it provides a number of maintenance messages. Also this logical channel may optionally run through one or more gatekeeper or directly between caller and callee (please refer to the Signaling models section for details).

  It should be noted that H.245 is a legacy protocol inherited from the collective work on multimedia conferencing over ATM, PSTN, and other networks. Hence it carries a lot of fields and procedures that do not apply to H.323 but make the protocol specification quite heavyweight.

  Optimizations: The conference control channel is also subject to optimizations. Per default it is transported over an exclusive TCP connection but it may also be tunneled within the signaling connection (H.245 tunneling). Other optimizations deal with the call setup time. The last chance to start a H.245 channel is on receipt of the CONNECT message which implies that the first seconds after the user accepted the call no media is transmitted. H.245 may also start parallel to the setup of the H.225 call signaling, which is not really a new feature but another way of dealing with H.245. Vendors often call this *Early connect* or *Early media*. Since H.323 V2 it is possible to start a call using a less powerful but sufficient capability exchange by simply offering possible media channels that just have to be accepted. This procedure is called *FastConnect* or *FastStart*, requires less round-trips and is transported over the H.225 channel. After the FastConnect procedure is finished or when it fails the normal H.245 procedures start.

A number of extensions to H.323 include mechanisms for more efficient call setup (H.323 Annex E) and reduction of protocol overhead e.g. for simple telephones (SETs, simple endpoint types, H.323 Annex F).

2.2.1.3. Gatekeeper Discovery and Registration

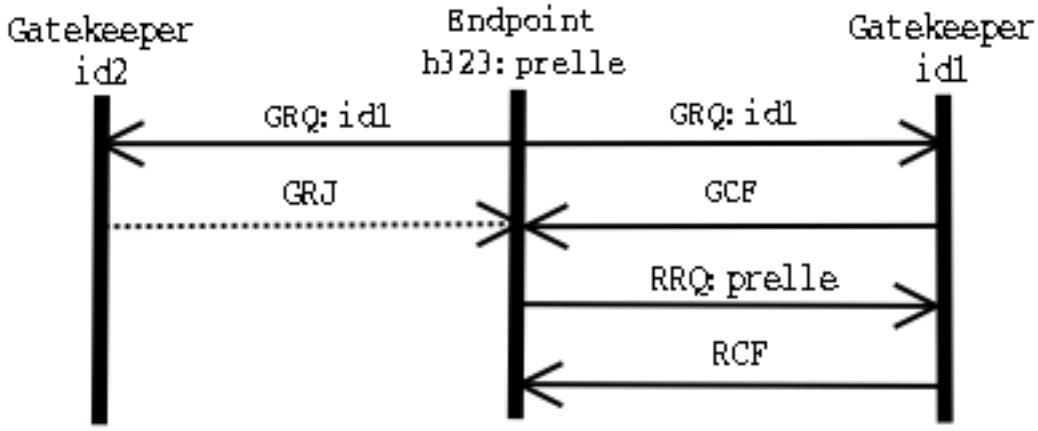
A H.323 endpoint usually registers with a gatekeeper that provides services like address resolution to end endpoint. There are two possibilities for an endpoint to find its gatekeeper:

- *Multicast discovery* - The endpoint sends a gatekeeper request (GRQ) to a well known multicast address (224.0.1.41) and port (1718). Receiving gatekeepers may confirm their responsibility for the endpoint (GCF) or ignore the request.

- *Configuration* - The endpoint knows the IP address of the gatekeeper by its configuration. While their is no need that a gatekeeper request (GRQ) must be sent to the preconfigured gatekeeper some products need this protocol step. If a gatekeeper receives a GRQ via unicast it must either confirm the request or reject it (GRJ).

When trying to discover the gatekeeper via multicast an endpoint may request gatekeeper or specify the request by adding a Gatekeeper identifier to the request. Only gatekeeper that have the requested identifier reply positively. (see figure Figure 2.3)

Figure 2.3. Discovery and registration process



After the endpoint knows the location of the gatekeeper it tries to register itself (RRQ). Such a registration includes (among other information):

- *The addresses of the endpoint* - For a terminal this may be the user ids or telephone numbers. An endpoint may have more than one address. In theory it is possible that addresses belong to different users to enable multiple users to share a single phone - in practice this depends on the phones and gatekeeper implementation.
- *Prefixes* - If the registering endpoint is a gateway it may register number prefixes instead of addresses.
- *Time to live* - An endpoint may request how long the registration shall last. This value can be overwritten by gatekeeper policies.

The gatekeeper checks the registration information and confirms the (eventually modified) values (RCF). It may also reject such a registration because of e.g. invalid addresses. In case of a confirmation the gatekeeper assigns a unique identifier to the endpoint that shall be used in subsequent requests to indicate that the endpoint is already registered.

2.2.1.3.1. Addresses and registrations

H.323 distinguishes several address types. Most commonly used an derived from the PSTN world is the *Dialed digit* that defines a number as dialed by the endpoint. It doesn't include further information (e.g. about the dial plan) and needs to be interpreted by the server. The server might convert the dialed number into an *Party Number* that includes information about the type of number and the dialplan.

To provide name dialing H.323 supports *H.323-IDs* that represent names or e-mail like addresses or the more general approach of an *URL-ID* which represents any kind of URL.

Unlike SIP in H.323 an address can be only registered by one endpoint (per zone) so a call to that address only resolves to a single endpoint. To call multiple destinations simultaneously requires a gatekeeper that actively maps a single address to multiple different addresses and tries to contact them.

2.2.1.3.2. Updating registrations

A registration expires after a defined time and must therefore be refreshed. This can be done by simply sending another registration request including the assigned endpoint identifier. To reduce the registration overhead in regularly registrations H.323 supports KeepAlive registrations that contain just the previously assigned endpoint identifier. Of course these registrations may only be sent if the registration information (exp. addresses) are unchanged.

Especially for registration endpoints with a huge amount of addresses to register (which would exceed the size of a UDP packet) H.323 version4 supports *Additive Registration*, a mechanism that allows an endpoint to send multiple registration requests (RRQ) in which the addresses don't replace existing registrations but are added to them.

2.2.1.4. Signaling models

The call signaling messages and the H.245 control messages may be exchanged either end-to-end between caller and callee or through a gatekeeper. Depending on the role the gatekeeper plays in the call signaling and in the H.245 signaling the H.323 specification foresees three different types of signaling models:

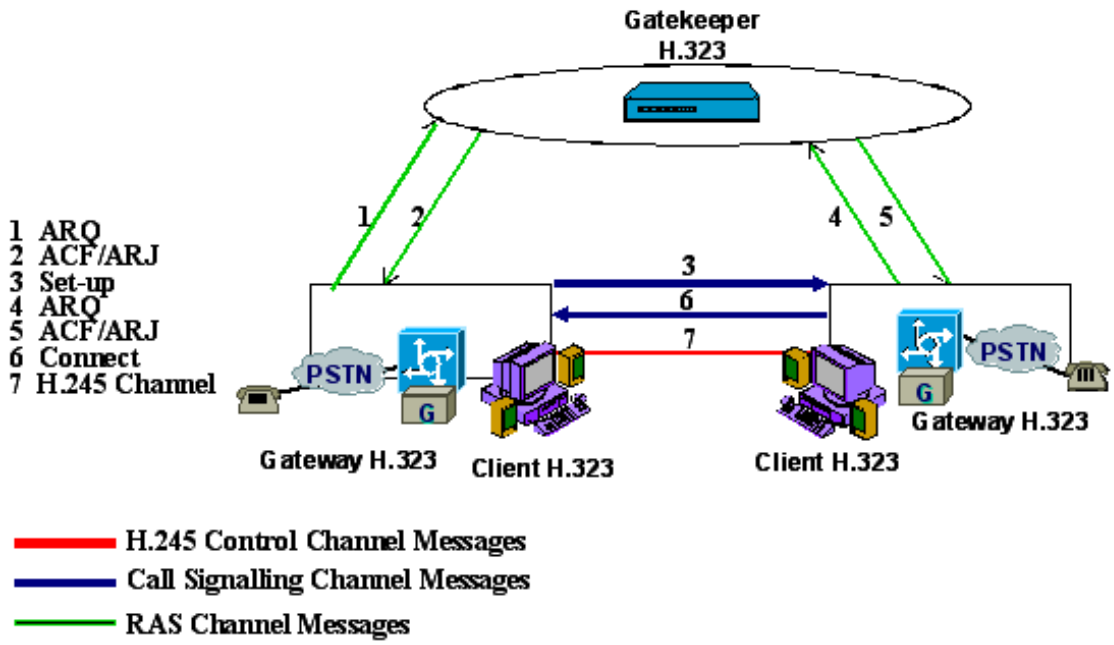
- Direct signaling, with this signaling model only H.225.0 RAS messages are routed through the Gatekeeper while the other logical channel messages are directly exchanged between the two endpoints;
- Gatekeeper routed call signaling, with this signaling model H.225.0 RAS and H.225.0 Call signaling messages are routed through the Gatekeeper while the H.245 Conference control messages are directly exchanged between the two endpoints;
- Gatekeeper routed H.245 control, H.225.0 RAS and H.225.0 Call signaling an H.245 Conference control messages are routed through the Gatekeeper and only the media streams are directly exchanged between the two endpoints.

In the following sub-sections we are going to detail each signaling model. The figures reported in this section apply both to the use of a single Gatekeeper and to the use of a "Gatekeeper network". Since the signaling model is decided by the endpoint's Gatekeeper configuration and apply to all the messages such Gatekeeper handles, the extensions to the multiple Gatekeeper case is straightforward (simply apply the definition of the signaling model described in the itemized list above to each Gatekeeper involved) except for the location of zone external targets (described later in Locating zone external targets section); we decided not to report those message exchange in any of this section figures as it is intended to remain bounded in the ellipse where the H.323 Gatekeeper is depicted and it is described in the Locating zone external targets section. Please note that there is no indication about the call termination in each signaling model sub-section, please refer to Communication phases section for details.

The *Direct signaling model* is depicted in Figure 2.4. In this model the H.225.0 Call signaling and H.245 Conference control messages are exchanged directly between the call termination. As shown in the figure, the communication starts with an ARQ (Admission ReQuest) message sent by the caller (which may be either a Terminal or a Gateway) to the Gatekeeper. The ARQ message is used by the endpoint to be allowed to access the packet-based network by the Gatekeeper, which either grants the request with an ACF (Admission ConFirm) or denies it with an ARJ (Admission ReJect), if an ARJ is issued the call is terminated. After this first step the Call signaling part of the call begins with the transmission of the SET UP message from the caller to the callee. The transport address of the SET UP message (and of all the H.225.0 Call signaling messages) is retrieved by the caller from the "destCallSignalAddress" field carried inside the ACF received, in the case of Direct signaling model it is the address of the destination endpoint. Upon receiving the SET UP message the callee starts its H.225.0 RAS procedure with the Gatekeeper, if successful a CONNECT message is sent back to the caller to indicate acceptance of the call. Before sending the CONNECT message, two other messages may be sent from the callee to the caller (those two messages are not depicted in the figure since we have reported only mandatory messages):

- ALERTING message, this message may be sent by the called user to indicate that called user alerting has been initiated (in everyday terms, the "phone is ringing");
- CALL PROCEEDING message, this message may be sent by the called user to indicate that requested call establishment has been initiated and no more call establishment information will be accepted.

Figure 2.4. Direct signaling model



The CONNECT message closes the H.225.0 Call signaling part of the call and make the Terminals starting the H.245 Conference control one. In such call model the H.245 Conference control messages are exchanged directly between the two endpoints (the correct "h245Address" was retrieved from the CONNECT message itself). The procedures started with the H.245 Conference control channel are used to:

- allow the exchange of audiovisual and data capabilities, with the TERMINAL CAPABILITY messages;
- request the transmission of a particular audiovisual and data mode, with the LOGICAL CHANNEL SIGNALING messages;
- to manage the logical channels used to transport the audiovisual and data information;
- to establish which terminal is the master terminal and which is the slave terminal for the purposes of managing logical channels, with the MASTER SLAVE DETERMINATION messages;
- to carry various control and indication signals;
- to control the bit rate of individual logical channels and the whole multiplex, with the MULTIPLEX TABLE SIGNALING messages;
- to measure the round trip delay, from one terminal to the other and back, with the ROUND TRIP DELAY messages.

Once the H.245 Conference control messages are exchanged the two endpoints have all the necessary information to open the media streams.

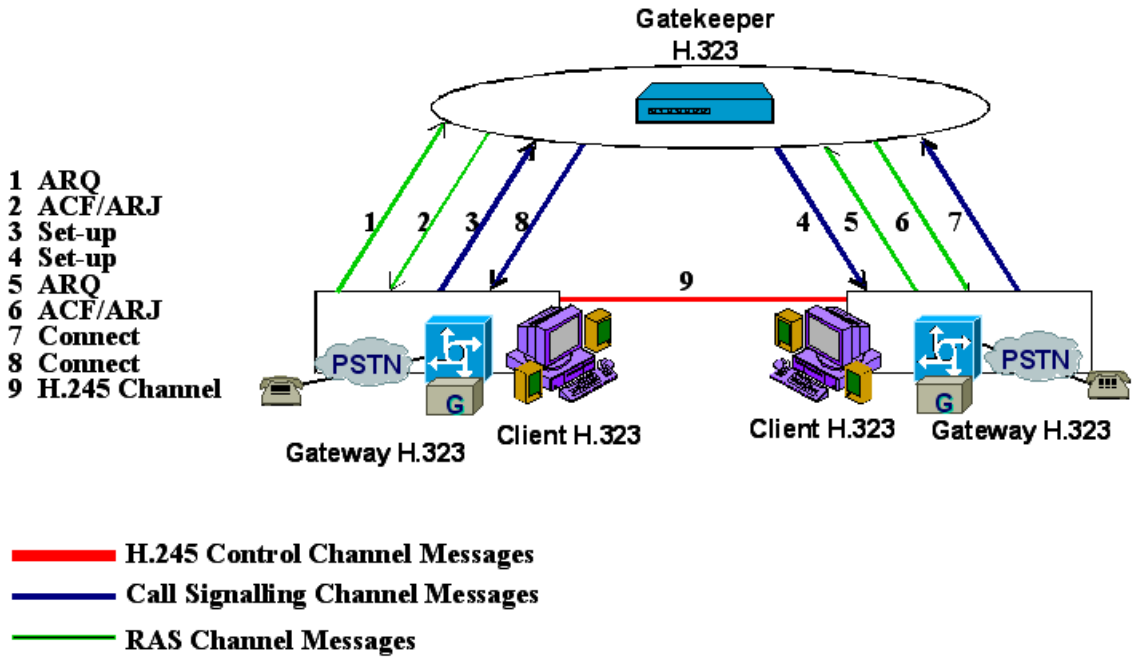
2.2.1.4.2. Gatekeeper routed call signaling model

The *Gatekeeper routed call signaling model* is depicted in Figure 2.5. In this model the H.245 Conference control messages are exchanged directly between the call termination. As each call, the communication starts with an ARQ (Admission ReQuest) message sent by the caller to its Gatekeeper. The ARQ message is used by the endpoint to be allowed to access the packet-based network by the Gatekeeper, which either grants the request with an ACF (Admission ConFirm) or denies it with an ARJ (Admission ReJect). After this first step the Call signaling part of the call begins with the transmission of the SET UP message from the caller to its Gatekeeper. The transport address of the SET UP message (and of all the H.225.0 Call signaling messages) is retrieved by the caller from the "destCallSignalAddress" field carried inside the ACF received, in the case of Gatekeeper routed

call signaling model it is the address of the Gatekeeper itself. The SET UP message is then forwarded by the Gatekeeper (or by the "Gatekeeper network") to the called endpoint. Upon receiving the SET UP message the callee starts its H.225.0 RAS procedure with its Gatekeeper, if successful a CONNECT message is sent to indicate acceptance of the call; because of the call model, also this message is sent to the called endpoint's Gatekeeper which is in charge of forwarding it to the caller endpoint (either directly or using the "Gatekeeper network"). Before sending the CONNECT message, two other messages may be sent from the callee to its Gatekeeper (those two messages are not depicted in the figure since we have reported only mandatory messages):

- ALERTING message, this message may be sent by the called user to indicate that called user alerting has been initiated (in everyday terms, the "phone is ringing");
- CALL PROCEEDING message, this message may be sent by the called user to indicate that requested call establishment has been initiated and no more call establishment information will be accepted.

Figure 2.5. Gatekeeper Routed call signaling model



The two optional messages listed above are then forwarded by the Gatekeeper (or by the "Gatekeeper network") to the caller. After receiving the CONNECT message, the caller starts the procedures H.245 Conference control channel procedures directly with the callee (the correct "h245Address" was retrieved from the CONNECT message itself). The H.245 Conference control channel procedure scopes are the same detailed above, please refer to Direct signaling model section for details.

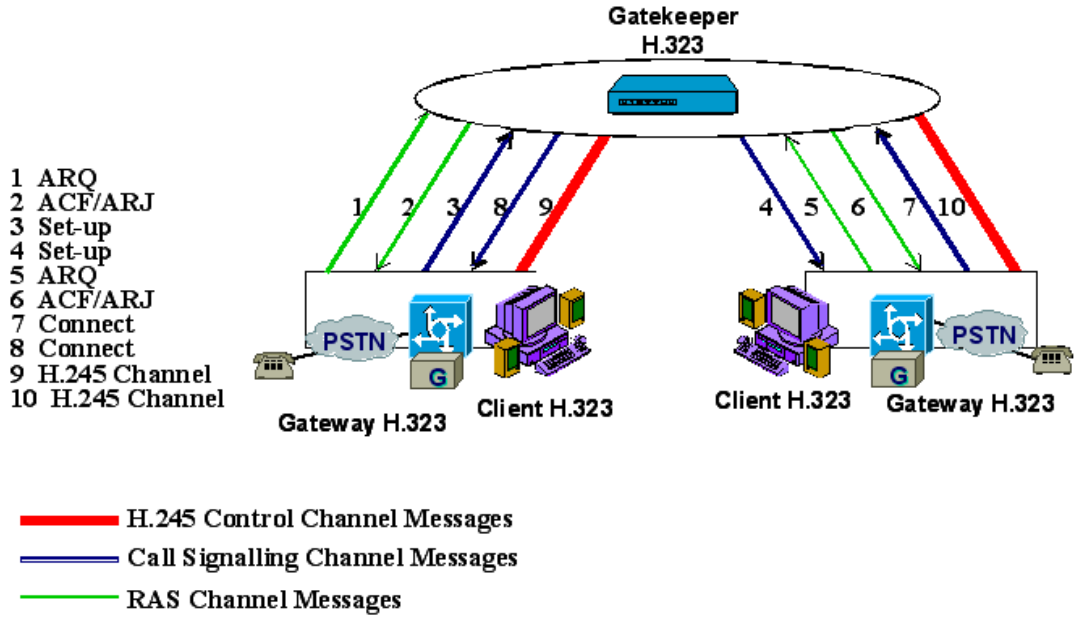
2.2.1.4.3. Gatekeeper routed H.245 control model

The *Gatekeeper routed H.245 control model* is depicted in Figure 2.6. In this model only the media streams are exchanged directly between the call termination. As each call, the communication starts with an ARQ (Admission ReQuest) message sent by the caller to its Gatekeeper. The ARQ message is used by the endpoint to be allowed to access the packet-based network by the Gatekeeper, which either grants the request with an ACF (Admission ConFirm) or denies it with an ARJ (Admission ReJect). After this first step the Call signaling part of the call begins with the transmission of the SET UP message from the caller to its Gatekeeper. The transport address of the SET UP message (and of all the H.225.0 Call signaling messages) is retrieved by the caller from the "destCallSignalAddress" field carried inside the ACF received, in the case of Gatekeeper routed H.245 control model it is the address of the Gatekeeper itself. The SET UP message is then forwarded by the Gatekeeper (or by the "Gatekeeper network") to the called endpoint. Upon receiving the SET UP message the callee starts its H.225.0 RAS procedure with its Gatekeeper, if successful a CONNECT message is sent to indicate acceptance

of the call; because of the call model, also this message is sent to the called endpoint's Gatekeeper which is in charge of forwarding it to the caller endpoint (either directly or using the "Gatekeeper network"). Before sending the CONNECT message, two other messages may be sent from the callee to its Gatekeeper (those two messages are not depicted in the figure since we have reported only mandatory messages):

- ALERTING message, this message may be sent by the called user to indicate that called user alerting has been initiated (in everyday terms, the "phone is ringing");
- CALL PROCEEDING message, this message may be sent by the called user to indicate that requested call establishment has been initiated and no more call establishment information will be accepted.

Figure 2.6. Gatekeeper Routed H.245 control model



The two optional messages listed above are then forwarded by the Gatekeeper (or by the "Gatekeeper network") to the caller. After receiving the CONNECT message, the caller starts the H.245 Conference control channel procedures with its Gatekeeper (the correct "h245Address" was retrieved from the CONNECT message itself). All the H.245 channel messages are then exchanged by the endpoints with their Gatekeeper (or Gatekeepers), it is the Gatekeeper (or "Gatekeeper network") which takes care of forwarding them up to the remote endpoint as foreseen by the Gatekeeper routed H.245 control model. The H.245 Conference control channel procedure scopes are the same detailed above, please refer to Direct signaling model section for details.

2.2.1.5. Communication Phases

In a H.323 communication may be identified 5 different phases:

- Call set up;
- Initial communication and capability exchange;
- Establishment of audiovisual communication;
- Call services;
- Call termination.

2.2.1.5.1. Call set up

Recommendation H.225.0 defines the Call set up messages and procedures here detailed. The recommendation foresees that requests for bandwidth reservation should take place at the earliest possible phase. Differently from other protocols, there is no explicit synchronization between two endpoints during the call setup procedure (two endpoints can send a Setup message each other at exactly the same time). Actions to be taken when problems of synchronization during SET UP message exchange arise are resolved by the application itself; applications not supporting multiple simultaneous calls should issue busy signal when they have an outstanding SET UP message, while applications supporting multiple simultaneous call should issue a busy signal only to the same endpoint to which they sent an outstanding SET UP message. Moreover, an endpoint shall be capable of sending the ALERTING messages. Alerting has the meaning that the called party has been alerted of an incoming call ("phone ringing" in the language of the old telephony). Only the ultimate called endpoint shall originate the ALERTING message and only when the application has already alerted the user. If a Gateway is involved, the Gateway shall send ALERTING when it receives a ring indication from the Switched Circuit Network (SCN). The sending of an ALERTING message is not required if an endpoint can respond to a SET UP message with a CONNECT, CALL PROCEEDING, or RELEASE COMPLETE within 4 seconds. After successfully sending a SET UP message an endpoint can expect to receive either an ALERTING, CONNECT, CALL PROCEEDING, or RELEASE COMPLETE message within 4 seconds after successful transmission. Finally, to maintain the consistency of the meaning of the CONNECT message between packet based networks and circuit switched networks, the CONNECT message should be sent only if it is certain that the capability exchange will successfully take place and a minimum level of communications can be performed.

The Call set up phase may have different realizations, basically we can identify different call set up:

- Basic call setup when neither endpoint are registered, in this call set up the two endpoints communicate directly;
- Both endpoints registered to the same gatekeeper, in this call set up the communication is decided by the signaling model configured on the Gatekeeper;
- Only calling endpoint has gatekeeper, in this call set up only the caller sends messages to the Gatekeeper depending on the signaling models configured while the callee sends the messages directly to the caller endpoint;
- Only called endpoint has gatekeeper, in this call set up only the callee sends messages to the Gatekeeper depending on the signaling models configured while the caller sends the messages directly to the called endpoint;
- Both endpoints registered to different gatekeepers, each of the two endpoints communicate with their Gatekeeper depending on the signaling model configured, additional H.225.0 RAS messages may be exchanged between gatekeeper in order to retrieve location information (see Locating zone external targets section for more details);
- Call set up with Fast connect procedure, in this call set up the media channels are established using either the "Fast Connect" procedure. The Fast Connect procedure speeds up the establishment of a basic point-to-point call (only one round-trip message exchange is needed), enabling immediate media stream delivery upon call connection. The Fast connect procedure is started if the calling endpoint initiates it by sending a SETUP message containing the fastStart element (to advice it is going to use the Fast Connect procedure). Such element contains, among the other things, a sequence of all of the parameters necessary to immediately open and begin transferring media on the channels. Fast Connect procedure may be refused by the called endpoint (motivations may be either because it wants to use features requiring use of H.245 or because it does not implement it). Fast Connect procedure may be refused with any H.225.0 Call signaling message up to and including the CONNECT one. Refusing the Fast Connect procedure (or not initiating it) requires that H.245 procedures be used for capabilities exchange and opening of media channels. Moreover, the Fast Connect procedure allows to have a more detailed view on H.323/SIP gatewaying (further details to be found in Chapter 4);
- Call setup via gateways, when a gateway is involved the call setup between it and the network endpoint is the same as the endpoint-to-endpoint call set up;
- Call setup with an MCU, when an MCU is involved all endpoints exchange call signaling with the MCU (and with the interested Gatekeepers if any). No changes are foreseen between an endpoint and the MCU call set up since it proceeds the same as the endpoint-to-endpoint;

- Broadcast call setup, this kind of call set up follows the procedures defined in Recommendation H.332.

2.2.1.5.2. Initial communication and capability exchange

After exchanging call setup messages, the endpoints shall, if they plan to use H.245, establish the H.245 Control Channel. The H.245 Control Channel is used for the capability exchange and to open the media channels. The H.245 Control channel procedures shall either not be started or closed if CONNECT does not arrive (an H.245 Control channel can be opened on reception of ALERTING or CALL PROCEEDING messages, too) or an endpoint sends RELEASE COMPLETE. H.323 endpoints shall support the capabilities exchange procedure of H.245. The H.245 TERMINALCAPABILITYSET message is used for endpoint system capabilities exchange. This message shall be the first H.245 message sent. Master-slave determination procedure of H.245 has to be supported by H.323 compliant endpoints as a must. In cases of multipoint conferencing (MC) capability is present in more than one endpoint, the master-slave determination is used for determining which MC will play an active role. The H.245 Control channel procedure also provides master-slave determination for opening bidirectional channels for data. After Terminal Capability Exchange has been initiated, master-slave determination procedure (consisting of either MASTERSLAVEDETERMINATION or MASTERSLAVEDETERMINATION-ACK) has to be started as the first H.245 Conference control procedure. Upon failure of initial capability exchange or master-slave determination procedures a maximum of two retries shall be performed before the endpoint passes to the Call Termination phase. Normally, after successful completion of the requirements of this phase, the endpoints shall proceed directly to Establishment of audiovisual communication phase.

2.2.1.5.2.1. Encapsulation of H.245 messages within H.225.0 Call signaling messages

Encapsulation of H.245 messages inside H.225.0 Call signaling messages instead of establishing a separate H.245 channel is possible in order to save resources, synchronize call signaling and control, and reduce call setup time. This process is named as "encapsulation" or "tunneling" of H.245 messages. This procedure allows the terminal to copy the encoded H.245 message using one structure inside the data of the Call Signaling Channel. If tunneling is used, any H.225.0 Call signaling message may contain one or more H.245 messages. If there is no need of sending an H.225.0 Call signaling message when an H.245 message has to be transmitted, a FACILITY message shall be sent detailing (with appropriate fields inside) the reason of such a message.

2.2.1.5.3. Establishment of audiovisual communication

The Establishment of audiovisual communication shall follow the procedures of Recommendation H.245. Open logical channels for the various information streams are opened using the H.245 procedures. The audio and video streams are transported using an unreliable protocol while data communications are transported using a reliable protocol. The transport address that the receiving endpoint has assigned to a specific logical channel (audio, video or data) is transported by the OPENLOGICALCHANNELACK message (an example is given in Figure 2.7). That transport address is used to transmit the information stream associated with that logical channel.

Figure 2.7. OPENLOGICALCHANNELACK message content

```
⊟ITU-T Recommendation H.245
  ⊟response
    ⊟openLogicalChannelAck
        forwardLogicalChannelNumber: 258
      ⊟forwardMultiplexAckParameters (h2250LogicalChannelAckParameters)
        ⊟h2250LogicalChannelAckParameters
            sessionID: 1
          ⊟mediaChannel (unicastAddress)
            ⊟unicastAddress
              ⊟iPAddress
                  network: 131.114.9.124 (131.114.9.124)
                  tsapIdentifier: 49608
          ⊟mediaControlChannel (unicastAddress)
            ⊟unicastAddress
              ⊟iPAddress
                  network: 131.114.9.124 (131.114.9.124)
                  tsapIdentifier: 49609
            flowControlToZero: False
```

2.2.1.5.4. Call services

When the call is active, the terminal may request additional call services, among those we report here on the Bandwidth changes services and on the Supplementary services. As regards as Bandwidth changes services, during a conference, the endpoints or Gatekeeper (if involved) may, at any time, request an increase or decrease in the call bandwidth. If the aggregate bit rate of all transmitted and received channels does not exceed the current call bandwidth then an endpoint may change the bit rate of a logical channel without requesting a bandwidth change. After requesting for bandwidth change, the endpoint shall wait for confirmation prior to actually changing the bit rate (confirmation usually comes from the Gatekeeper). Asking call bandwidth changes is performed using a BANDWIDTH CHANGE REQUEST (BRQ) message, if the request is not accepted, a BANDWIDTH CHANGE REJECT (BRJ) message is returned to endpoint. If the request is accepted, a BANDWIDTH CHANGE CONFIRM (BCF) is sent back to the endpoint. As regards as Supplementary services, support for them is optional. The H.450-Series of Recommendations describe a method of providing Supplementary Services in the H.323 environment. Figure 2.8 reports some of the supplementary services defined so far and their Recommendation number.

Figure 2.8. Supplementary services of the H.450-Series

Recommendation number	Recommendation Title
H.450.1	Supplementary Services Framework
H.450.2	Call Transfer Supplementary Service
H.450.3	Call Diversion Supplementary Service
H.450.4	Call Hold Supplementary Service
H.450.5	Call Park and Pickup Supplementary Service
H.450.6	Call Waiting Supplementary Service
H.450.7	Message Waiting Indication Supplementary Service
H.450.8	Name Identification Supplementary Service
H.450.9	Call Completion Supplementary Service
H.450.10	Call Offer Supplementary Service
H.450.11	Call Intrusion Supplementary Service

2.2.1.5.5. Call termination

A call may be terminated either by both endpoint or by the Gatekeeper. Call termination is defined using the following procedure:

- video should be terminated after a complete picture and then all logical channels for video closed;
- data transmission should be terminated and then all logical channels for data closed;
- audio transmission should be terminated and then all logical channels for audio closed;
- the H.245 ENDSESSIONCOMMAND message (H.245 Control Channel) should be sent by the endpoint/ Gatekeeper, this message indicates that the call has to be disconnected, then the H.245 message transmission should be terminated;
- the ENDSESSIONCOMMAND message should be sent back to the sending endpoint and then the H.245 Control Channel should be closed;
- a RELEASE COMPLETE message should be sent closing the Call Signaling channel if this is still open;

An endpoint receiving ENDSESSIONCOMMAND message does not need to receive it back again after replying to it in order to clear a call. Terminating a call within a conference does not mean the all conference needs to be terminated. In order to terminate a conference an H.245 message (DROPCONFERENCE) is used, then the MC should terminate the calls with the endpoint as described above.

A call may be terminated differently depending on the Gatekeeper presence and on the party issuing the call termination:

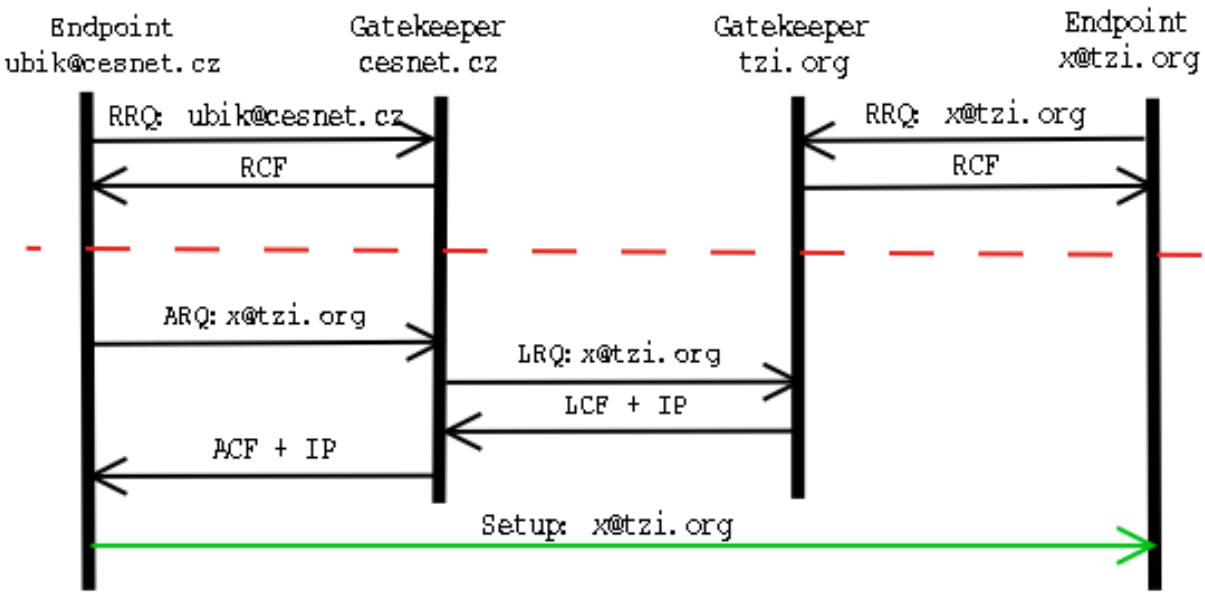
- *Call clearing without a Gatekeeper* - No further action is required.
- *Call clearing with a Gatekeeper* - The Gatekeeper needs to be informed about the Call termination. After RELEASE COMPLETE is sent, an H.225.0 DISENGAGE REQUEST (DRQ) message should be sent by each endpoint to its Gatekeeper. A Disengage Confirm (DCF) message is sent back to the endpoints to acknowledge the reception.
- *Call clearing issued by the Gatekeeper* - A call may be terminated by the Gatekeeper by sending a DRQ to an endpoint. The procedure described above for Call termination should be immediately followed by the endpoint up to the RELEASE COMPLETE message included,then a reply to the Gatekeeper should be sent using a DCF message. The other endpoint should follow the same Call termination procedures upon receiving the ENDSESSIONCOMMAND message. Moreover, if a multipoint conference is taking place, in order to close the entire conference, the Gatekeeper should send a DRQ to each endpoint in the conference.

2.2.1.6. Locating zone external targets

When calling an address that is registered at the same gatekeeper the callee is registered with is simple - the gatekeeper just needs to look up its internal tables to resolve the address. It is more complex if the destination is registered with another gatekeeper. While Chapter 7 will cover this topic more detailed the most basic mechanism H.323 provides shall be explained here.

A gatekeeper may explicitly request the resolution of an address from other gatekeepers. On receipt of an request to call an address that the gatekeeper hasn't registered it can send out a location request (LRQ) to other gatekeepers (see figure Figure 2.9). The receiving gatekeeper - assuming it knows the address - will reply with the TSAP (IP+Port) of either the requested address or its own call signaling TSAP.

Figure 2.9. External address resolution using LRQs



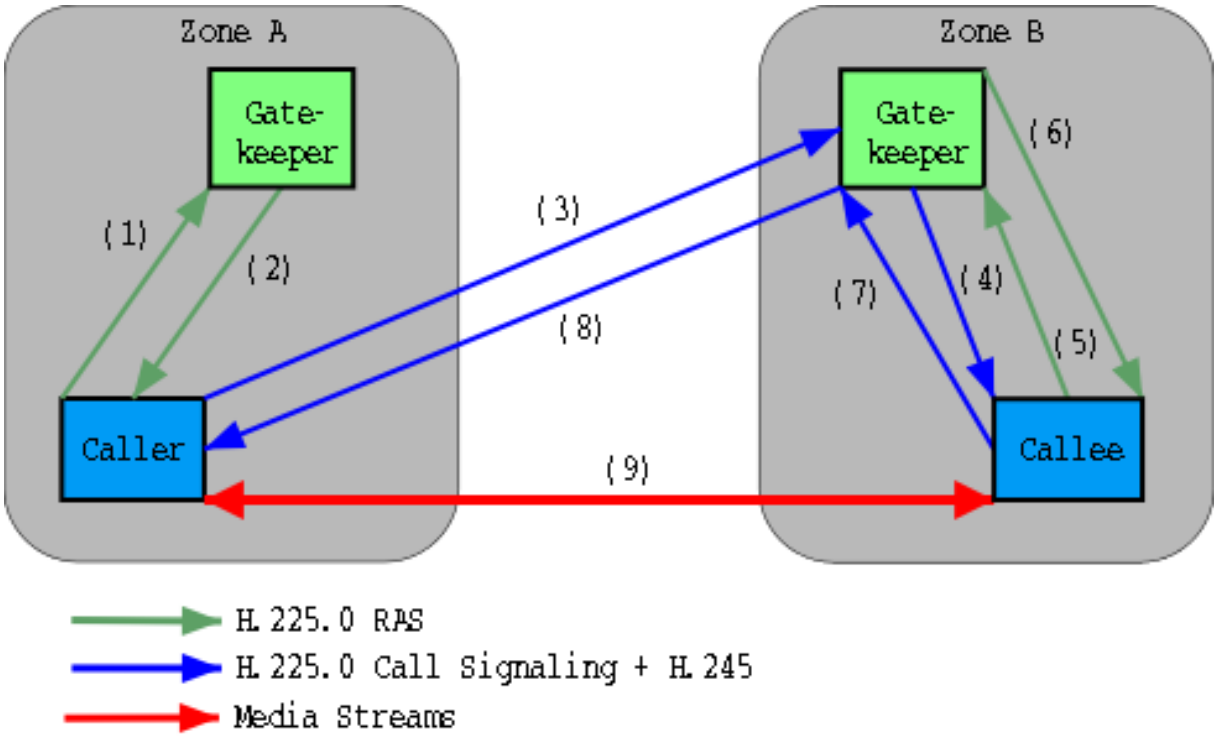
A location request can be sent via Unicast or Multicast. If sent via Multicast only the gatekeeper that can resolve the address shall reply. If a gatekeeper receives a unicast LRQ it shall either confirm or reject the request.

This mechanism can be used to have a list of peer gatekeeper to ask parallel or sequential. It is also possible to assign a domain suffix or number prefix to each peer so an address with a number prefix of an institution will result in a request to the gatekeeper of that institution. By defining default peers one could also build a hierarchy of gatekeepers (Again, see Chapter 7 for more details.)

2.2.1.7. Sample Call Scenario

Figure 2.10 depicts an example for inter-zone call setup using H.323. The caller in zone A contacts its gatekeeper to ask for permission to call the callee in zone B (1). Gatekeeper of zone A confirms this request and provides the caller with the address of zone B's gatekeeper (2).1 The caller establishes a call signaling channel (and subsequently / in parallel the conference control channel) to the gatekeeper of zone B (3), which determines the location of the callee and forwards the request to the callee (4).

Figure 2.10. Sample H.323 Call Setup Scenario



The callee explicitly confirms with its gatekeeper that it is allowed to accept the call (5, 6) and, if yes, alerts the recipient of the call, returns an alerting indication and (once the receiving user picks up the call) eventually an indication of successful connection setup back to the caller (7, 8). In (parallel to) this exchange, capability negotiation and media stream configuration take place. When the setup has completed, both parties start sending media streams directly to each other.

2.2.1.8. Additional (Call) Services

As known from daily interaction with PBXes, telephony service comprises far more than just call setup and teardown: n-way conferencing and various supplementary services (such as call transfer, call waiting, etc.) are available. Similar features - at least the more commonly known and used ones - need to be provided by IP telephony systems as well to be accepted by customers. Additional call services in H.323 can be grouped into three categories:

- *Conferencing* -- H.323 inherently supports multipoint tightly-coupled conferencing - i.e. conferences with access control, optional support for conference chairs, and close synchronization of conference state among all participants - from the outset: through the concept of a Multipoint Controller and an optional Multipoint Processor. While control is centralized in the MC, data exchange may be either via IP multicast, multi-unicast (i.e. peer-wise fan-out between endpoints without MP), or through an MP. The distribution mode may be selected per-media and per endpoint peer and is controlled by the MC.

- *"Broadcast conferencing"* -- H.323 also provides an interface to support large loosely-coupled conferences as are frequently used in the Mbone to multicast seminars, events, etc. In this case, the MC defines session description (using the Session Description Protocol, SDP, see below) for the H.323 media sessions (which have to operate using multicast) and announces this description by some means (e.g. the Session Announcement Protocol, SAP). Details are defined in ITU-T H.332.

- *Supplementary Services* -- H.323 provides a variety of supplementary services with additional ones continuously being defined. While some services can be accomplished using the basic H.323 specifications, the H.450.x Recommendations define a framework (derived from QSIG, the ECMA/ISO/ETSI standard for supplementary service signaling in PBXes) and a number of services (call transfer, call diversion, call hold, call park & pickup, call waiting, message waiting indication, call completion).

Further extensions for supplementary services and other functional enhancements are on the way. In particular,

an HTTP-based extension framework is being defined at the time of writing to enable rapid introduction of new services without the need for standardization.

2.2.1.9. H.235 Security

The H.235 recommendation defines ways of security for H.323. This includes:

- *Authentication* - Authentication can be achieved by using a shared secret (password) or digital signatures. The RAS messages include a token that was generated using either the shared secret or the signature. A receiving entity authenticate the sender by comparing the received token with a self generated token.
- *Message Integrity* - Integrity is achieved by generating a password based checksum over the message.
- *Privacy* - Mechanisms are provided to setup encryption on the media streams. They must be used in conjunction with the H.245 protocol and use DES, Triple DES or RC2 - the use of SRTP isn't supported yet (in H.235v2).

Those mechanisms are grouped into so called *Security Profiles*, where the *Baseline Security Profile* provides authentication and message integrity making it suitable for subscription based environments and the *Voice Encryption Profile* that provides confidential end-to-end media channels.

2.2.1.10. Protocol Profiles

H.323 has its origin - as mentioned before - in the area of multimedia conferencing. This implies that a vast number of options are available which are not necessary for providing telephony services. The TIPHON project of the European Telecommunication Standards Institute (ETSI) has defined a Telephony Profile for H.323 that specifies which combination of options should be implemented.

Similarly, H.323 contains a security framework (H.235) that describes a collection of algorithms and protocol mechanisms but lacks - because of international political constraints - a precise specification of a mandatory baseline. This is accounted for by the ETSI TIPHON security profile: this specification fills in the gaps and provides the foundation for inter operable implementations.

In summary, it can be said that the H.323 family of standards provides a mature basis for commercial products in the field of IP telephony. While the details of the protocol are often dominated by their legacy from various earlier ITU protocols, there is an active effort to profile and simplify the protocol to reduce the complexity.

2.2.2. SIP

2.2.2.1. Purpose of SIP

SIP stands for Session Initiation Protocol. It is an application-layer control protocol which has been developed and designed within the IETF[5]. The protocol has been designed with easy implementation, good scalability, and flexibility in mind.

The specification is available in form of several RFCs, the most important one is RFC3261[6] which contains the core protocol specification. The protocol is used for creating, modifying, and terminating sessions with one or more participants. By sessions we understand a set of senders and receivers that communicate and the state kept in those senders and receivers during the communication. Examples of a session can include Internet telephone calls, distribution of multimedia, multimedia conferences, distributed computer games, etc.

SIP is not the only protocol that the communicating devices will need. It is not meant to be a general purpose protocol. Purpose of SIP is just to make the communication possible, the communication itself must be achieved by another means (and possibly another protocol). Two protocols that are most often used along with SIP are RTP and SDP. RTP protocol is used to carry the real-time multimedia data (including audio, video, and text), the protocol makes it possible to encode and split the data into packets and transport such packets over the Internet. Another important protocol is SDP--Session Description Protocol, which is used to describe and encode capabilities of session participants. Such a description is then used to negotiate the characteristics of the session so that

[5] http://www.ietf.org
[6] http://www.ietf.org/rfc/rfc3261.txt

all the devices can participate (that includes, for example, negotiation of codecs used to encode media so all the participants will be able to decode it, negotiation of transport protocol used and so on).

SIP has been designed in conformance with the Internet model. It is an end-to-end oriented signaling protocol which means, that all the logic is stored in end devices (except routing of SIP messages). State is also stored in end-devices only, there is no single point of failure and networks designed this way scale well. The price that we have to pay for the distributiveness and scalability is higher message overhead, caused by the messages being sent end-to-end.

It is worth mentioning that the end-to-end concept of SIP is a significant divergence from regular PSTN (Public Switched Telephone Network) where all the state and logic is stored in the network and end devices (telephones) are very primitive. Aim of SIP is to provide the same functionality that the traditional PSTNs have, but the end-to-end design makes SIP networks much more powerful and open to the implementation of new services that can be hardly implemented in the traditional PSTNs.

SIP is based on HTTP protocol. The HTTP protocol inherited format of message headers from RFC822[7]. HTTP is probably the most successful and widely used protocol in the Internet. It tries to combine the best of the both. In fact, HTTP can be classified as a signaling protocol too, because user agents use the protocol to tell a HTTP server in which documents they are interested in. SIP is used to carry the description of session parameters, the description is encoded into a document using SDP. Both protocols (HTTP and SIP) have inherited encoding of message headers from RFC822[8]. The encoding has proven to be robust and flexible over the years.

2.2.2.1.1. SIP URI

SIP entities are identified using SIP URI (Uniform Resource Identifier). A SIP URI has form of sip:username@domain, for instance, sip:joe@company.com. As we can see, SIP URI consists of username part and domain name part delimited by @ (at) character. SIP URIs are similar to e-mail addresses, it is, for instance, possible to use the same URI for e-mail and SIP communication, such URIs are easy to remember.

2.2.2.2. SIP Network Elements

Although in the simplest configuration it is possible to use just two user agents that send SIP messages directly to each other, a typical SIP network will contain more than one type of SIP elements. Basic SIP elements are user agents, proxies, registrars, and redirect servers. We will briefly describe them in this section.

Note that the elements, as presented in this section, are often only logical entities. It is often profitable to co-locate them together, for instance, to increase the speed of processing, but that depends on a particular implementation and configuration.

2.2.2.2.1. User Agents

Internet end points that use SIP to find each other and to negotiate a session characteristics are called *user agents*. User agents usually, but not necessarily, reside on a user's computer in form of an application--this is currently the most widely used approach, but user agents can be also cellular phones, PSTN gateways, PDAs, automated IVR systems and so on.

User agents are often referred to as *User Agent Server* (UAS) and *User Agent Client* (UAC). UAS and UAC are logical entities only, each user agent contains a UAC and UAS. UAC is the part of the user agent that sends requests and receives responses. UAS is the part of the user agent that receives requests and sends responses.

Because a user agent contains both UAC and UAS, we often say that a user agent behaves like a UAC or UAS. For instance, caller's user agent behaves like UAC when it sends an INVITE requests and receives responses to the request. Callee's user agent behaves like a UAS when it receives the INVITE and sends responses.

But this situation changes when the callee decides to send a BYE and terminate the session. In this case the callee's user agent (sending BYE) behaves like UAC and the caller's user agent behaves like UAS.

Figure 2.11. UAC and UAS

[7] http://www.ietf.org/rfc/rfc822.txt
[8] http://www.ietf.org/rfc/rfc822.txt

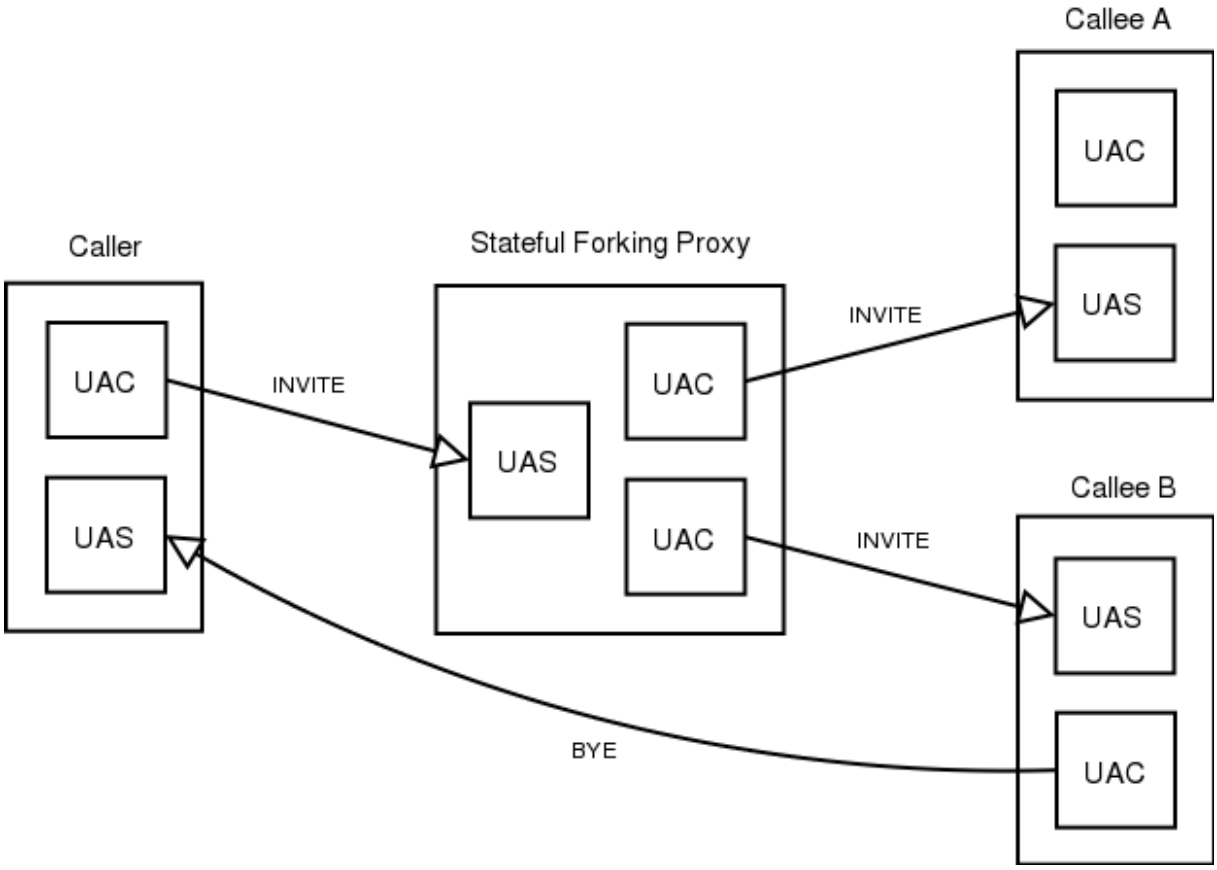


Figure 2.11 shows three user agents and one stateful forking proxy. Each user agent contains UAC and UAS. The part of the proxy that receives the INVITE from the caller in fact acts as a UAS. When forwarding the request statefully the proxy creates two UACs, each of them is responsible for one branch.

In our example callee B picked up and later when he wants to tear down the call it sends a BYE. At this time the user agent that was previously UAS becomes a UAC and vice versa.

2.2.2.2.2. Proxy Servers

In addition to that SIP allows creation of an infrastructure of network hosts called *proxy servers*. User agents can send messages to a proxy server. Proxy servers are very important entities in the SIP infrastructure. They perform routing of a session invitations according to invitee's current location, authentication, accounting and many other important functions.

The most important task of a proxy server is to route session invitations “closer” to callee. The session invitation will usually traverse a set of proxies until it finds one which knows the actual location of the callee. Such a proxy will forward the session invitation directly to the callee and the callee will then accept or decline the session invitation.

There are two basic types of SIP proxy servers--stateless and stateful.

2.2.2.2.2.1. Stateless Servers

Stateless servers are simple message forwarders. They forward messages independently of each other. Although messages are usually arranged into transactions (see Section 2.2.2.4), stateless proxies do not take care of transactions.

Stateless proxies are simple, but faster than stateful proxy servers. They can be used as simple load balancers, message translators and routers. One of drawbacks of stateless proxies is that they are unable to absorb retransmissions of messages and perform more advanced routing, for instance, forking or recursive traversal.

2.2.2.2.2.2. Stateful Servers

Stateful proxies are more complex. Upon reception of a request, stateful proxies create a state and keep the state until the transaction finishes. Some transactions, especially those created by INVITE, can last quite long (until callee picks up or declines the call). Because stateful proxies must maintain the state for the duration of the transactions, their performance is limited.

The ability to associate SIP messages into transactions gives stateful proxies some interesting features. Stateful proxies can perform forking, that means upon reception of a message two or more messages will be sent out.

Stateful proxies can absorb re-transmissions because they know, from the transaction state, if they have already received the same message (stateless proxies cannot do the check because they keep no state).

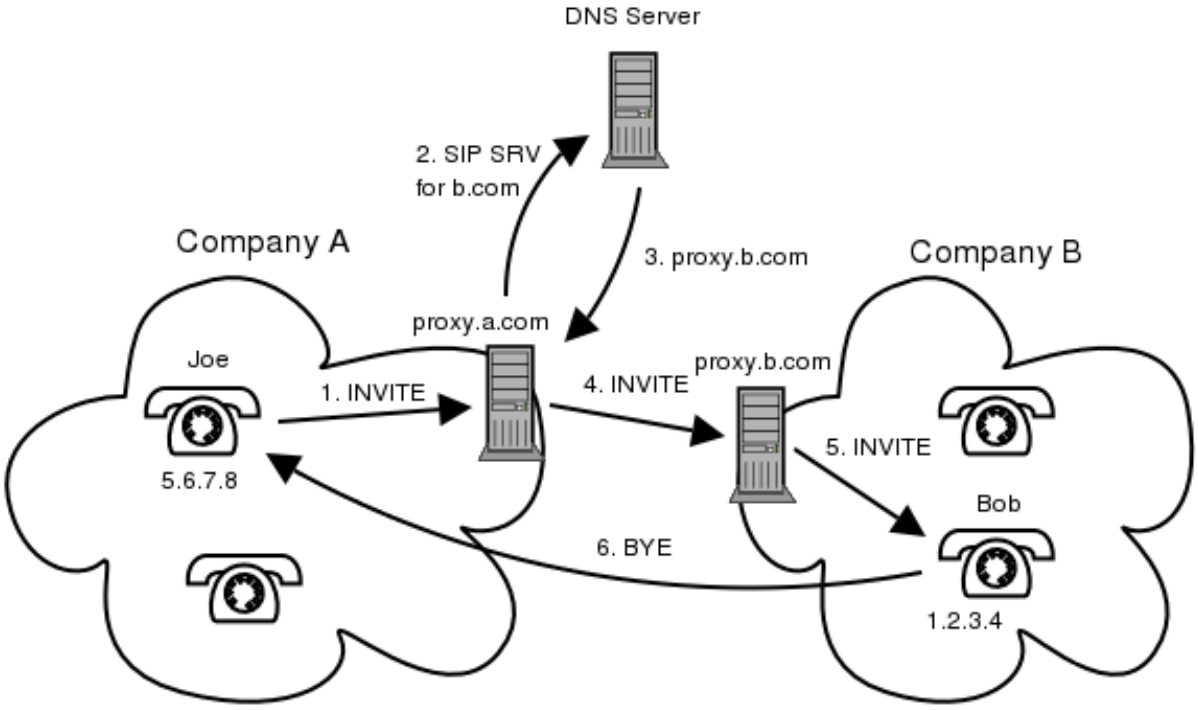
Stateful proxies can perform more complicated methods of finding a user. It is, for instance, possible to try to reach user's office phone and when he doesn't pick up then the call is redirected to his cell phone. Stateless proxies can't do this because they have no way of knowing how the transaction targeted to the office phone finished.

Most SIP proxies today are stateful because their configuration is usually very complex. They often perform accounting, forking, some sort of NAT traversal aid and all those features require a stateful proxy.

2.2.2.2.2.3. Proxy Server Usage

A typical configuration is that each centrally administered entity (a company, for instance) has it's own SIP proxy server which is used by all user agents in the entity. Let's suppose that there are two companies A and B and each of them has it's own proxy server. Figure 2.12 shows how a session invitation from employee Joe in company A will reach employee Bob in company B.

Figure 2.12. Session Invitation



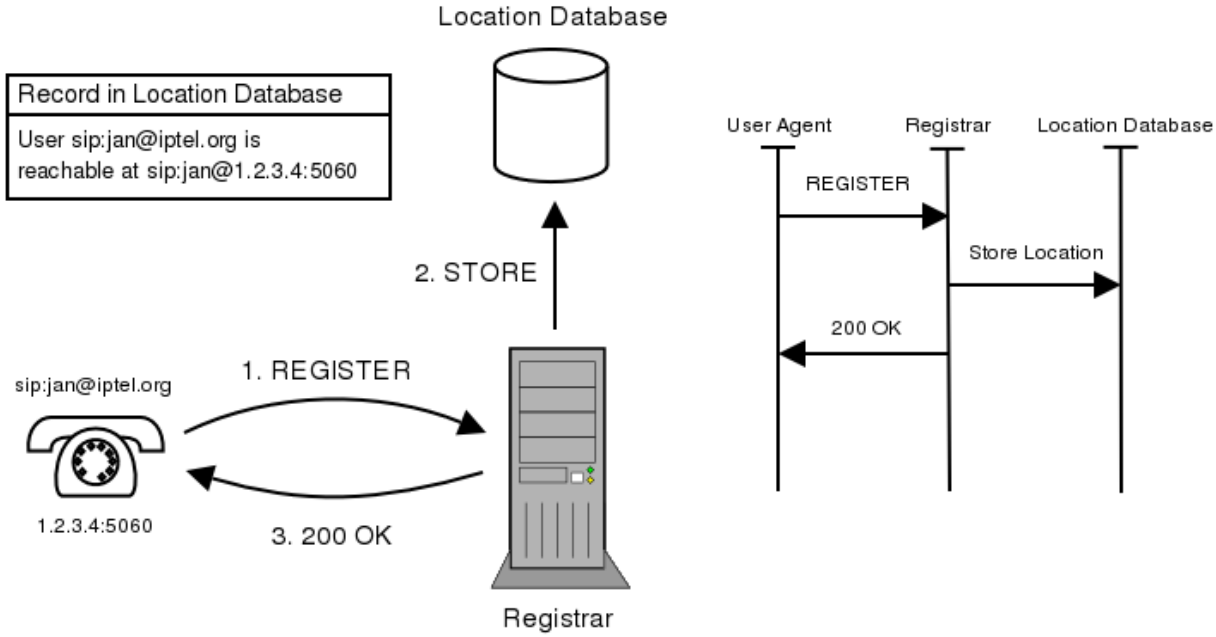
User Joe uses address sip:bob@b.com to call Bob. Joe's user agent doesn't know how to route the invitation itself but it is configured to send all outbound traffic to the company SIP proxy server proxy.a.com. The proxy server figures out that user sip:bob@b.com is in a different company so it will look up B's SIP proxy server and send the invitation there. B's proxy server can be either pre-configured at proxy.a.com or the proxy will use DNS SRV records to find B's proxy server. The invitation reaches proxy.bo.com. The proxy knows that Bob is currently sitting in his office and is reachable through phone on his desk, which has IP address 1.2.3.4, so the proxy will send the invitation there.

2.2.2.2.3. Registrar

We mentioned that the SIP proxy at proxy.b.com knows current Bob's location but haven't mentioned yet how a proxy can learn current location of a user. Bob's user agent (SIP phone) must register with a *registrar*. The registrar is a special SIP entity that receives registrations from users, extracts information about their current location (IP address, port and username in this case) and stores the information into location database. Purpose of the location database is to map sip:bob@b.com to something like sip:bob@1.2.3.4:5060. The location database is then used by B's proxy server. When the proxy receives an invitation for sip:bob@b.com it will search the location database. It finds sip:bob@1.2.3.4:5060 and will send the invitation there. A registrar is very often a logical entity only. Because of their tight coupling with proxies, registrars are usually co-located with proxy servers.

Figure 2.13 shows a typical SIP registration. A REGISTER message containing Address of Record sip:jan@iptel.org and contact address sip:jan@1.2.3.4:5060 where 1.2.3.4 is IP address of the phone, is sent to the registrar. The registrar extracts this information and stores it into the location database. If everything went well then the registrar sends a 200 OK response to the phone and the process of registration is finished.

Figure 2.13. Registrar Overview



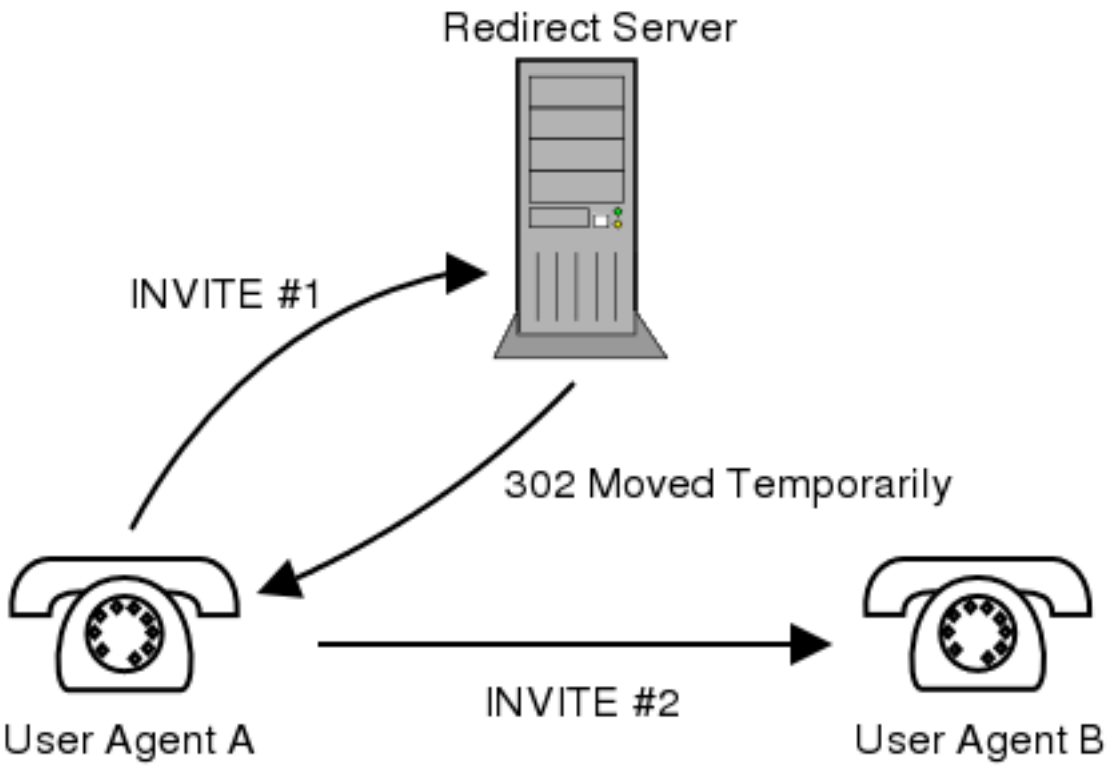
Each registration has a limited life span. Expires header field or expires parameter of Contact header field determines for how long is the registration valid. The user agent must refresh the registration within the life span otherwise it will expire and the user will become unavailable.

2.2.2.2.4. Redirect Server

The entity that receives a request and sends back a reply containing a list of the current location of a particular user is called *redirect server*. A redirect server receives requests and looks up the intended recipient of the request in the location database created by a registrar. It then creates a list of current locations of the user and sends it to the request originator in a response within 3xx class.

The originator of the request then extracts the list of destinations and sends another request directly to them. Figure 2.14 shows a typical redirection.

Figure 2.14. SIP Redirection



2.2.2.3. SIP Messages

Communication using SIP (often called signaling) comprises of series of *messages*. Messages can be transported independently by the network. Usually they are transported in a separate UDP datagram each. Each message consist of “first line”, message header, and message body. The first line identifies type of the message. There are two types of messages--*requests* and *responses*. Requests are usually used to initiate some action or inform recipient of the request of something. Replies are used to confirm that a request was received and processed and contain the status of the processing.

A typical SIP request looks like this:

```
INVITE sip:7170@iptel.org SIP/2.0
Via: SIP/2.0/UDP 195.37.77.100:5040;rport
Max-Forwards: 10
From: "jiri" <sip:jiri@iptel.org>;tag=76ff7a07-c091-4192-84a0-d56e91fe104f
To: <sip:jiri@bat.iptel.org>
Call-ID: d10815e0-bf17-4afa-8412-d9130a793d96@213.20.128.35
CSeq: 2 INVITE
Contact: <sip:213.20.128.35:9315>
User-Agent: Windows RTC/1.0
Proxy-Authorization: Digest username="jiri", realm="iptel.org",
  algorithm="MD5", uri="sip:jiri@bat.iptel.org",
  nonce="3cef753900000001771328f5ae1b8b7f0d742da1feb5753c",
  response="53fe98db10e1074
b03b3e06438bda70f"
Content-Type: application/sdp
Content-Length: 451

v=0
o=jku2 0 0 IN IP4 213.20.128.35
s=session
c=IN IP4 213.20.128.35
b=CT:1000
t=0 0
m=audio 54742 RTP/AVP 97 111 112 6 0 8 4 5 3 101
```

```
a=rtpmap:97 red/8000
a=rtpmap:111 SIREN/16000
a=fmtp:111 bitrate=16000
a=rtpmap:112 G7221/16000
a=fmtp:112 bitrate=24000
a=rtpmap:6 DVI4/16000
a=rtpmap:0 PCMU/8000
a=rtpmap:4 G723/8000
a=rtpmap: 3 GSM/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
```

The first line tells us that this is INVITE message which is used to establish a session. The URI on the first line--sip:7170@iptel.org is called *Request URI* and contains URI of the next hop of the message. In this case it will be host iptel.org.

A SIP request can contain one or more Via header fields which are used to record path of the request. They are later used to route SIP responses exactly the same way. The INVITE message contains just one Via header field which was created by the user agent that sent the request. From the Via field we can tell that the user agent is running on host 195.37.77.100 and port 5060.

From and To header fields identify initiator (caller) and recipient (callee) of the invitation (just like in SMTP where they identify sender and recipient of a message). From header field contains a tag parameter which serves as a dialog identifier and will be described in Section 2.2.2.5.

Call-ID header field is a dialog identifier and it's purpose is to identify messages belonging to the same call. Such messages have the same Call-ID identifier. CSeq is used to maintain order of requests. Because requests can be sent over an unreliable transport that can re-order messages, a sequence number must be present in the messages so that recipient can identify re-transmissions and out of order requests.

Contact header field contains IP address and port on which the sender is awaiting further requests sent by callee. Other header fields are not important and will be not described here.

Message header is delimited from message body by an empty line. Message body of the INVITE request contains a description of the media type accepted by the sender and encoded in SDP.

2.2.2.3.1. SIP Requests

We have described how an INVITE request looks like and said that the request is used to invite a callee to a session. Other important requests are:

- *ACK*--This message acknowledges receipt of a final response to INVITE. Establishing of a session utilizes 3-way hand-shaking due to asymmetric nature of the invitation. It may take a while before the callee accepts or declines the call so the callee's user agent periodically re-transmits a positive final response until it receives an ACK (which indicates that the caller is still there and ready to communicate).
- *BYE*--Bye messages are used to tear down multimedia sessions. A party wishing to tear down a session sends a BYE to the other party.
- *CANCEL*--Cancel is used to cancel not yet fully established session. It is used when the callee hasn't replied with a final response yet but the caller wants to abort the call (typically when a callee doesn't respond for some time).
- *REGISTER*--Purpose of REGISTER request is to let registrar know of current user's location. Information about current IP address and port on which a user can be reached is carried in REGISTER messages. Registrar extracts this information and puts it into a location database. The database can be later used by SIP proxy servers to route calls to the user. Registrations are time-limited and need to be periodically refreshed.

The listed requests usually have no message body because it is not needed in most situations (but can have one). In addition to that many other request types have been defined but their description is out of the scope of this document.

2.2.2.3.2. SIP Responses

When a user agent or proxy server receives a request it sends a reply. Each request must be replied except ACK requests which trigger no replies.

A typical reply looks like this:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.30:5060;received=66.87.48.68
From: sip:sip2@iptel.org
To: sip:sip2@iptel.org;tag=794fe65c16edfdf45da4fc39a5d2867c.b713
Call-ID: 2443936363@192.168.1.30
CSeq: 63629 REGISTER
Contact: <sip:sip2@66.87.48.68:5060;transport=udp>;q=0.00;expires=120
Server: Sip EXpress router (0.8.11pre21xrc (i386/linux))
Content-Length: 0
Warning: 392 195.37.77.101:5060 "Noisy feedback tells:
    pid=5110 req_src_ip=66.87.48.68 req_src_port=5060 in_uri=sip:iptel.org
    out_uri=sip:iptel.org via_cnt==1"
```

As we can see, responses are very similar to the requests, except for the first line. The first line of response contains protocol version (SIP/2.0), reply code, and reason phrase.

The *reply code* is an integer number from 100 to 699 and indicates type of the response. There are 6 classes of responses:

- *1xx* are *provisional* responses. A provisional response is response that tells to its recipient that the associated request was received but result of the processing is not known yet. Provisional responses are sent only when the processing doesn't finish immediately. The sender must stop re-transmitting the request upon reception of a provisional response.

  Typically proxy servers send responses with code 100 when they start processing an INVITE and user agents send responses with code 180 (Ringing) which means that the callee's phone is ringing.

- *2xx* responses are *positive final* responses. A final response is the ultimate response that the originator of the request will ever receive. Therefore final responses express result of the processing of the associated request. Final responses also terminate transactions. Responses with code from 200 to 299 are positive responses that means that the request was processed successfully and accepted. For instance a 200 OK response is sent when a user accepts invitation to a session (INVITE request).

  A UAC may receive several 200 messages to a single INVITE request. This is because a forking proxy (described later) can fork the request so it will reach several UAS and each of them will accept the invitation. In this case each response is distinguished by the tag parameter in To header field. Each response represents a distinct dialog with unambiguous dialog identifier.

- *3xx* responses are used to redirect a caller. A redirection response gives information about the user's new location or an alternative service that the caller might use to satisfy the call. Redirection responses are usually sent by proxy servers. When a proxy receives a request and doesn't want or can't process it for any reason, it will send a redirection response to the caller and put another location into the response which the caller might want to try. It can be the location of another proxy or the current location of the callee (from the location database created by a registrar). The caller is then supposed to re-send the request to the new location. 3xx responses are final.

- *4xx* are *negative final* responses. a 4xx response means that the problem is on the sender's side. The request couldn't be processed because it contains bad syntax or cannot be fulfilled at that server.

- *5xx* means that the problem is on server's side. The request is apparently valid but the server failed to fulfill it. Clients should usually retry the request later.

- *6xx* reply code means that the request cannot be fulfilled at any server. This response is usually sent by a server that has definitive information about a particular user. User agents usually send a 603 Decline response when the user doesn't want to participate in the session.

In addition to the response class the first line also contains *reason phrase*. The code number is intended to be processed by machines. It is not very human-friendly but it is very easy to parse and understand by machines. The reason phrase usually contains a human-readable message describing the result of the processing. A user agent should render the reason phrase to the user.

The request to which a particular response belongs is identified using the CSeq header field. In addition to the sequence number this header field also contains method of corresponding request. In our example it was REGISTER request.

2.2.2.4. SIP Transactions

Although we said that SIP messages are sent independently over the network, they are usually arranged into *transactions* by user agents and certain types of proxy servers. Therefore SIP is said to be a *transactional protocol*.

A transaction is a sequence of SIP messages exchanged between SIP network elements. A transaction consists of one request and all responses to that request. That includes zero or more provisional responses and one or more final responses (remember that an INVITE might be answered by more than one final response when a proxy server forks the request).

If a transaction was initiated by an INVITE request then the same transaction also includes ACK, but only if the final response was not a 2xx response. If the final response was a 2xx response then the ACK is not considered part of the transaction.

As we can see this is quite asymmetric behavior--ACK is part of transactions with a negative final response but is not part of transactions with positive final responses. The reason for this separation is the importance of delivery of all 200 OK messages. Not only that they establish a session, but also 200 OK can be generated by multiple entities when a proxy server forks the request and all of them must be delivered to the calling user agent. Therefore user agents take responsibility in this case and retransmit 200 OK responses until they receive an ACK. Also note that only responses to INVITE are retransmitted !

SIP entities that have notion of transactions are called *stateful*. Such entities usually create a state associated with a transaction that is kept in the memory for the duration of the transaction. When a request or response comes, a stateful entity tries to associate the request (or response) to existing transactions. To be able to do it it must extract a unique transaction identifier from the message and compare it to identifiers of all existing transactions. If such a transaction exists then it's state gets updated from the message.

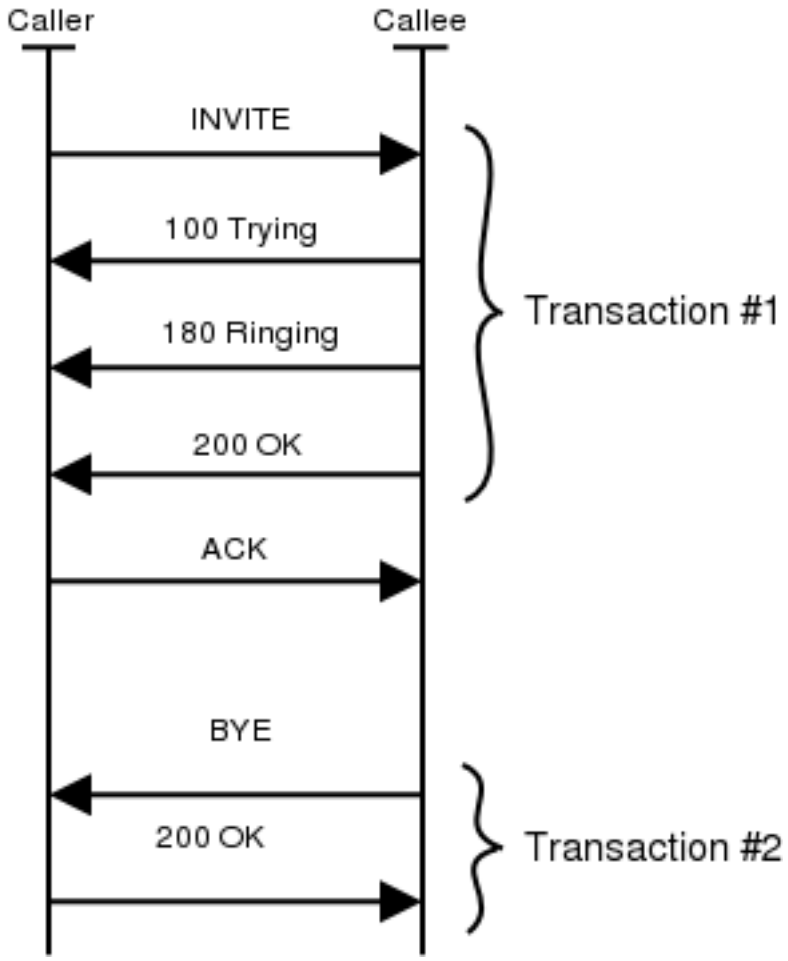
In the previous SIP RFC2543[9] the transaction identifier was calculated as hash of all important message header fields (that included To, From, Request-URI and CSeq). This proved to be very slow and complex, during interoperability tests such transaction identifiers used to be a common source of problems.

In the new RFC3261[10] the way of calculating transaction identifiers was completely changed. Instead of complicated hashing of important header fields a SIP message now includes the identifier directly. Branch parameter of Via header fields contains directly the transaction identifier. This is significant simplification, but there still exist old implementations that don't support the new way of calculating of transaction identifier so even new implementations have to support the old way. They must be backwards compatible.

Figure 2.15 shows what messages belong to what transactions during a conversation of two user agents.

Figure 2.15. SIP Transactions

[9] http://www.ietf.org/rfc/rfc2543.txt
[10] http://www.ietf.org/rfc/rfc3261.txt

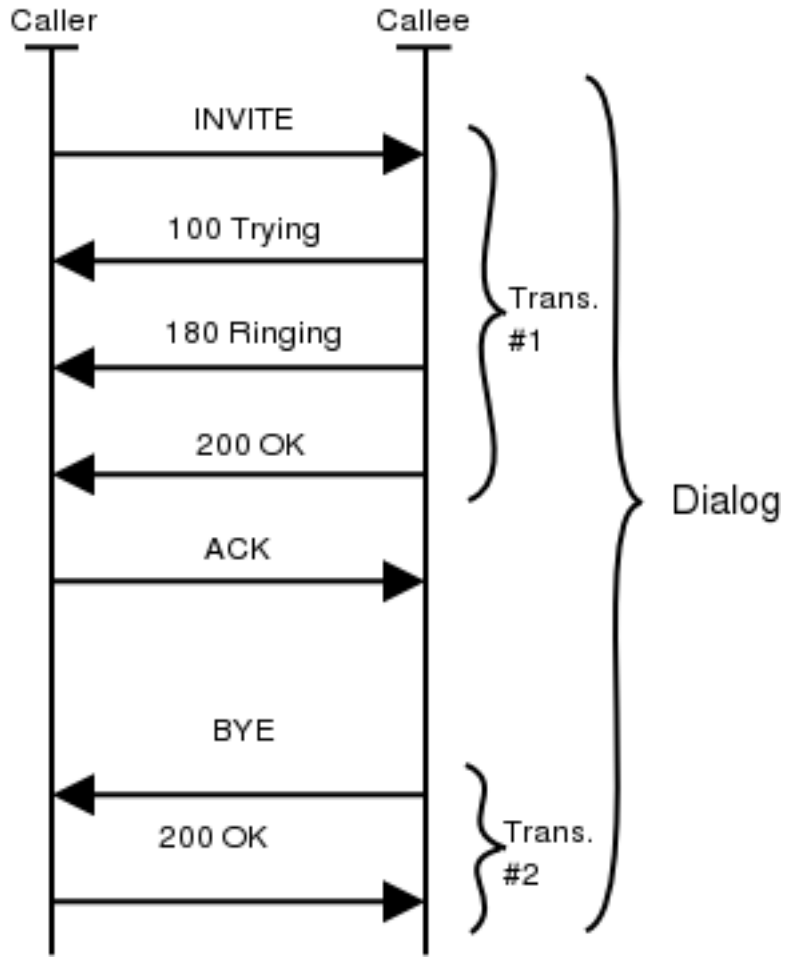


2.2.2.5. SIP Dialogs

We have shown what transactions are, that one transaction includes INVITE and it's responses and another transaction includes BYE and it responses when a session is being torn down. But we feel that those two transactions should be somehow related--both of them belong to the same *dialog*. A dialog represents a peer-to-peer SIP relationship between two user agents. A dialog persists for some time and it is very important concept for user agents. Dialogs facilitate proper sequencing and routing of messages between SIP endpoints.

Dialogs are identified using Call-ID, From tag, and To tag. Messages that belong to the same dialog must have these fields equal. We have shown that CSeq header field is used to order messages, in fact it is used to order messages within a dialog. The number must be monotonically increased for each message sent within a dialog otherwise the peer will handle it as out of order request or retransmission. In fact, the CSeq number identifies a transaction within a dialog because we have said that requests and associated responses are called transaction. This means that only one transaction in each direction can be active within a dialog. One could also say that a *dialog is a sequence of transactions*. Figure 2.16 extends Figure 2.15 to show which messages belong to the same dialog.

Figure 2.16. SIP Dialog



Some messages establish a dialog and some do not. This allows to explicitly express the relationship of messages and also to send messages that are not related to other messages outside a dialog. That is easier to implement because user agent don't have to keep the dialog state.

For instance, INVITE message establishes a dialog, because it will be later followed by BYE request which will tear down the session established by the INVITE. This BYE is sent within the dialog established by the INVITE.

But if a user agent sends a MESSAGE request, such a request doesn't establish any dialog. Any subsequent messages (even MESSAGE) will be sent independently of the previous one.

2.2.2.5.1. Dialogs Facilitate Routing

We have said that dialogs are also used to route the messages between user agents, let's describe this a little bit.

Let's suppose that user sip:bob@a.com wants to talk to user sip:pete@b.com. He knows SIP address of the callee (sip:pete@b.com) but this address doesn't say anything about current location of the user--i.e. the caller doesn't know to which host to send the request. Therefore the INVITE request will be sent to a proxy server.

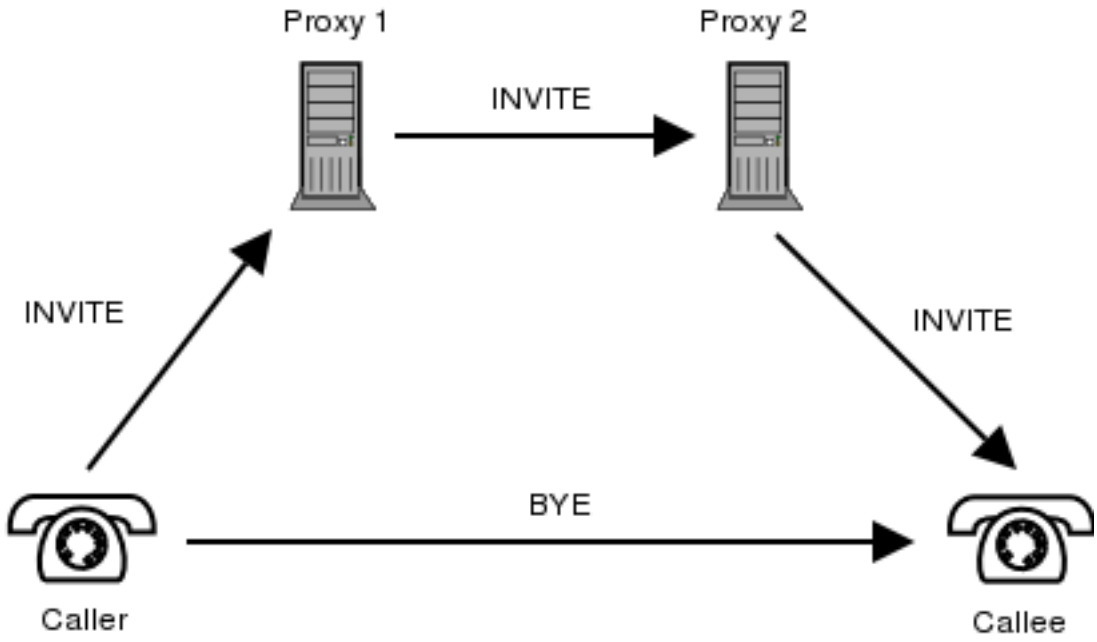
The request will be sent from proxy to proxy until it reaches one that knows current location of the callee. This process is called routing. Once the request reaches the callee, the callee's user agent will create a response that will be sent back to the caller. Callee's user agent will also put Contact header field into the response which will contain the current location of the user. The original request also contained Contact header field which means that both user agents know the current location of the peer.

Because the user agents know location of each other, it is not necessary to send further requests to any proxy--they can be sent directly from user agent to user agent. That's exactly how dialogs facilitate routing.

Further messages within a dialog are sent directly from user agent to user agent. This is a significant performance improvement because proxies do not see all the messages within a dialog, they are used to route just the

first request that establishes the dialog. The direct messages are also delivered with much smaller latency because a typical proxy usually implements complex routing logic. Figure 2.17 contains an example of a message within a dialog (BYE) that bypasses the proxies.

Figure 2.17. SIP Trapezoid



2.2.2.5.2. Dialog Identifiers

We have already shown that dialog identifiers consist of three parts, Call-Id, From tag, and To tag, but it is not that clear why are dialog identifiers created exactly this way and who contributes which part.

Call-ID is so called *call identifier*. It must be a unique string that identifies a call. A call consists of one or more dialogs. Multiple user agents may respond to a request when a proxy along the path forks the request. Each user agent that sends a 2xx establishes a separate dialog with the caller. All such dialogs are part of the same call and have the same Call-ID.

From tag is generated by the caller and it uniquely identifies the dialog in the caller's user agent.

To tag is generated by a callee and it uniquely identifies, just like From tag, the dialog in the callee's user agent.

This hierarchical dialog identifier is necessary because a single call invitation can create several dialogs and caller must be able to distinguish them.

2.2.2.6. Typical SIP Scenarios

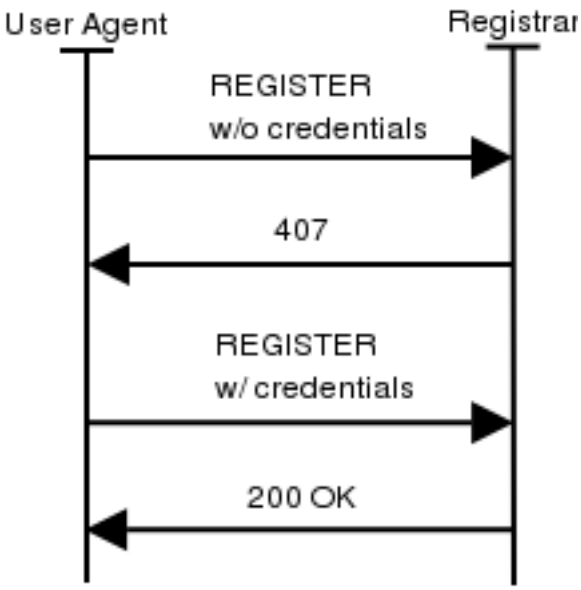
This section gives a brief overview of typical SIP scenarios that usually make up the SIP traffic.

2.2.2.6.1. Registration

Users must register themselves with a registrar to be reachable by other users. A registration comprises a REGISTER message followed by a 200 OK sent by registrar if the registration was successful. Registrations are usually authorized so a 407 reply can appear if the user didn't provide valid credentials. Figure 2.18 shows an ex-

ample of registration.

Figure 2.18. REGISTER Message Flow

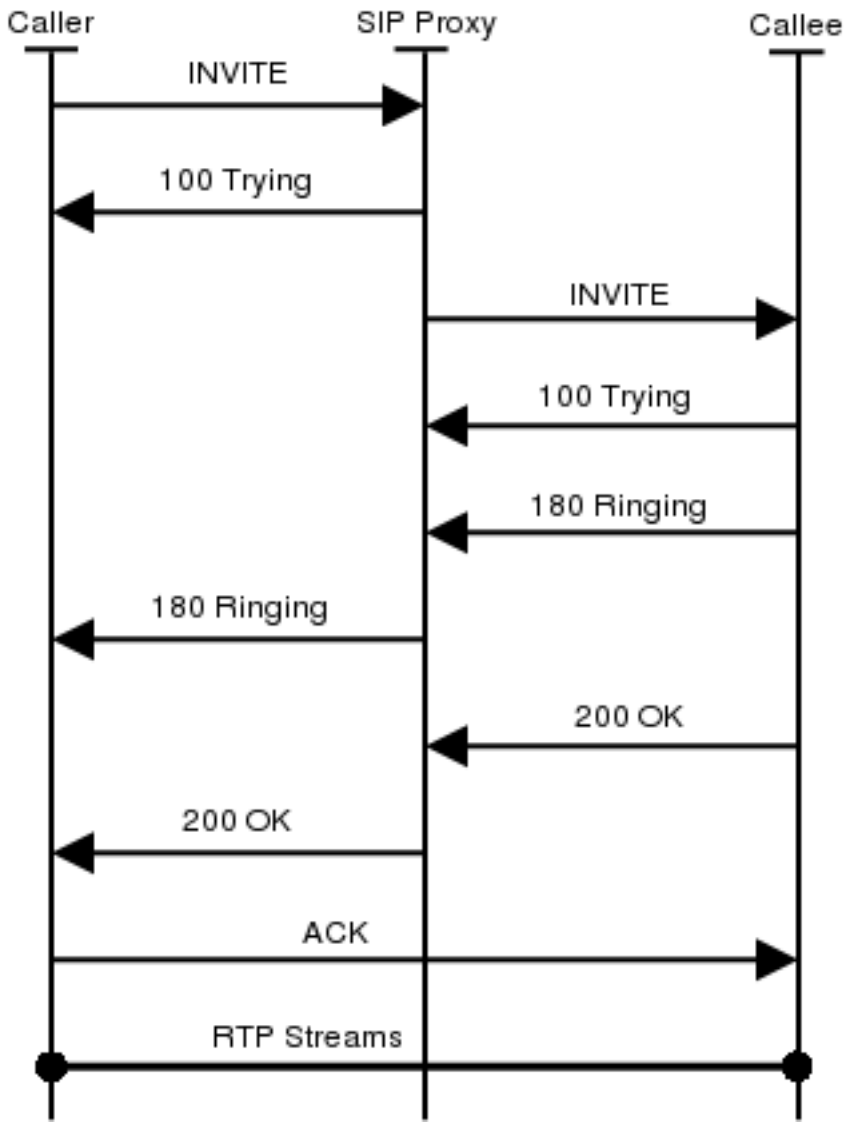


2.2.2.6.2. Session Invitation

A session invitation consists of one INVITE request which is usually sent to a proxy. The proxy sends immediately a 100 Trying reply to stop re-transmissions and forwards the request further.

All provisional responses generated by callee are sent back to the caller. See 180 Ringing response in the call flow. The response is generated when callee's phone starts ringing.

Figure 2.19. INVITE Message Flow



A 200 OK is generated once the callee picks up the phone and it is retransmitted by the callee's user agent until it receives an ACK from the caller. The session is established at this point.

2.2.2.6.3. Session Termination

Session termination is accomplished by sending a BYE request within dialog established bye INVITE. BYE messages are sent directly from one user agent to the other unless a proxy on the path of the INVITE request indicated that it wishes to stay on the path by using record routing (see Section 2.2.2.6.4.

Party wishing to tear down a session sends a BYE request to the other party involved in the session. The other party sends a 200 OK response to confirm the BYE and the session is terminated. See Figure 2.20, left message flow.

2.2.2.6.4. Record Routing

All requests sent within a dialog are by default sent directly from one user agent to the other. Only requests outside a dialog traverse SIP proxies. This approach makes SIP network more scalable because only a small number of SIP messages hit the proxies.

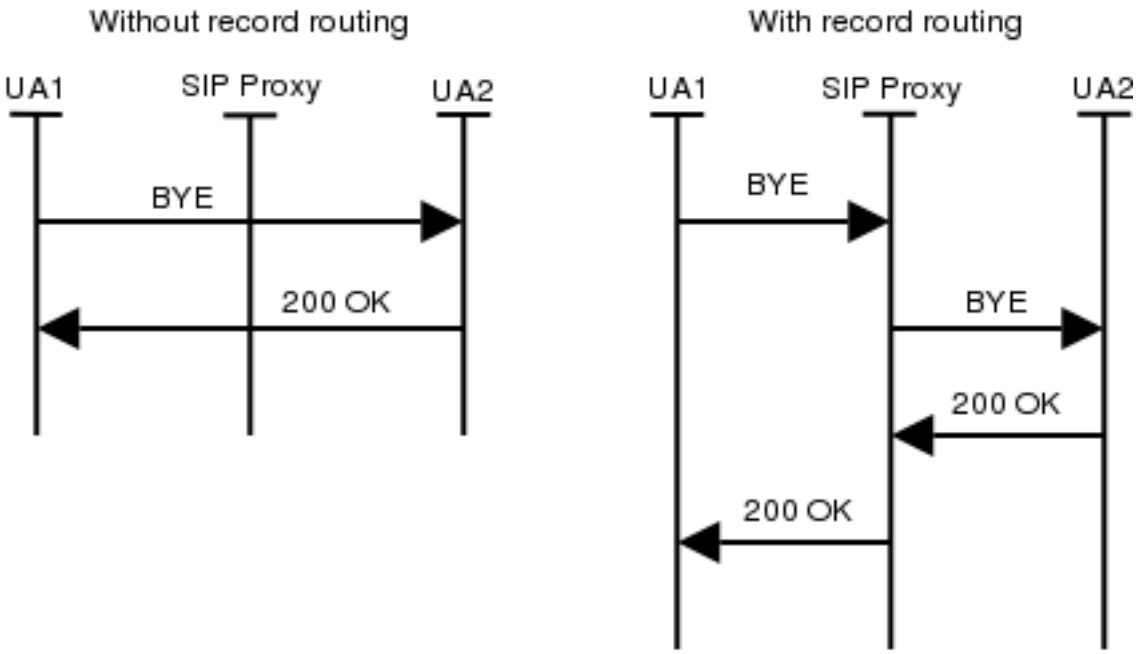
There are certain situations in which a SIP proxy need to stay on the path of all further messages. For instance, proxies controlling a NAT box or proxies doing accounting need to stay on the path of BYE requests.

Mechanism by which a proxy can inform user agents that it wishes to stay on the path of all further messages is

called *record routing*. Such a proxy would insert Record-Route header field into SIP messages which contain address of the proxy. Messages sent within a dialog will then traverse all SIP proxies that put a Record-Route header field into the message.

The recipient of the request receives a set of Record-Route header fields in the message. It must mirror all the Record-Route header fields into responses because the originator of the request also needs to know the set of proxies.

Figure 2.20. BYE Message Flow (With and without Record Routing)



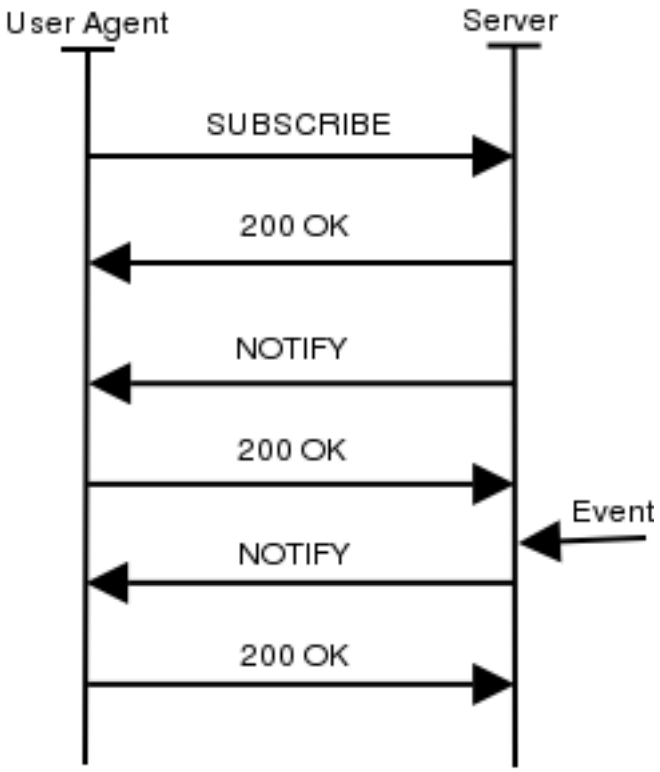
Left message flow of Figure 2.20 show how a BYE (request within dialog established by INVITE) is sent directly to the other user agent when there is no Record-Route header field in the message. Right message flow show how the situation changes when the proxy puts a Record-Route header field into the message.

2.2.2.6.5. Event Subscription And Notification

The SIP specification has been extended to support a general mechanism allowing subscription to asynchronous events. Such evens can include SIP proxy statistics changes, presence information, session changes and so on.

The mechanism is used mainly to convey information on presence (willingness to communicate) of users. Figure 2.21 show the basic message flow.

Figure 2.21. Event Subscription And Notification



A user agent interested in event notification sends a SUBSCRIBE message to a SIP server. The SUBSCRIBE message establishes a dialog and is immediately replied by the server using 200 OK response. At this point the dialog is established. The server sends a NOTIFY request to the user every time the event to which the user subscribed changes. NOTIFY messages are sent within the dialog established by the SUBSCRIBE.

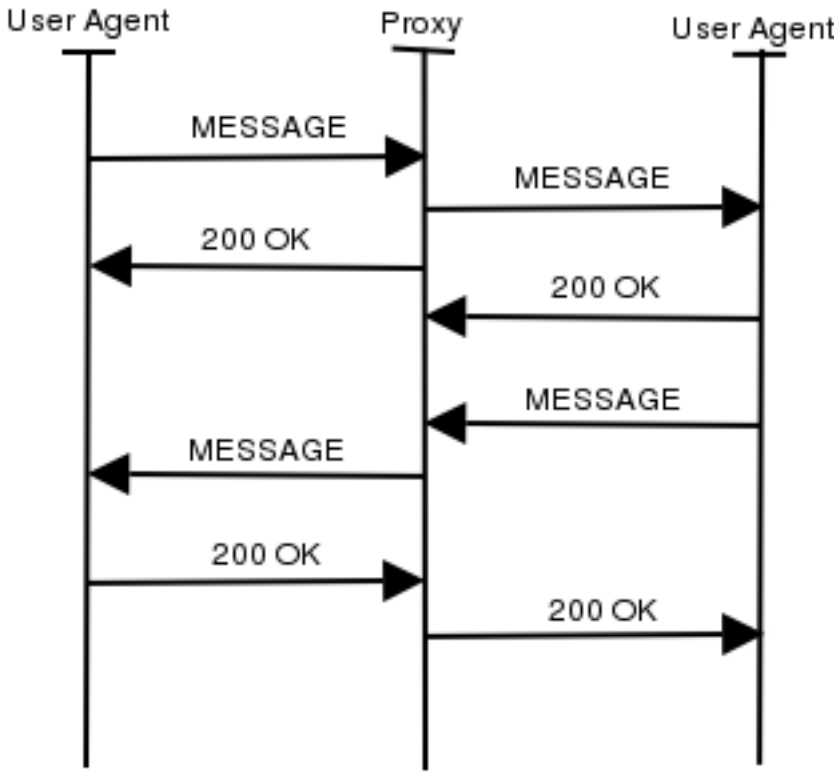
Note that the first NOTIFY message in Figure 2.21 is sent regardless of any event that triggers notifications.

Subscriptions--as well as registrations--have limited life span and therefore must be periodically refreshed.

2.2.2.6.6. Instant Messages

Instant messages are sent using MESSAGE request. MESSAGE requests do not establish a dialog and therefore they will always traverse the same set of proxies. This is the simplest form of sending instant messages. The text of the instant message is transported in the body of the SIP request.

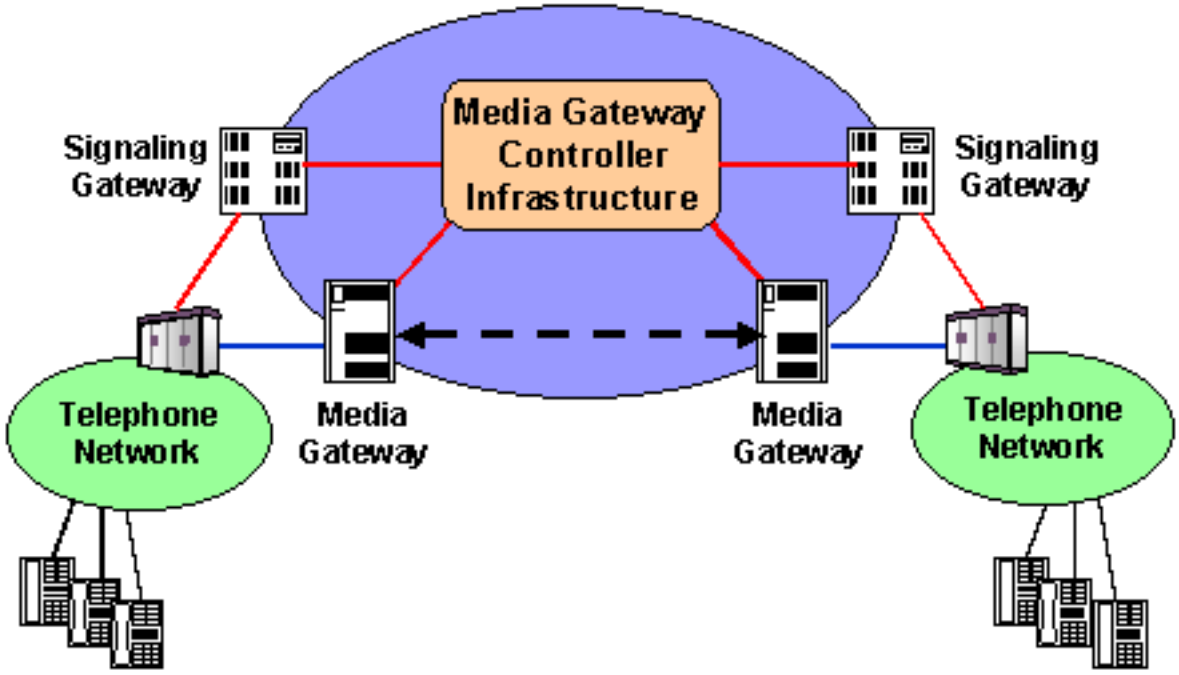
Figure 2.22. Instant Messages



2.2.3. Media Gateway Control Protocols

In a traditional telephone network, the infrastructure consists of large telephone switches which interconnect with each other to create the backbone network and which also connect to customer premise equipment (PBXs, telephones). While the internal network today is based upon digital communication, links to customers may be either analog (PSTN) or digital (ISDN). The links to customers are shared between call signaling (for dialing, invocation of supplementary services, etc.) and carriage of voice/data; in the backbone, dedicated (virtual) links interconnecting switches are reserved for call signaling (de-facto creating a dedicated network of its own) whereas voice/data traffic is carried on separate links. The Signaling System No. 7 (SS7) or variants of it are used as the call signaling protocol between switches; this protocol is used to route voice/data channels across the backbone network by instructing each switch on the way which incoming "line" is to be forwarded to which outgoing "line" and which other processing (such as simple voice compression, in-band signaling detection to customer premise equipment, etc.) is to be applied. Voice/data channels themselves are plain bit pipes identified by roughly a trunk and line identifier at each switch.

Figure 2.23. Application Scenario for Media Gateway Control Protocols



A similar construction is now considered by a number of telcos for IP-based backbone networks that may successively replace parts of their overall switched network infrastructure, as depicted in figure 3.7. Instead of voice switches, IP routers are used to build up a backbone network - which employs IP routing, possibly MPLS, and, most likely, some explicit form of QoS support to carry voice and data packets from any point in the network to any other. In contrast to voice switches, this does not require explicit configuration of the individual routers per voice connection; rather, only the entry and exit points need to be configured with each others' addresses so that they know where to send their voice/data packets to. Two types of gateways are used at the edges of the IP network to connect to the conventional telephone network: signaling gateways to convert SS7 signaling into IP-based call control (which may make use of H.323 or SIP or simply provide a transport to carry SS7 signaling in IP packets [SIGTRAN]) and media gateways that perform voice transcoding.1 Some central entity (actually, probably a number of co-operating entities) forms the intelligent core of the backbone: the Media Gateway Controller(s). They interpret call signaling and decide how to route calls, provide supplementary services, etc. Having decided on how a call is to be established, they inform the (largely passive and "dumb") media gateways at the edges (ingress and egress gateways) how and where to transmit the voice packets. The Media Gateway Controllers also re-configure the Gateways in case of any changes in the call, invocation of supplementary services, etc. The media gateways may be capable of detecting invocation of control features in the media channel (e.g. through DTMF tones) and notify the Media Gateway Controller(s) which then initiate the appropriate actions.

A number of protocols have been defined for communication between Media Gateway Controllers and media gateways: initial versions were developed by multiple camps, some of which merged to create the Media Gateway Control Protocol (MGCP), the only one of the proprietary protocols that is documented as an Informational RFC (RFC 2705). An effort was launched to make the two remaining camps cooperate and develop a single protocol to be standardized which resulted in work groups in the ITU-T (rooted in Study Group 16, Q.14) and in the IETF (Media Gateway Control, MEGACO WG). The protocol being jointly developed is referred to as H.248 in the ITU-T and as MEGACO in the IETF.

It turned out that due to largely political issues between the two camps and "religious" differences between the IETF and the ITU-T (use of text vs. binary encoding) the work progressed slowly with much time being spent discussing procedural aspects and sorting out minor technical work. Nevertheless, progress was made over time and improvements have been achieved over the individual input protocols, and it is expected that this work will come to closure some time during the year 2000. In the meantime, various stages of the proprietary protocols are being deployed, with vendors probably providing migration paths to the finally standardized H.248/MEGACO.

One particular protocol extension currently discussed in the IETF is the definition of a protocol for communication with an IP telephone at the customer premises that fits seamlessly with the Media Gateway Control architecture. Such a telephone would be a rather simple entity essentially capable of transmitting and receiving events

and reacting to them while the call services are provided directly by the network infrastructure.

2.2.4. Proprietary Signaling Protocols

Today nearly every vendor that offers VoIP products uses his own VoIP protocol - e.g. Cisco's Skinny or Siemens's CorNet. They were invented by the vendors to be able to provide more or specific supplementary services in the Voice over IP world to give customers all the features they already know from their classic PBX. The enterprise solutions usually feature such a proprietary protocol and provide simple support for a standardized protocol (until now usually H.323) with only basic call functionality.

Giving detailed information about those protocols is out of the scope of this document - and usually difficult to provide for most protocols aren't public available.

2.2.5. Real Time Protocol (RTP) and Real Time Control Protocol (RTCP)

RTP and RTCP are the transport protocol used for IP telephony. Both of them were defined in RFC1889[11]; the former as a protocol to carry data that has real-time properties, the latter to monitor the quality of service and to convey information about the participants in on-going session. The services provided by the RTP protocol are:

- identification of the carried information (audio and video codecs);
- checking packet in-order delivery and, if necessary, re-ordering the out-of-sequence blocks;
- transport of the coder/decoder synchronization information;
- monitoring of the information delivery.

The RTP protocol uses the underlying User Datagram Protocol (UDP) to manage multiple connections between two entities and to check for data integrity (checksum). An important point to stress is that RTP neither provides any mean to have a guaranteed QoS nor assumes the underlying network delivers ordered packets.

The RTCP protocol uses the same protocols as RTP to periodically send control packets to all session participants. Every RTP channel using port number N has its own RTCP protocol channel with port number equal to N+1. The services provided by the RTCP are:

- giving a feedback on the data quality distribution, feedback used to keep control of the active codecs;
- transporting a constant identifier for the RTP source (CNAME), used by the receiver to link a SSRC identifier and its source to synchronize audio and video data;
- advertising the number of session participants, number used to adjust the RTCP data transmission rate;
- carrying session control information, used to identify the session participants.

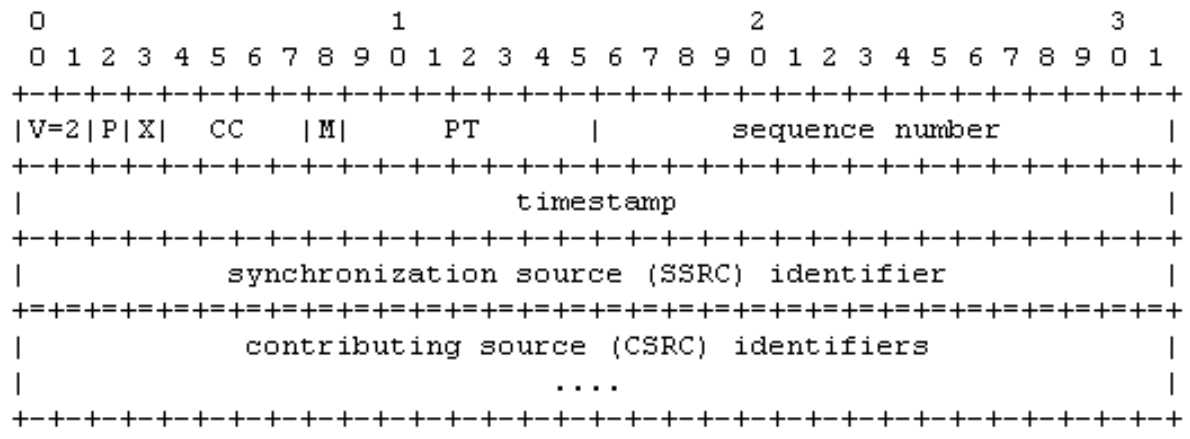
In the next two subsection we are going to describe the RTP and RTCP header and the different types of packet those two protocol use.

2.2.5.1. RTP Header

Figure 2.24 shows the RTP header. The first 12 bytes are present in all the RTP packets, the last bytes containing the CSRC (Contributing SouRCe) identifiers list is present only when a mixer is crossed (with mixer we refer to a system which receives two or more RTP flows, combines them and forwards the resulting flow).

Figure 2.24. RTP Header

[11] http://www.ietf.org/rfc/rfc1889.txt



The header fields are here detailed:

- version (V - 2 bits), contains the RTP protocol version;
- padding (P - 1 bit), if set to 1 then the packet contains one or more additional bytes after the data field;
- extension (X - 1 bit), if set to 1 then the header is followed by an extension;
- CSRC count (CC - 4 bits), contains the CSRC identifiers number which follow the header;
- marker (M - 1 bit), application available field;
- payload type (PT - 7 bits), identifies the data field format of the RTP packet and determines its interpretation by the application;
- sequence number (16 bits), value incremented by one for each RTP packet sent, used by the receiver to detect losses and to determine the right sequence;
- RTP timestamp (32 bits), is the sampling time of the first RTP byte, used for synchronization and jitter calculation;
- SSRC ID (32 bits), identifies the synchronization source, chosen randomly within a RTP session;
- CSRC ID list (from 0 to 15*32 bits), optional field identifying the sources which contribute to the data in the packet, the number of the CSRC IDs is written in the CSRC count field.

2.2.5.2. RTCP packet types and format

In order to transport the session control information, the RTCP foresees a number of packet types:

- SR, Sender Report, to carry the information sent by the transmitters, to give notice to the other participants on the control information they should receive (number of bytes, number of packets, etc.);
- RR, Receiver Report, to carry the statistics of the session participants which are not active transmitters;
- SDES, Source DESscription, to carry the session description (including the CNAME identifier);
- BYE, to notify the intention of leaving the session;
- AAP, to carry application specific functions, used by experimental use of new applications.

Every RTCP packet begins with a fixed part similar to the one of the RTP ones, such part is then followed by

structural element of variable length. More than one RTCP packet may be linked together to build a COMPOUND PACKET. Moreover, in order to maximize the statistics resolution, the SR and the RR packet types are to be sent more often than the other packet types.

Chapter 3. IP Telephony Scenarios

3.1. Introduction

This chapter describes, using the technology background terminology defined in the previous chapter, the most interesting scenarios the user may address when building an IP telephony infrastructure. The scenarios here detailed address topics with increasing complexity (from a long-distance least cost routing scenario to a fully integrated IP telephony - videoconferencing one). The aim of this chapter is to provide the user with both a high level overview of each scenario and with motivations on why they are to be deployed. Each scenario is analyzed from the user needs point of view and is described architecturally giving an example of integration.

3.2. Scenario 1: Long-distance least cost routing

This scenario is likely to be adopted by enterprises with high-cost call volumes. Since traditionally there is a separation of transport of data and voice between two locations (see Figure 3.1), to reduce costs, accounts with a lower-cost long-distance carrier are established. Voice over IP offers a second solution for this kind of problems: existing enterprises' data networks (using IP protocol) may be used to carry long distance voice traffic to certain destinations thus lowering the total costs (see Figure 3.2). A combination of lower-cost long-distance carrier and Voice over IP voice-data integration is foreseen as the most cost-effective solution in this area. This solution requires to route calls to the lowest-cost network depending on time of day and destination and it is referred to as Least Cost Routing (LCR). In order to achieve the bigger savings, it routes calls to destinations by re-dialing them through the lowest cost alternative carrier / terminator of the moment. A basic scenario's architecture (depicted in Figure 3.3) shall be able to handle all calls originated from the enterprise network. Elements needed to deploy this scenario are: terminals (both IP and PSTN) and Gateways (necessary in order to route call from the IP network to the ISDN/PSTN and vice versa); other elements like MCU or servers may be optionally present but are not required to.

Figure 3.1. Traditional separation of data and telephony between locations

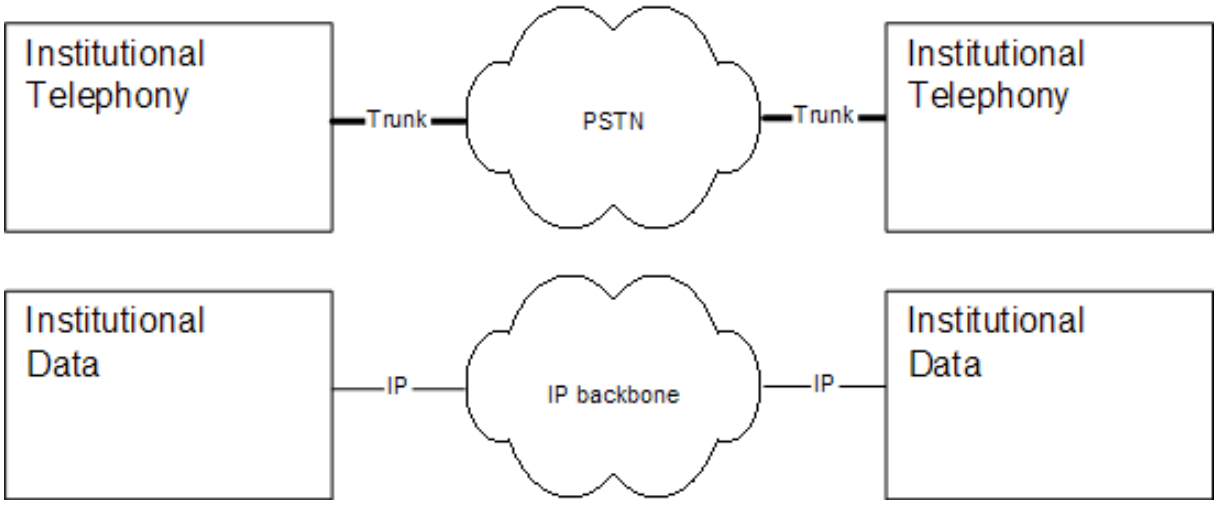
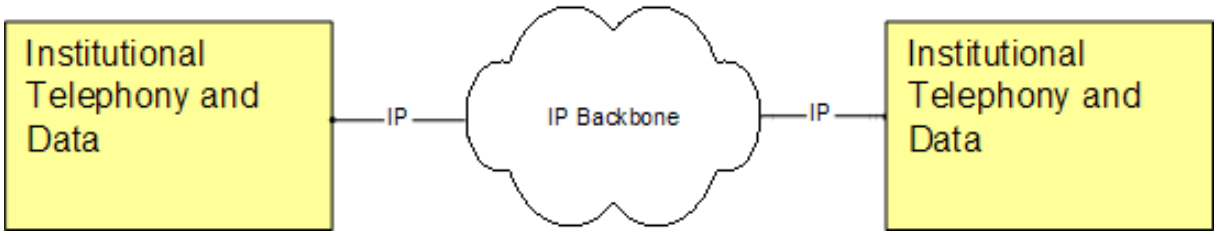
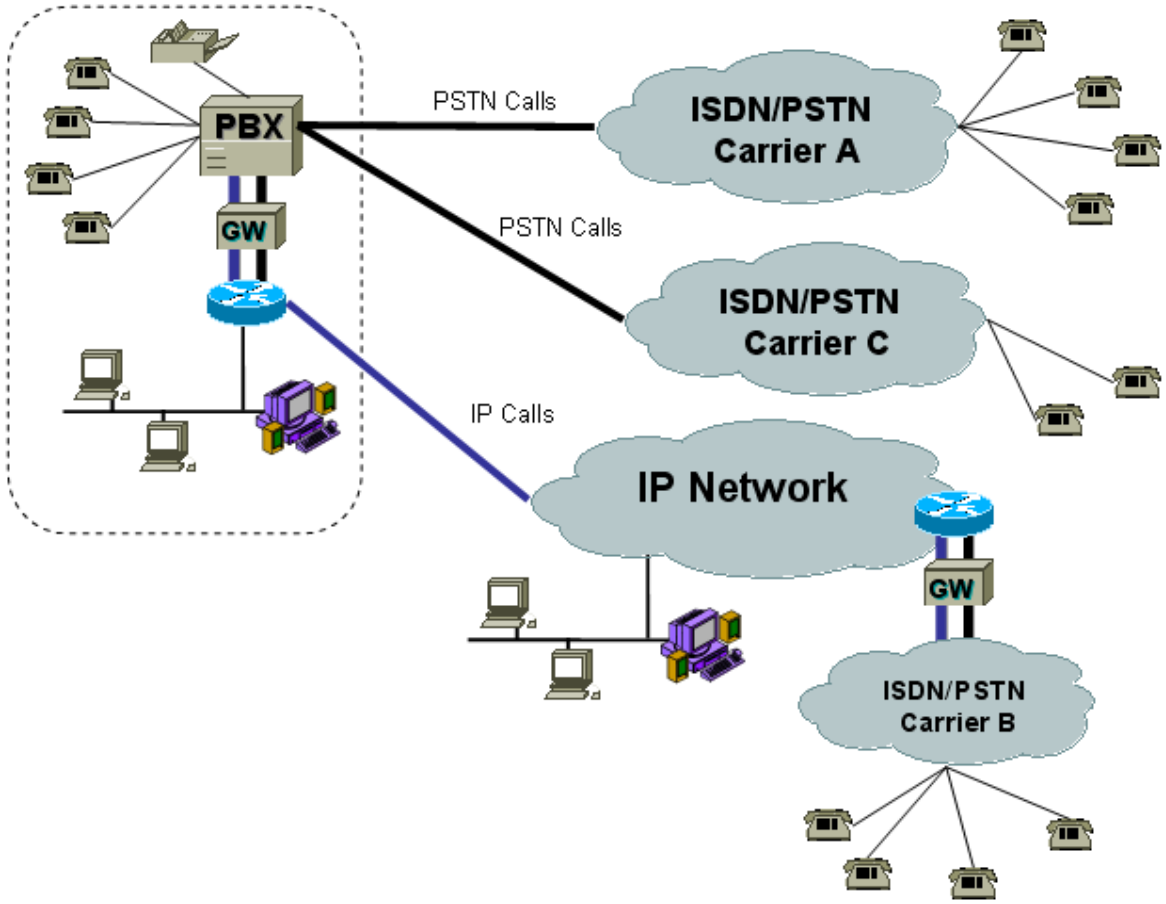


Figure 3.2. Integration of data and telephony between locations



The hybrid situation including both traditional processing of calls over PSTN/ISDN and an added IP Telephony part results in this detailed architecture:

Figure 3.3. Least cost routing architecture



The features such an architecture may provide can be basically resumed in:

- Call routing by time of day and day of the week, allowing to select the best rates applying to certain hours in the day.
- Call routing by destination, allowing to select the best rates depending on the destination of the call.
- Number modification, allowing to add or subtract digits from the original number dialed to facilitate prefix-based routing.
- Class of Service management, allowing to manage individual extensions with differentiated class of service to give that higher level of service to users who need it.

3.2.1. Least Cost Routing - An example of integration

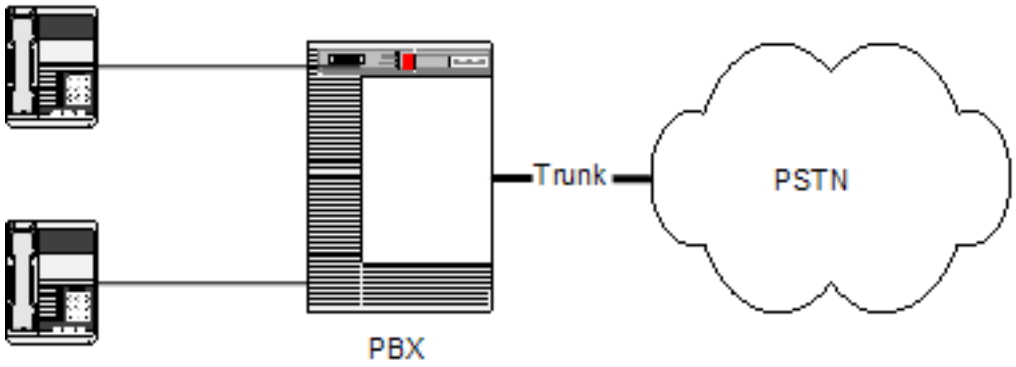
A company with one head-quarter and more than one branch-office site in Europe makes daily long distance calls to contact customers located all over the world. Since many telephone carriers provide cheap telephone rates depending on geographical areas, the competitive telephone market may be used to reduce communication costs. A first solution would require to keep an up-to-date table based on the savings depending on the time of the day and destination. The problem arising with this solution in the maintenance and distribution to the employees of this table are evident. Moreover, provided that not every employee remembers to dial the extra digits for the leading code both because of time-consuming reasons and because of negligence, an engineering process is needed to keep the cost low. Least Cost Routing is the solution to this kind of problems, it allows the telephone system to automatically route the long distance call to the most economical telephony carrier / network, saving money on the long distance bill and reduces the employee's effort in making the calls.

In order to put in place such solution the company needs to deploy a set of Gateways in the locations where a branch office is located, to take advantage of the integration of data and telephony between locations as depicted in Figure 3.2. This in order to save both on calls located in the area of one of the branch offices and on office-to-office calls using, when possible, the data network connecting the company's sites. Moreover a distributed routing table has to be implemented in order to facilitate the control by the system administrator which may update it anytime whenever changes in long distance rates occur.

3.3. Scenario 2: Legacy PBX replacement

Traditionally, institutions and companies are equipped with a PBX on (each of) its premises. Telephones are wired to the PBX which supplies them with power. The PBX handles all intelligence and routes calls to the PSTN over trunks (E1, T1, J1, ISDN30 etc).

Figure 3.4. Legacy PBX which trunks to the PSTN



One of the most economically feasible deployments of IP Telephony currently is in the area of installing voice over IP as a replacement of inter-building PSTN connections within one company, or even the complete replacement of the PBX phone system (and terminals) itself.

This chapter first describes the scenario's in which IP phones can be deployed in a peer-to-peer fashion without additional control entities in the network. This scenario is only covered briefly because the practical use is limited.

Then a common scenario will be described to introduce IP Telephony in the existing telephony infrastructure. The legacy PBX is still functional in this scenario, but the transportation of the voice calls can not only be done over regular PSTN trunks but also over IP backbones.

The last scenario describes total replacement of a PBX infrastructure by IP Telephony equipment.

3.3.1. Scenario 2a: IP-Phone without PBX

The simplest case of IP Telephony is making a point-to-point call between IP Phones. To make this succeed, the calling party needs to know the IP address of the called party, or its DNS entry.

Figure 3.5. IP-Phone to IP-Phone without PBX

For a (mission-critical) company or institutional phone system, this is an awkward method. Moreover, it is not possible to make a call to a regular telephone within the institution or to the PSTN over an VoIP-to-PSTN gateway. Also, common features like central address books, call forwarding etcetera are harder to integrate with the phone terminal. If the destination is unreachable, nothing useful can be done with the call, like redirecting it to voicemail etcetera. This setup is therefore only recommended for testing purposes.

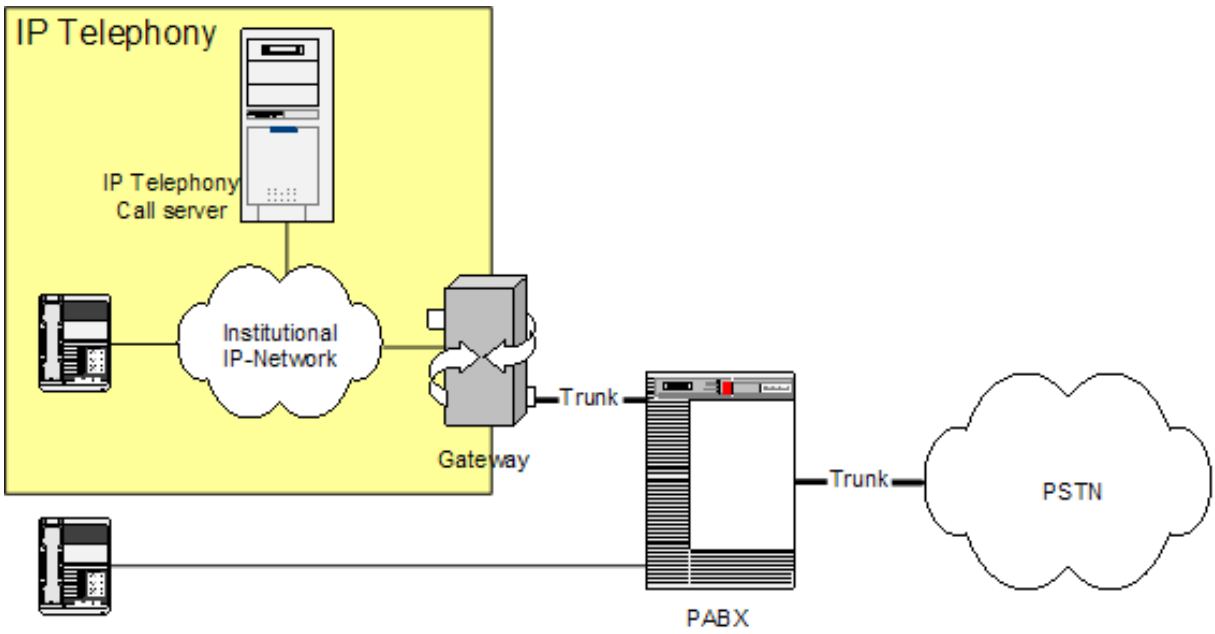
The working of a call setup is very simple, either when using H.323, SIP or any variations on these protocols. Since the calling party directly enters the IP address of the destination, the call initiation signaling is sent directly to that IP address. If the terminal is running, it will process the signaling and the called party will be prompted to pick up the phone. When that happens, the call is setup, a codec is negotiated and the voice stream will start until signaling is exchanged that terminates the call.

3.3.2. Scenario 2b: Integration with legacy PBX systems

This scenario allows coexistence and intercommunication of institutional conventional telephony network (conventional phones connected to PBX) and local IP telephony network. The scenario is suitable when the local IP telephony network is constructed gradually in an institution that already has a conventional telephony network. In a later stage, the conventional telephony network and the PBX can be totally replaced by the IP telephony network, thus converging to Scenario 2c.

In this chapter we give an overview of options for interconnecting a PBX to a Voice Gateway (VoGW). These options also apply to Scenario 1. More technical details for individual interfaces are given in Chapter 4.

Figure 3.6. Integration of IP-Telephony with legacy PBX system



The choice of particular interface between a PBX and VoGW depends on required functionality, availability of interconnection ports on both sides and also on cost constraints. As regards as interfaces, they can be divided into analog and digital. The first ones include a 2-wire U-interface with subscriber loop and various types of E&M interfaces. The latter ones include E1/CAS trunk with MFC-R2 signaling and ISDN with DSS1 or QSIG signaling. Giving technical details about the trunks and interfaces here described is out of the scope of this Chapter, refer to Chapter 4 for these details. On the other hand, interesting details for technical people who wants to understand this kind of scenario are relative to advantages and shortcomings of individual interfaces and are summarized in the following list:

- *Subscriber loop* - Suitable when conventional phones should be connected directly to VoGW (Voice GateWay) via FXS interface (an FXS interface connects directly to a standard telephone and supplies ring, voltage, and dial tone but can also be used for PBX interconnection). A disadvantage is that when calling inward PBX, an extension number can be dialed only as DTMF (Dual-Tone Multifrequency) suffix after a call was established and is already accounted. Usually a low cost solution.

- *E&M interfaces* - E&M is commonly explained as Ear and Mouth or recEive and transMit. Allows extension dialing before the conversation begins. Requires a special interface card for PBX, but if PBX is already equipped with this card, this can also be a low-cost solution.

- *E1/CAS trunk with MFC-R2 signaling* - CAS (Channel Associated Signaling) exists in many variants that operate over analog and digital interfaces. A common digital interface used is E1 (European version). The advantage of a digital interface is its ability to transfer the identification of the calling party, which is important for detailed accounting. This is the first digital solution that was used, which was later mostly replaced by ISDN interfaces. Requires special interface cards on both sides of interconnection, it is rather expensive solution.

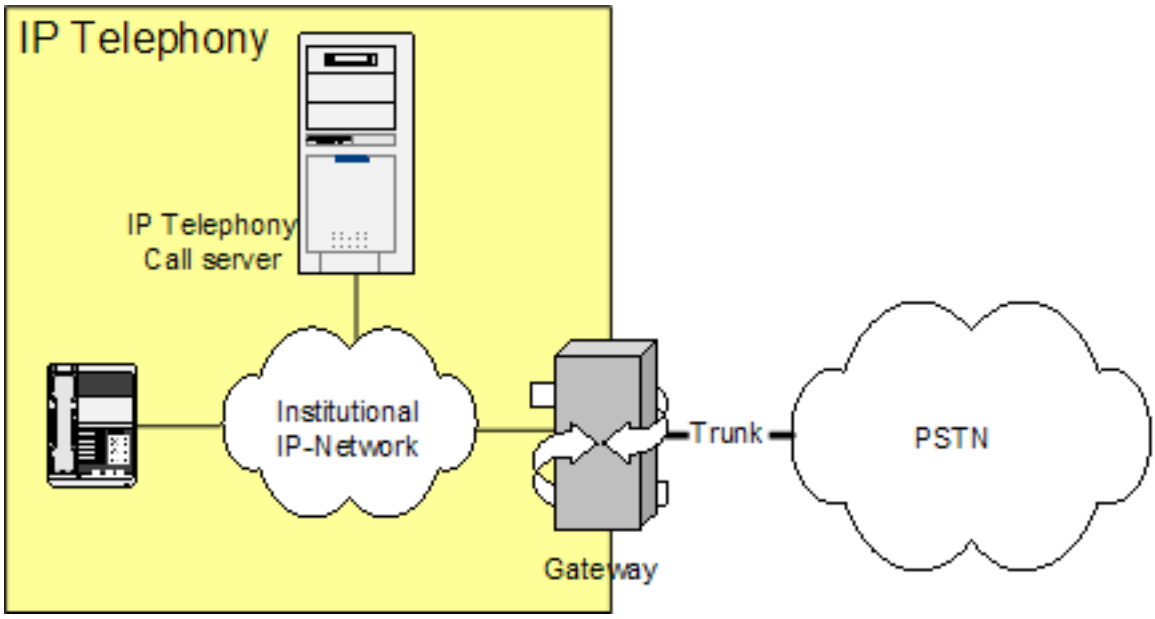
- *ISDN with DSS1 signaling* - In addition to calling party identification, supplementary services are available such as Call Waiting, Do not disturb, etc. Can be used with a BRI interface (Basic Rate Interface, up to 2 simultaneous calls) or PRI interface (Primary Rate Interface, up to 30 simultaneous calls). The interface card is usually already in place on modern PBXs. The PRI interface is economically preferable when more than 8 channels (4xBRI) are required.

- *ISDN with QSIG signaling* - QSIG signaling supports more supplementary services, such as Call completion, Path replacements, etc. However, QSIG uses proprietary features of PBX from particular manufacturers and is therefore suitable only for corporate networks, where IP telephony is used to interconnect PBXs in company branches.

3.3.3. Scenario 2c: Full replacement

In greenfield situations or when an existing PBX is fully depreciated, the time has come to consider IP Telephony as an alternative to a traditional PBX.

Figure 3.7. IP-Telephony fully replacing PBX



The design of a IPBX system includes a couple of choices.

3.3.3.1. Intelligent vs simple terminals

IPBXes can support terminals in two ways: either they have analogue ports that support analogue terminals, or they are IP only. The latter means that the terminals are intelligent devices, including an implementation of many signaling functions. Since so much intelligence is included anyway, these terminals are often equipped with interactive screens and other sophisticated functions. This makes the equipment expensive and requires the terminals to be provided with power, which can be supplied with Power over Ethernet. A feature that most of these advanced terminals support is pass-through of Ethernet, so that one wall outlet can be used for both IP Telephony as well as a computer.

3.3.3.2. Signalling

Though the choice between H.323, SIP or a proprietary system seems a purely technical one, it has implications for the interoperability with future expansions, inter-department trunking and the development of new advanced features like messaging etcetera. It is wise to have a supplier comply to at least one of the open standards.

3.3.3.3. Inter-department trunking

The choice for a full IP based institutional voice network does not automatically lead to a choice for the way that geographically separated locations are connected. The cookbook supports this choice in the Least Cost Routing section.

The inter-department architecture also includes a choice whether to break out local calls at local PSTN trunks, or to centralize all PSTN trunking on one of the locations of the institution. This choice depends on the tariff structure that the public operator(s) offer for centralized break out as well as the amount of calls that have a local public destination compared to long-distance public and private destined calls.

3.3.3.4. Legacy functionality

Traditional PBXes have the advantage that some legacy functions have been incorporated in the past. They need to be included in the IP Telephony architecture as well. The most elementary are:

- Emergency call handling to public emergency numbers (911, 112 etc)
- Public dialplan routing (regular numbers, blocking/routing of premiumnumbers etc)
- Integration with public wireless telephony
- Voice/data integration and call distribution for call center/help desk department
- Support for beeper systems

- Support for private wireless telephony
- Support for elevator phones

3.3.3.5. Wireless VoIP

With three manufacturers currently presenting wireless IP terminals that can use IEEE 802.11b (Wi-Fi) wireless data communication, a new aspect for VoIP is emerging. Where DECT has a strong position in the wireless PBX market, it can be expected that institutions that have WIFI networks in place want to reuse this infrastructure for their wireless telephony network, obtaining similar consolidation advantages as in the fixed IP telephony case. Wireless IP phones are equally intelligent as their fixed equivalents, but are different on the Ethernet level. The usual issues in wireless data communications hold in their case: low battery usage, portability, coverage etcetera. Developments show that manufacturers of public network mobile phones like GSM are planning to include Wireless VoIP into their terminals. This enables seamless roaming from public wireless telephony networks to the campus environment, potentially reducing costs when calling on campus.

3.3.3.6. Issues

Since the field of IPBX is rapidly emerging, many features that are known in the traditional PBXes are quickly adopted. New issues arise when data networks develop. An example is the introduction of network access control by IEEE 802.1X. This standard forces equipment that wants to send Ethernet frames to a network to first authenticate at a RADIUS server. All equipment on 802.1X enabled network ports should be 802.1X enabled as well. With the adoption of 802.1X, terminals are announced to include the standard as well.

A similar situation holds for VLANs. In case a network administrator chooses to put IP telephones in a different VLAN compared to PC (groups) and the IP telephones are in pass-through configuration, they should support VLAN trunking. This feature is arising in the market.

3.4. Scenario 3: Integration of VoIP and Videoconferencing

When referring to VoIP and IP telephony the main focus is around voice services, such a consideration may mislead users making them thinking that these kind of solution are only related to voice. IP telephony standards (see Chapter 2) do have the capabilities of signaling and initiate multimedia communications. This scenario details how voice over IP technologies / standards and videoconferencing solutions may be seamless integrated. The goal is to provide the users with a global architecture derived from IP telephony standards giving videoconferencing systems the chance of being adopted on a broad area. Videoconferencing systems have the task of making people feeling like they are all sharing the same space and communicating as they were in the same room. Successful videoconferencing is when technology is no longer noticeable. Even if the perceived quality of video and audio plays a very important role in fulfilling such a requirement, there are a number of other factors influencing the overall quality of videoconferencing that only the integration with IP telephony may provide:

- accessibility of the system, the system should be accessible on a broad area giving users the easiest way of communicating without worrying on how to join a conference or on how to find reachability information about the party to call;
- value added services, data/applications sharing and voice mail are only two example of value-added services not feasible with classic telephony systems which may improve the quality experienced by the user;
- interoperability among different technologies, the system should be transparent to different technologies in order to give the users the chance of having seamless connectivity.

In order to describe the possible integration scenario of VoIP and Videoconferencing we need to start detailing which are the possible applications related to the Videoconferencing scenario. Basic use of videoconferencing systems relates to meetings (special cases of meetings are classroom and collaboration meetings); more specific applications may be developed on top of basic ones with enhanced facilities, here we report a set of the most significant applications:

- *Telecommuting* - Telecommuting is a broad term referring to corporate employees who interact electronically with corporate resources and people. Extensions of the term refer to the individual interaction and collaboration that takes place between home-based consultants and inter-company business partners.

- *Telemedicine* - Videoconferencing solutions delivering high-quality video images to remote medical specialists. Specialized videoconferencing devices may be required to enable high-quality video contents not available with the standard videoconferencing systems.

- *Distance Learning* - Video lectures, remote guest speakers invitation into a classroom and private lessons group of students located in different locations are educational processes using videoconferencing tools.

- *Customer Services* - Videoconferencing-based customer services enable call centers to be more effective while interacting with customers.

- *Justice services* - Many legal systems introduces the use of videoconferencing to enable police officers to attend legal proceedings. This optimizes the time police need to spend in jails and courthouses.

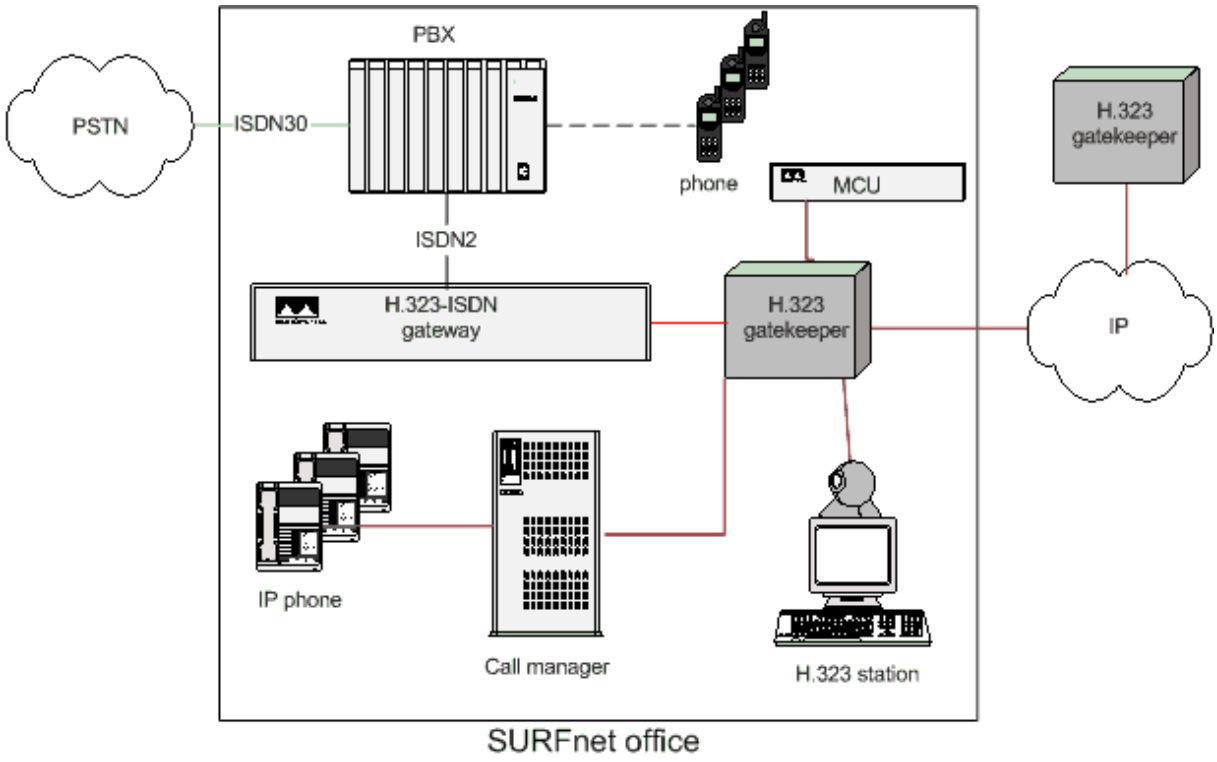
- *Virtual/Remote Laboratories* - Remote laboratories allow researchers to share advanced appliances using existing network infrastructure. As the Telemedicine application specialized devices may be required to enable high-quality video contents not available with the standard videoconferencing systems; moreover, special considerations applies when such interaction modes are considered.

A successful integration scenario would require, from the application side, application specific devices (for basic meetings maybe simple desktop conferencing equipment may be enough) to deliver the content to the end-user and, from the technical side, servers (to build a global architecture accessible by all group user), gateways (to provide interoperability with different access technologies and different IP telephony protocols), conference bridges and multipoint conferencing units (to provide multipoint capabilities for multipoint conferencing). The user advantages in deploying such advanced collaborative environments are clearly evinced if thinking at the returns in terms of time, travel costs, productivity and equipment acquisition.

3.4.1. Integrating Voice and Videoconferencing over IP - an example

In order to give the reader the chance of understanding a complex scenario as the one integrating VoIP and Videoconferencing over IP we detail here an example actually implemented at the SURFnet offices in The Netherlands. An integrated voice and video environment is setup based on H.323 technology (endpoints, MCU and gateway), cisco CallManager (using Skinny protocol), and a PBX. This therefore also is an example of scenario 2b (Integration with legacy PBX systems). Figure 3.x gives an overview of the components and how they are connected.

Figure 3.8. Integrated Voice and Video over IP architecture at SURFnet offices



The goal of the integration of voice and videoconferencing over IP was to make it possible to abstract from the device the user wants to use to communicate. When someone contacts the user by any means (PSTN or H.323 of H.320). The components necessary to establish such an integrated infrastructure are:

- a PBX, connected to the PSTN, in this case a Philips Sopho, to handle all incoming regular voice calls;
- a H.323 gateway, in this case a Radvision L2W (LAN to WAN H.323) gateway, on the one side connected to the PBX (by ISDN2 lines) and on the other side to the local IP network;
- a H.323 gatekeeper, in this case the build-in gatekeeper at the Radvision gateway, to route all calls on the IP network, including making decisions when to route the call off-net (to the PSTN through the PBX);
- a Callmanager, in this case a Cisco Callmanager, being the gatekeeper performing PBX-like functions for the IP-phones;
- endstations, being the user's terminal(s).

The terminals used here are:

- IP phones, in this case Selsius and Cisco IP phones, registered to the Callmanager;
- H.323 videoconferencing stations, in this case VCON EscortPro boards in PCs with Meetingpoint 4.6 and Polyspan Viewstations (128 and FX), registered at the H.323 gatekeeper;
- regular DECT phones, in this case Philips, registered at the PBX;

For allowing multipoint calls a MCU (Radvision MCU323) has to be added and the conference feature on the PBX to be enabled. These are not necessary functionalities, but can enhance the communication experience.

The means by which integration was established was the unique Numbering and Dial Plan. The Global Dialing System (GDS, see section on dialplans and http://www.wvn.ac.uk/support/h323address.htm) is adopted, the full E.164 addressing and number of videoconferencing and IP tel endpoints is used like they were regular phones.

Thereby it is guaranteed that even if the terminals were called/dialed from the PSTN the number would be routed to the PBX. Also the other way around, from the voice and video over IP terminals any regular PSTN number could be dialed without the need for rewriting. GDS is supported by the ViDeNet H.323 gatekeeper hierarchy, which resembles the phone system in that it is a hierarchy of distributed gatekeepers that route IP calls based on prefix resolution.

In the examples below, "A"'s phone number is 030-2305367, or international number 0031302305367. His IP phone number is 030-2305167. For demonstration purposes he also numbered his H.323 station 030-2305367. 00 is the international access code in the Netherlands, 030 is the area code of Utrecht, 23053 and 23051 are the prefixes/numberblocks SURFnet has and 67 is the local office number assigned to "A" (Note: to "A" and not his devices, because there is more than 1 numberblock that holds 67 (367 and 167).

"A" registers his H.323 station with the number 67 at SURFnet's office gatekeeper, which itself is registered with prefixes 3023051 and 3023053 at the Dutch national gatekeeper, which itself is registered with prefix 31 at the ViDeNet root gatekeepers. The gateway is registered as service at the office gatekeeper (with the prefix 5) and connected to the PBX. In the PBX is configured that all calls to 367 and 167 have to be forwarded to the gateway. In today's PBXs this is easily configurable and can often be made available as webbased service so people can maintain their own records and preferences. At the PBX also the group number (for making all telephones in a group ring) 501 for the group "A" belongs to is defined. At the gatekeeper the 500 number is configured as backup number that will be called if the H.323 call doesn't get answered. The IPphones are registered with their 1xx number (in my case 167) at the Callmanager which itself is registered as a gateway serving all these numbers (so 167, 109, 1xx etc) at the gatekeeper.

We can distinguish the following situations (not complete list of possibilities, but a couple of descriptive ones):

a) a user on the PSTN calls "A" at SURFnet, which has decided to take all calls on his H.323 station. When the call for 030-2305367 comes in at the PBX of SURFnet, the PBX looks for the terminal (telephone) 67. It recognizes that the call has to be forwarded to the gateway. When the call is routed there the H.323 gatekeeper picks it up and looks in his tables for a terminal with number 67, finds it as "A"'s H.323 station and forwards the call. The ISDN signalling is done between the PBX and the gateway and the call is setup. "A" answers the incoming call on his videoconferencing station, only getting audio since there is no video attached to the signal from the PSTN user. If "A" doesn't answer his H.323 station then the gatekeeper sees this and dials the backup number 501. The gatekeeper recognizes this as a call for the voice service at the gateway (prefix 5) and routes the call there. The gatekeeper passes it off to the PBX which make all phones in the group ring. "A" or one of his colleagues can than answer the call by picking up any of the phones in the group.

b) a user using an IP phone dials A's phone number. For this example "A" has his regular phone registered with 67 at the PBX and his H.323 station as 97 at the gatekeeper. The H.323 IP phones (or the Cisco IP phones through the Callmanager) when connected to the GDS/ViDeNet system can find each other through that hierarchy. If someone using an IP phone dials 0031302305367 then the Call manager recognizes this as an international call and forwards it to the local gatekeeper. This gatekeeper sees that it is an international call and forwards it to the ViDeNet root gatekeeper. Here the prefix 31 is recognized and the call is forwarded to the Dutch National gatekeeper. There the prefix 3023053 is recognized and the call is forwarded to the SURFnet office gatekeeper. Here the number 67 is recognized. Not having a station with 67 registered there it falls back to forwarding the call to the gateway which routes it to the PBX. Here the phone with 67 is found and the call is setup.

c) a videoconferencing station dials A's IP phone. Someone using a H.323 videoconferencing station finds "A"'s number on a card as 00312305167. He dials the number. Like in the example above, through the ViDeNet hierarchy the call gets to the SURFnet office gatekeeper who sees that the call is for 167. In its tables it finds that that number belongs to the Callmanager and routes the call there. The callmanager acts as a H.323-Skinny gateway and forwards the call to the IPphone.

Note. If "A" had also used the number 030-2305367 for his IPphone he would have had to make the choice at the gatekeeper to route all calls to the H.323 VC terminal OR to the IPphone, since there cannot be two devices registered with the same alias (E.164 no.) at the same gatekeeper.

Local dialing between the systems is supported too, "A" can call 97 from his phone to reach his H.323 station, or 167 to reach his IPphone. The other way around (from IPphone or H.323 station) he needs to use a defined prefix (5 for voice calls, see above), so 5367 will ring his normal PSTN phone.

In case the MCU was involved then people, either using a PSTN device or a H.323 IP phone or videoconferencing station would dial the routable E.164 number of the multipoint conference that is registered at the office gatekeeper as if it was a H.323 terminal.

The next step towards full integration is the introduction of a SIP proxy and SIP-H.323 gateway making it possible to extend the range of used devices even further.

Note, the example above relies on a numeric dialing plan. Alphanumeric dialing and routing is implemented differently, see section 2.xx on Addressing.

Chapter 4. Setting up basic services

Abstract

In Chapter 2 we saw how the H.323 and SIP protocols work. This chapter explains how to set up an infrastructure including H.323 and/or SIP components while given real world configuration examples for popular equipment. The focus will be on setting up basic services, meaning the equipment that are necessary to provide basic call services authentication and billing.

4.1. General concepts

Before giving examples on how to set up SIP or H.323 zones using equipment from specific vendors this section introduces the general concepts. With these concepts in mind it will be easier to understand the vendor specific parts that follow.

4.1.1. Architecture

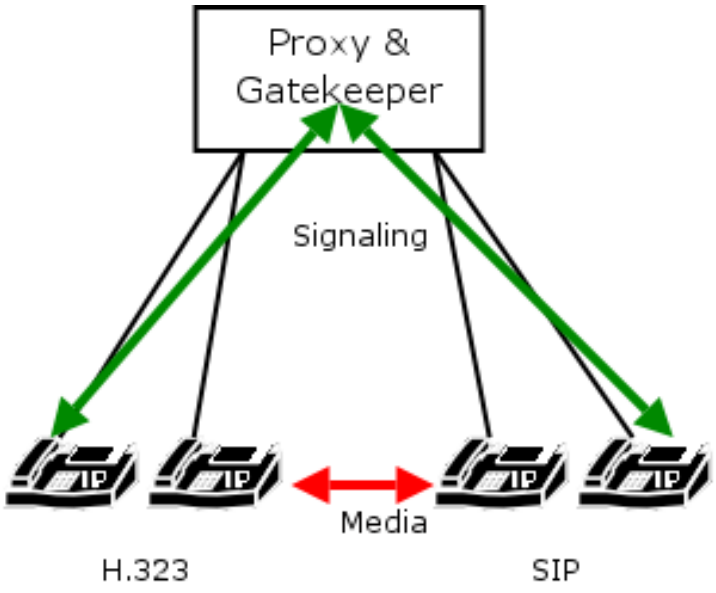
Chapter 2 introduced us to the architectures of pure H.323 and SIP environments. Basically, there is one central server and multiple telephony endpoints registering with that server. The server's task - next to others - is to resolve a dialed address to an IP target. However, when we talk about real world set-ups this server infrastructure tends to be more complicated. Reasons for this are:

- Usage of redundant servers to increase availability or provide load balancing (see Section 4.1.2)
- Usage of multiple servers - e.g. for branch offices
- More than one signaling protocol

There are two possibilities to support multiple signaling protocols within one zone: have servers with built-in support for every protocol or have dedicated servers for each protocol and signaling gateways between them.

A server that supports more than one signaling protocol (see Figure 4.1) is the best solution. It is easier to manage since there is just one configuration to take care of and there will not be any problems with server-to-server interaction.

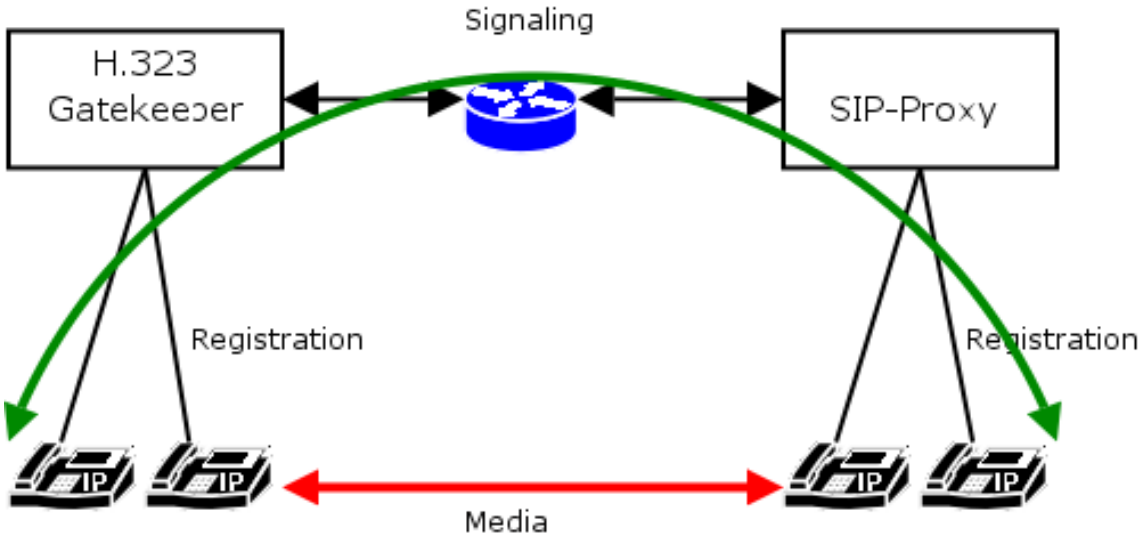
Figure 4.1. SIP/H.323 zone using a multiprotocol server



Unfortunately there seems to be no equipment that provides fully featured SIP and H.323 support on the same machine.1

If a zone includes both a SIP-proxy as well as a H.323 gatekeeper then call routing inside the domain becomes an issue. A signaling gateway is required to enable a H.323 endpoint to call a SIP endpoint and vice versa (see Figure 4.2).

Figure 4.2. SIP/H.323 zone using a signaling gateway



The gateway is used to translate signaling between the two worlds. The media stream may still be exchanged directly between the endpoints, but eventually (see Section 5.1.4.4) the gateway needs to transcode different codecs, even if both endpoints support the same codec. This problem might occur, if either the gateway or the entity calling the gateway (endpoint or gatekeeper) does not support FastConnect (Section 2.2.1.5.1).

A architectural problem similar to the last one is the usage of servers that feature a proprietary IP telephony protocol and provide a SIP or H.323 interface that is limited to basic call functionality without any supplementary services or security features. An example for this is the popular Cisco CallManager™.

4.1.1.1. PSTN gateways / PBX migration

The most common scenario for introducing IP telephony systems is to integrate them with an existing PBX. On the technical side this usually involves a gateway that translates between H.323 or SIP and QSIG or other protocols over an S2M interface. The services that can be provided between the legacy PBX and the IP world will depend on the QSIG implementation of the PBX and the Gateway vendor. There is no general advice here but to test before buying.

Beside technical aspects, organizational aspects of PBX-VoIP integration call for careful planning and analysis. The question here is how to integrate legacy and IP telephony equipment into the dialplan of an organization. Is it necessary the phone number to reflect whether the participant is a VoIP user or a PBX user?

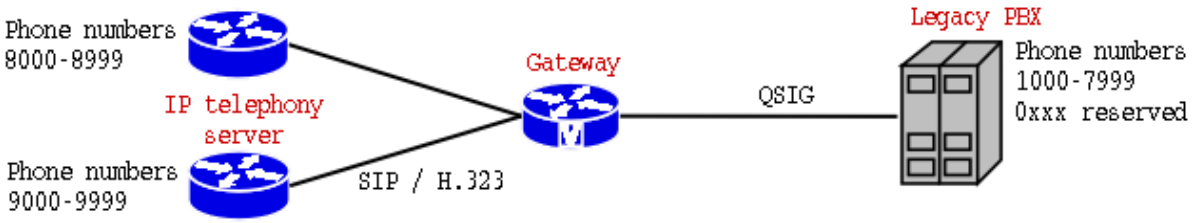
Generally, there are three possibilities to explore as explained below. You may find more details on setting up an IP telephony gateway in section Section 5.1.

4.1.1.1.1. Routing based on a number prefix

This option requires that in the dialplan there is a numberblock available for IP telephony use. It is easy to implement on legacy PBXes and IP telephony servers but results in new phone numbers for every user that switches from legacy to IP telephony.

1Actually there is a product that claims support for both SIP and H.323 protocols: the Asterisk™ PBX http://www.asterisk.org/. How well it supports SIP and H.323 is not yet known.

Figure 4.3. Routing based on number prefix

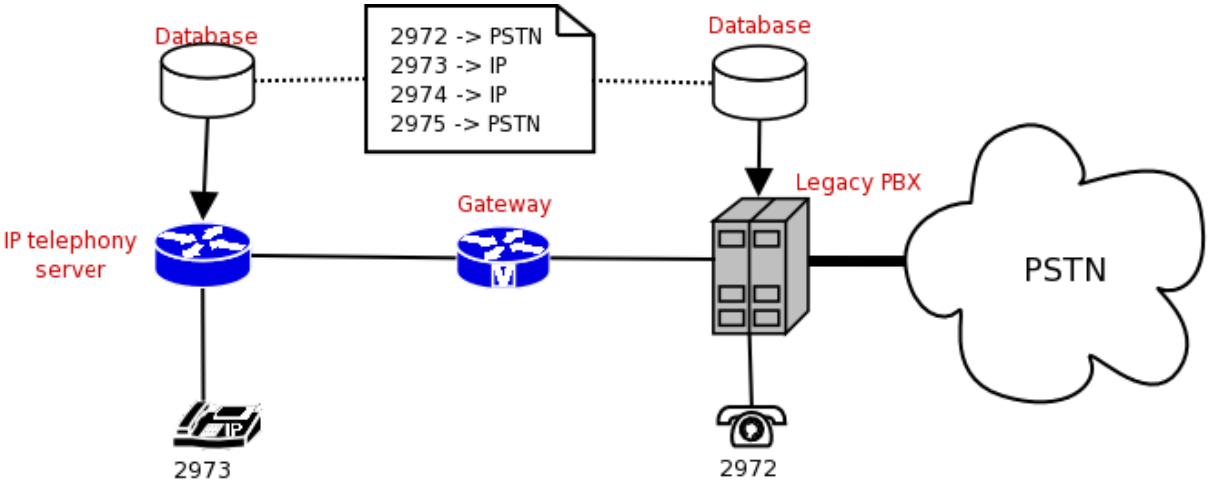


In this example the PBX is configured to forward every call starting with the prefix 8 or 9 to the gateway. The gateway passes the call to one of the IP telephony servers depending on the prefix 8 or 9. Unknown target prefixes are routed to the legacy PBX. An IP telephony server just needs to route all internal calls with unknown prefixes to the gateway.

4.1.1.1.2. Per number routing, one server

Per number routing on the PBX. When migrating from an legacy PBX towards IP telephony it is often required to provide seamless migration for users switching over from the PBX to the IP world, e.g. when a user decides to switch to IP telephony but wants to keep his telephone number.

Figure 4.4. Per number routing



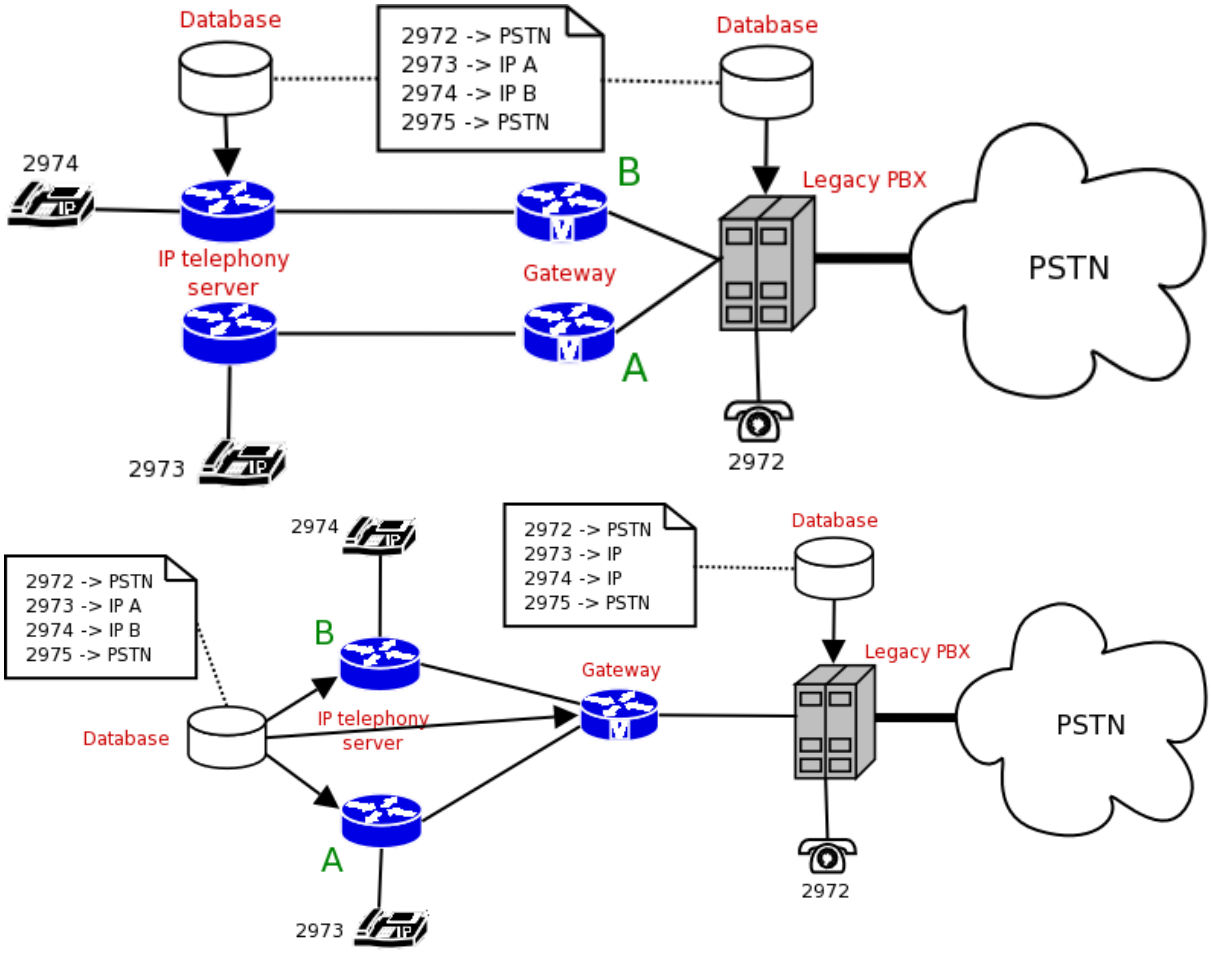
This is quite easy to achieve by setting up a database that stores the information for telephone numbers that belong to the PBX or the IP world. (See Figure 4.4). The IP telephony server and the PBX access the same data to decide where to route a call. Calls to external targets are routed to the PBX and out into the PSTN.

There are variations to this scenario. Indeed, it is quite unusual that an IP telephony server uses the same database as the PBX, unless they are from the same manufacturer. Then there are two possibilities: setting up a second database suitable for the IP telephony server (and risk inconsistencies) or let the IP telephony server route calls to unregistered targets to the PBX. The latter is easier to implement but prevents the discarding of the PBX and the use of IP telephony providers in the future.

4.1.1.1.3. Per number routing, more than one server

A similar but more complex scheme is the variant where there is more than one IP telephony server in the IP world - e.g. a server for each signaling protocol used. In this case there needs to be a database that not only contains the information about which number shall be reached on the PBX and which in the IP world, but also the information which IP server (and signaling protocol) to use.

Figure 4.5. Per number routing with a) two or b) one gateways



The first scenario uses two gateways and thus allows the PBX to decide where to route a call to. The IP telephony servers route calls they cannot resolve locally through a dedicated gateway to the PBX and let it make the routing decision. If both IP telephony servers support the same signaling protocol they may contact each other directly (see Section 4.1.1.2).

The second scenario uses only one gateway so the PBX does not need to know which IP telephony server to contact anymore but merely the fact that the call has an IP target.The components in the IP world share a common configuration database that locates a specific number on server *A* (e.g. a SIP proxy) or *B* (e.g. a H.323 gatekeeper). This allows any server or gateway component to make routing decisions.

The described scenario may vary in many ways. The assumption that all IP components use the same routing database is often difficult to achieve, especially if products from different manufacturers are used because there is no common format for such entries. In this case it might well be that a separate database is maintained for each component - leading to administrative overhead for their synchronization with a high risk of inconsistency.

4.1.1.2. Trunking

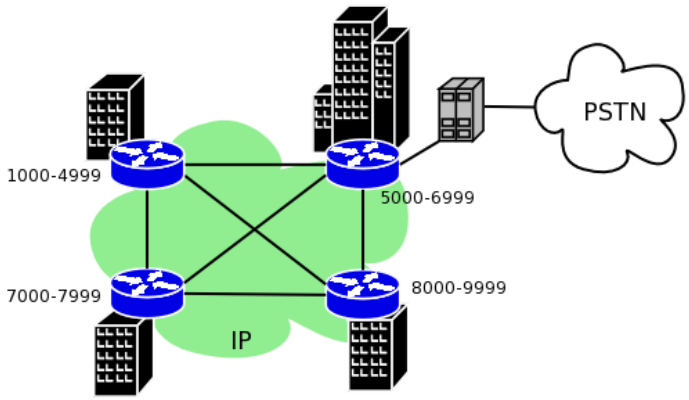
The bigger an institution is the more complex its organizational infrastructure is. This may be due to the need to support multiple locations (sites) or because certain organizational units need to administrate their own communication solutions (and eventually do not adhere the institution's standard procedures). Either way, the IP telephony system probably consists of more than one server, all with the need to share the same dialing space.

4.1.1.2.1. Prefix based

One possibility of connecting two or more networks is to give each network a different prefix and make routing

decisions based upon those prefixes. This works best if the current PSTN dialplan already provides these prefixes, e.g. in the form of area codes plus subscriber prefix.

Figure 4.6. Prefix based trunking



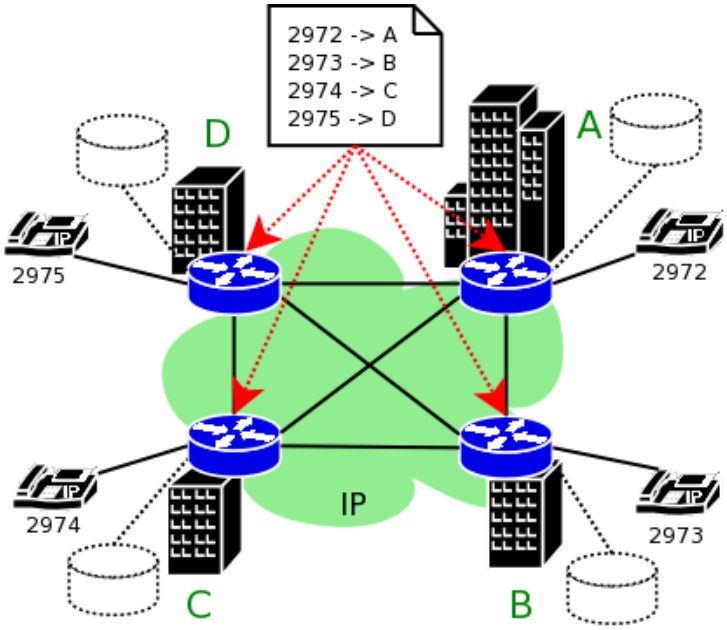
This is the classic *Branch office* scenario that is based upon the assumption that both networks have different locations and dialing prefixes already exist in the PSTN. This is the same problem as the one already addressed in Section 4.1.1.1.1.

4.1.1.2.2. Static individual routing

Prefix based routing fails to work if you have more than one IP telephony server and a PBX - all of them within the same dialing address space - and want to allow users to change their legacy PBX phone for an IP phone without switching to a new phone number.

An example of such a scenario is a university that has no structure in its PBX dialplan and that now introduces IP telephony, but has a computer science faculty that runs its own IP telephony server. How does the system know where to route a dialed number?

Figure 4.7. Static individual trunking



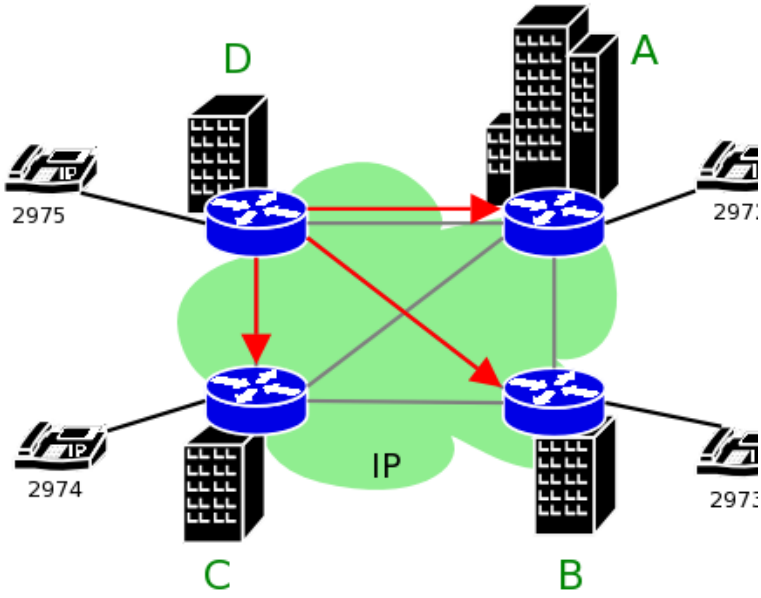
Obviously a central database storing routing information for each phone number would be a good idea (see Figure 4.7). This usually works only if all routing entities support the same kind of database - meaning that the legacy PBX and the new IP telephony server must share the same database. Such a solution is most probably only possible if the PBX and the IP telephony server come from the same vendor.

More likely is a solution where a PBX "knows" which numbers are located in the IP world and the IP telephony servers use a shared database that defines which number belongs to which server. If there are different types of IP telephony servers it may be that they are unable to share a common routing database in which case each server has its own database - resulting in administration overhead and risk of inconsistency.

4.1.1.2.3. Dynamic individual routing

The previously introduced possibilities required some kind of static configuration to bind a number to a specific server. These scenarios usually end up in using at least one routing database - but often more. The requirement for such a burden has its roots presumably in the classic PBX mentality of static configuration. In the IP world the network offers the possibility to carry information from one server to another so that IP telephony protocols provide the means to dynamically exchange routing information. The mechanisms can generally be divided into Push and Pull techniques. When using a *Push mechanism* a server informs its peers of every endpoint that registers or unregisters from the server, thus allowing its peer servers to immediately make routing decisions immediately if necessary. On the other hand, care must be taken of the fact that servers that are integrated later will be sync-ed.

Figure 4.8. Dynamic individual trunking



Pull mechanisms are used when a server asks its peer servers for an target address when it needs to resolve an external target address. The peer giving a positive reply receives the routed call (see Figure 4.8)

In H.323 trunking is achieved by LRQ messages (see Section 7.1.1. SIP has no extra mechanism for address resolution but is able to fan out a call to multiple servers at once. Both signaling protocols may use TRIP (see Section 7.1.3) to distribute numeric addresses - but not names.

4.1.2. Robustness

A highly available telephony infrastructure must deal with the fact that an IP telephony server might crash or be down for administrative reasons. Telephony services can be affected adversely in various ways, when a server goes down and another server takes over. The following are some approaches for implementing robustness in the server infrastructure.

1. The first approach is to set up more than one server for a zone and treat them as separate routers (see Section 4.1.1.2.3) that share the same configuration. In this case, there is no replication of registration or call data across multiple servers.

   If the primary server fails, a calling phone eventually will not even notice at first because the UDP media stream usually is transmitted directly between the two endpoints. But the TCP signaling connections do not survive such a crash and so the first TCP message sent afterwards leads to an error and very likely to a call clearing.

   A phone that is not currently in a call has no way of detecting a server crash in time. After its registration period expires it will try to refresh its registration with the primary server and fail. It then needs to find a new IP telephony server. H.323 provides a mechanism called “Alternate Gatekeeper” which basically defines that a gatekeeper registering an endpoint informs it of possible secondary gatekeepers that can be used alternatively. The telephone stores this information and in case of a server failure tries to contact the other listed gatekeepers.

   Another possibility that works for SIP and H.323 is to configure a prioritized list of H.323 gatekeepers or SIP proxies in a DNS SRV record for the zone. This requires that the telephone is aware of its DNS domain and is able to query DNS servers - a concept that is common in the SIP world, but seldom found in H.323 devices.

   In general, without synchronization between the replicated servers the failure of one server normally results in the loss of all calls. The server loss is discovered at least after the defined *registration timeout*, which usually is measured in minutes - but theoretically can also be set to days. After that time the phones should be able to find an alternate IP telephony server to register with.
2. Another approach is to use servers that maintain *replicated registration data* while only one of them is the active server, and the other is the standby server. If the active server fails the standby server detects this instantly and can use the replicated information about which devices are registered to inform all endpoints (phones) that it is now the new active server. As a result, the outage will be noticeable only for a few seconds - of course, active calls will still cleared, and definitely not resumed.
3. If the previous approach is pushed a bit further both servers could replicate every kind of state they keep internally, down to the connection layer. In case the active server crashes, the other system takes over and can announce (via ARP) the same MAC address as the crashed server. This kind of *"Hot Standby Server"* would take over instantly and seamlessly, allowing even ongoing calls to continue without noticeable interruption.

   In terms of server infrastructure, this is the most advanced and complicated solution a manufacturer could implement. It does not require the phones to be intelligent or support any kind of robustness mechanisms. The downside of this approach is that the ARP rewriting mechanism might not work in switched networks - which would force both servers to be in the same shared network segment.

A general advice on which kind of robustness mechanism to use is hard to give. The third solution allows the use of "dumb" endpoints because operation of the backup server is completely transparent to them, reducing the cost of endpoint equipment. The other two solutions offer the possibility to put the redundant server into different buildings - allowing the telephone system to operate even if one building burns down. A general observation is that telephones having the capability to switch servers immediately are not very common, and servers supporting Hot Standby as described above are equally hard to get.

Every manufacturer that offers IP telephony solutions implements some robustness mechanism or another. One should just be aware of the endpoint requirements that must be met to take advantage of the mechanism offered.

4.1.3. Management issues

When setting up an IP telephony infrastructure certain issues concerning administration tasks should be considered ahead of time.

4.1.3.1. Multiple account databases

Having to migrate legacy and IP telephony gives rise to the problem of maintaining multiple account databases. The legacy PBX already has a configuration that defines valid numbers. The same usually applies to an IP telephony server (see Figure 4.4). A shared configuration database for both the PBX and the IP telephony server is very uncommon, unless they are of the same manufacturer and have similar configuration interfaces. Of course, keeping two separate databases consistent is difficult on the long run.

To make this problem even worse it is possible that the gateway between the IP and the PSTN world needs access to the valid numbers as well. Potentially, this implies a third database with valid numbers - making the administration of telephony accounts (e.g. creating a new account or moving one account from legacy telephony to IP telephony) a tough job.

4.1.3.2. Decentralization

Another issue occurs regarding the question who is allowed to administrate what in the telephone system. In classic environments there is a small group of PBX administrators that sits in a special location while the network world - at least on campuses - often consists of locally administrated networks.

When introducing IP telephony there is the chance - or pitfall, depending on your point of view - to apply the structures from the network world to the telephony world. For instance consider an IP telephony infrastructure of a university that gives every student a telephony account. At the start of the semester several hundred new accounts must be created. To reduce the workload this could be delegated to administrators of the different departments. Or consider a research staff member that moves from one office to another which might require a configuration change when using port based authentication. It would be fine if this changes might be done decentralized.

Decentralized administration does not necessarily mean that all administrators have the same permissions. An IP telephony server might for example separate the permission to change account data from the permissions to view the call detail records.

Many available products allow remote administration through a web interface which generally allows decentralized administration. If different administration permissions can be granted highly depends on the products used.

4.1.4. Operation of SIP Servers

4.1.4.1. Recommended Operational Practices

Operation of a SIP server is not always an easy task. Server administrators face many challenges of broken or misconfigured user agents, network and host failures, hostile attacks and other stress-makers. All such situations may lead to an operational failure. It is sometimes very difficult to figure out the root reason of a failure, particularly in a distributed environment with many SIP components involved. In this section, we share some of our practices and refer to tools which have proven to make the life of administrators easier.

4.1.4.
1.1. Message logging.

Frequently, operational errors are discovered or reported with a delay. Users frustrated by an error frequently approach administrators and scream "even though my SIP requests were absolutely OK yesterday, they were mistakenly denied by your server". If administrators do not record all SIP traffic at their site, they will not be able to identify the reason of the problem. We thus recommend that site operators record all messages passing their site and keep them stored for some period of time. They may use utilities such as ngrep or tcpdump .

4.1.4.
1.2. Real-time Traffic Watching.

Looking at SIP messages in real-time may help to gain understanding of problems. Though there are commercial tools available, using a simple, text-oriented tool such as ngrep is sufficient for the job, thanks to SIP's textual nature.

Example 4.1. Using ngrep

In this example, all messages at port 5060 which include the string “bkraegelin” are captured and displayed.

```
[jiri@fox s]$ ngrep bkraegelin@ port 5060
interface: eth0 (195.37.77.96/255.255.255.240)
filter: ip and ( port 5060 )
match: bkraegelin@
#
U +0.000000 153.96.14.162:50240 -> 195.37.77.101:5060
REGISTER sip:iptel.org SIP/2.0.
Via: SIP/2.0/UDP 153.96.14.162:5060.
From: sip:bkraegelin@iptel.org.
To: sip:bkraegelin@iptel.org.
Call-ID: 0009b7aa-1249b554-6407d246-72d2450a@153.96.14.162.
Date: Thu, 26 Sep 2002 22:03:55 GMT.
CSeq: 101 REGISTER.
Expires: 10.
Content-Length: 0.
.

#
U +0.000406 195.37.77.101:5060 -> 153.96.14.162:5060
SIP/2.0 401 Unauthorized.
Via: SIP/2.0/UDP 153.96.14.162:5060.
From: sip:bkraegelin@iptel.org.
To: sip:bkraegelin@iptel.org.
Call-ID: 0009b7aa-1249b554-6407d246-72d2450a@153.96.14.162.
CSeq: 101 REGISTER.
WWW-Authenticate: Digest realm="iptel.org", \
  nonce="3d9385170000000043acbf6ba...", algorithm=MD5.
Server: Sip EXpress router(0.8.8 (i386/linux)).
Content-Length: 0.
Warning: 392 127.0.0.1:5060 "Noisy feedback tells: pid=31604 ".
```

4.1.4.
1.3. Tracing Errors in Server Chains.

A request may pass any number of proxy servers on its path to its destination. If an error occurs in the chain, it is difficult for upstream troubleshooters and/or users complaining to administrators to learn more about error circumstances.

A nice utility for debugging server chains is sipsak, SIP Swiss Army Knife, traceroute-like tool for SIP developed at iptel.org. It allows you to send OPTIONS request with low, increasing Max-Forwards header-fields and follow how it propagates in SIP network. See its webpage at http://sipsak.berlios.de/ [2].

Example 4.2. Use of SIPSak for Learning SIP Path

```
[jiri@bat sipsak]$ ./sipsak -T -s sip:7271@iptel.org
warning: IP extract from warning activated to be more informational
0: 127.0.0.1 (0.456 ms) SIP/2.0 483 Too Many Hops
1: ?? (31.657 ms) SIP/2.0 200 OK
        without Contact header
```

Note that in this example, the second hop server does not issue any warning header fields in replies and it is thus impossible to display its IP address in SIPsak's output.

4.1.4.
1.4. Server Status Monitoring.

[2] http://sipsak.berlios.de/

It is essential for solid operation to monitor server status continuously. We've been using two tools for this purpose. Sipsak does a great job of “pinging” a server, which may be used for alerting administrators of unresponsive servers.

Monit is a server watching utility which alerts administrators when a server dies.

4.1.4.
1.5. Dealing with DNS.

The SIP standard leverages DNS. Administrators of SIP servers should be aware of the impact of DNS on server's operation. A server's attempt to resolve an unresolvable address may block a server's process for a duration in the order of seconds. To be safer that the server does not stop responding due to being blocked by DNS resolving, we recommend the following practices:

- Start a sufficient number of children processes. If one is blocked, the other children will keep serving.
- Use DNS caching. For example, in Linux, there is an nscd daemon available for this purpose.
- Process transactions statefuly if memory allows. That helps to absorb retransmissions without having to make DNS queries for each of them.

4.2. Dialplans

The previous sections already addressed issues regarding the dialplan that is used. There is no ideal solution to address all different needs but just a number of techniques to solve specific. This section addresses the most common problems faced when dealing with dialplans.

1. *IP telephony numberblocks*

   The general situation is that there is an already existing PBX dialplan and IP telephony shall be integrated into this dialplan. If the existing dialplan has a free number block then the first approach would be to give IP telephony the whole block. This makes the configuration very simple because it allows prefix based routing (see Section 4.1.1.1.1). The problem is that either only new telephone users get an IP telephone or that every user who wants to use IP telephony gets a new phone number. So this approach is not suitable for a seamless migration towards IP telephony

2. *IP telephony service prefix*

   Another solution is to define a prefix that has to be dialed to reach an IP telephone. As mentioned before prefix routing is the easiest option to configure. An IP telephony prefix would also allow a user changing from a legacy phone to an IP telephony phone to keep his number modified in a way that is easy to remember (e.g. if the internal number was 2972 and the prefix for VoIP is 99 than the new number is 992972 - which applies for all numbers).

   On the other hand there must be a way to decide of a call that originates on the IP side has an IP telephony target or a phone on the PBX. This can again be realized with a service prefix for legacy phones or by making the PBX the default route for targets that are not registered at the same server. This must be carefully considered to avoid that a call from an IP phone to another IP phone target that is not currently registered is routed back and forth between IP telephony server and PBX.

   The problem with this solution is that you have to know if the person you want to call has an IP phone or not and this constitutes a number change which still requires all business cards to be reprinted.

   To avoid this number change to be "visible" the PBX might set up a mapping table that maps outdated old addresses to the new addresses. So the PBX maps the dialed 2972 to 992972 and routes the call to the IP world.

3. *Per number routing*

   The cleanest way to handle call routing is to perform routing decisions on the individual number (see Section 4.1.1.1.2). The fact that a number belongs to an IP phone or PBX phone is fully transparent to the user and no error-prone default routes are required. It is also the solution that has the highest configuration and administration effort because there are most probably at least two databases that must be kept consistent.

4. The call routing problem gets worse as soon as multiple call signaling protocols are deployed in the IP world and no single server supporting all of them at once (see Figure 4.5). Every IP telephony server must be aware that a number that belongs to another server must be routed to the gateway, or otherwise the gateway must be the default route for unknown targets. In any case, calls for unknown targets land on a gateway. Now the gateway needs to decide where to route a call. Because it is desirable that gateways are dumb (to prevent having yet another place to configure routing details) the gateway will hand the call to the PBX which makes the final routing decision - which eventually means to hand the call to another gateway (or back to the originating if there is only one multi-protocol gateway).

All problems and solutions mentioned above are highly dependent on specific products and the features they support, so unfortunately there can be no general advice on how to implement dialplan migration.

4.3. Authentication and Billing

To charge individuals for used services it is necessary to have means of authentication for registration and call signaling. This section gives an overview on the mechanisms used for this purpose in H.323 and SIP.

4.3.1. Authentication in H.323

The H.323 protocol framework uses H.235 "“Security and encryption for H-series (H.323 and other H.245-based) multimedia terminals”" for optional security features. This recommendation describes how to incorporate *authentication*, *integrity* and *confidentiality* for H.323 communication and what kind of security infrastructure and techniques are supported.

4.3.1.1. Areas of application

H.235 can be applied to all aspects of H.323 communication which can be broken into two basic categories: Security for signaling messages and Security for media streams.

Security for signaling messages includes the RAS channel (see Section 2.2.1.3) that is used for registration of endpoints, admission and status of calls, as well as the call signaling channel (H.225) that is used for call establishment and the media control channel (H.245).

Security for media streams is used to provide confidentiality for transmitted audio and video data.

4.3.1.2. User Authentication

User authentication is the process by which a user performing an action proofs his identity to the server entity. Basically three different approaches can be classified: passwords with symmetric encryption, passwords with hashing and public-key mechanisms.

1. *Passwords with symmetric encryption*: This approach is based on the idea that both communicating entities share a secret - e.g. a password. - and that each endpoint has a unique *generalID* that has been configured before by means outside the protocol and that is known to both endpoints.

   The endpoint that wishes to be authenticated generates a *CryptoToken* which consists of the sender's generalID, the receivers generalID, a timestamp and a random number (that is monotonically increasing, making messages with the same timestamp unique) encoded with the secret key (derived from the shared password).

   Encryption can be done by a selection of mechanisms such as DES, 3-DES or any other algorithm that is registered in ISO/IEC 99792. It is also possible that a manufacturer can use other algorithms although this

will not be interoperable.

2. *Passwords with hashing*: this is a similar approach, where *CryptoHashedToken* is computed using the generalID and a timestamp to be passed through a hashing function like HMAC-MD5 or algorithms defined in ISO/IEC 9797.

3. *Public-key mechanisms*: this approach also generates a *CryptoToken* but uses asymmetric encryption. This enables the use of signature cards and certificates.

4.3.1.3. Integrity

Integrity refers to message integrity and ensures that a received message is identical to that which was transmitted by the sender and has not been modified.

H.235 supports two mechanisms to achieve integrity: the use of *CryptoTokens* and the *IntegrityCheckValue*.

1. *CryptoTokens* are already described in Section 4.3.1.2. To allow an integrity check the whole message is used to compute a MAC/digital signature - instead of just a small subset required for authentication.

   This mechanism can be used for any signaling channel (RAS/H.225.0/H.245).

2. *IntegrityCheckValue* refers to an element that occurs in RAS messages. Again the hash-value of the message (without the hash-value) is transmitted.

   This second mechanism was introduced before the adoption of H.235, because it was deemed critical that there was not a data integrity mechanism for the unreliable RAS channel. Since the adoption of H.235, the CryptoToken method is the preferred way to check integrity.

4.3.1.4. Confidentiality

Confidentiality ranges from secured H.225.0 signaling channels to secured media streams. The H.323/H.235 suite of protocols does not specify a way to secure the signaling channels, because they are used first in every call but have to be secured from the very beginning. Instead *Transport Layer Security (TLS)* or IPSEC shall be used to secure the H.225.0 channel. An endpoint supporting such a mechanism must listen on a well known port (e.g. 1300 for TLS) to receive secured connections.

Within H.225.0 the security capabilities for the H.245 media control channel can be exchanged. The H.245 channel itself can be used to negotiate media encryption.

4.3.1.5. Security profiles

H.235 defines three security profiles - the "“Baseline security profile”", the "“Signature Security Profile”" and the "“Voice encryption security profile”". Each profiles defines a collection of H.235 mechanisms that must be supported by an endpoint.

The *Baseline security profile* is the simplest profile and is suitable for providing authentication and integrity in password based environments. Most endpoints that claim H.235 support implement (only) this profile.

The *Signature security profile* is the same as the Baseline Security Profile but uses digital signatures instead of passwords.

The *Voice encryption security profile* defines mechanisms to achieve confidentiality for the media streams. It can be used along with one of the other security profiles that achieve authentication.

4.3.1.6. H.235 and the real world

While H.235 exists for some years now there are not many H.323 products that support it. Some products only use H.235 (baseline security) for the communication between gatekeeper and gateway and not for communica-

2An unofficial list can be found at http://www.isg.rhul.ac.uk/~cjm/ISO-register/.

tion with the endpoint. And even in cases where endpoint communication uses H.235, it is often not interoperable between different vendor products because H.235 does not mandate a minimum set of algorithms that can be used, or which elements generated tokens must consist of. So, when interested in H.235 in your IP telephony network, ask your IP telephony vendor for a list of compatible equipment.

4.3.2. Authentication in SIP

In this section we describe authentication mechanisms used in SIP based networks. First we describe basics of digest authentication, while in following sections we give a brief overview of how digest authentication applies to SIP messages, when it should be used and when it should not.

4.3.2.1. Overview of Digest Authentication

Digest authentication is a simple authentication mechanism developed originally for HTTP (it is often called HTTP digest) and it is described in RFC2671. The authentication mechanism is very simple. It is based on cryptographic hashes to prevent transferring of the user's password in clear-text.

Digest authentication verifies that both parties that communicate know a shared secret (a password).

When a server wants to authenticate a user, it generates digest challenge and sends it to user. A typical digest challenge looks like this:

```
Digest realm="iptel.org", qop="auth,auth-int",
  nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093", opaque="", algorithm=MD5
```

It consists of a set of parameters that are sent to the user. The user then uses the parameters to generate proper digest reply and send it back to the server. The meaning of the parameters in the digest challenge is as follows:

- *realm*: The realm parameter is mandatory and must be present in all challenges. Its purpose is to identify credentials within a SIP message. In case of SIP, it is usually set to the domain the proxy server is responsible for.

  SIP user agents are supposed to display the contents of the parameter to the user when they prompt him for username and password so that he uses the appropriate username and password (for this server).

- *nonce*: this is a server specified data string which is uniquely generated each time a server generates a digest challenge. Nonce is usually constructed as the MD5 hash of some data. The data usually includes time-stamp and a secret phrase of the generating server. That ensures that each nonce has limited lifetime (i.e. expires after some time and can not be used later) and also is unique (i.e. no other server will be able to generate the same nonce).

  Clients use the nonce to generate a digest response and thus the server will receive the contents of the nonce back in a digest response. It usually checks the validity of the nonce before it checks the rest of the digest response.

  So, basically, nonce is a sort of an identifier that ensures that received digest credentials have really been generated for a particular digest challenge, and also limits the lifetime of the digest response, preventing replay attacks in the future.

- opaque

- *algorithm*: the algorithm used to calculate hashes. Currently only MD5 is supported.

- *qop*: The quality of protection. The parameter specifies what protection schemes does the server support. A client will pick one from the list. The value “auth” indicates just authentication, the value “auth-int” indicates authentication with some integrity protection. For more detailed description, see RFC2617[4].

After receiving the digest challenge, a user agent will prompt the user for username and password (if not preconfigured), generate a digest response and send the response to the server. A digest response might look like this:

[4] http://www.ietf.org/rfc/rfc2617.txt

```
Digest username="jan", realm="iptel.org",
  nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093", uri="sip:iptel.org",
  qop=auth, nc=00000001, cnonce="0a4f113b",
  response="6629fae49393a05397450978507c4ef1", opaque=""
```

As we can see, the digest response is similar to the digest challenge. Those parameters that are the same have the same meaning as in the digest challenge. We will briefly describe only new parameters:

- *uri* - The parameter contains URI the clients wants to access.
- *qop* - The level of protection chosen by the client.
- *nc* - Nonce count, the value is the hexadecimal count of the number of requests (including the current request) that the client has sent with the nonce value in this request. For example, in the first request sent in response to a given nonce value, the client sends “nc=00000001”. The purpose of this directive is to allow the server to detect request replays by maintaining its own copy of this count - if the same value is seen twice, then the request is a replay.
- *cnonce* - The value is an opaque quoted string value provided by the client and used by both client and server to avoid chosen plain-text attacks, to provide mutual authentication, and to provide some message integrity protection.
- *response* - A string computed by the user agent which proves that the user knows a password.

Upon reception of a digest response the server recalculates the value of the response parameter for comparison purposes, using attributes provided by the client and the password stored on the server. If the result is identical to the response received from the client then the client has proven knowledge of the password and he is authenticated.

4.3.2.2. Digest Authentication and SIP

We have described what digest challenge and response looks like, but we have not described yet how they are applied to SIP messages. Since the authentication mechanism was originally developed for the HTTP protocol and SIP is very similar to that protocol, mapping of digest challenge and response to SIP messages is easy and straightforward. It is described in RFC3261[5]

When a SIP server receives a SIP request and wants to verify the authenticity of the user before processing the requests, it looks if the request contains digest credentials. If there are no credentials in the SIP request, it will generate a negative final response and include digest challenge into the response.

When a client receives the response (containing digest challenge), it is supposed to calculate proper digest response and send the request again, this time including the calculated digest credentials.

The server then verifies the digest response and processes the request is the verification was successful.

Proxy servers use “407 Proxy Authentication Required” response and include the digest challenge into the “Proxy-Authenticate” header field. An example of such a challenge might look like:

```
SIP/2.0 407 Proxy Authentication Required.
Via: SIP/2.0/UDP 195.37.78.121:5060.
From: sip:jan@iptel.org;tag=3944790419.
To: <sip:5060@iptel.org;user=phone>;tag=794fe65c16edfdf45da4fc39a5d2867
Call-ID: 3541699089@195.37.78.121.
CSeq: 1 INVITE.
Proxy-Authenticate: Digest realm="iptel.org", \
  nonce="3f9fc19cf91f65958f664122c1310d4c28cc61a2".
Content-Length: 0.
```

[5] http://www.ietf.org/rfc/rfc3261.txt

SIP user agents (including registrars and back-to-back user agents) use the “401 Unauthorized” response for the digest challenge. An example of such a challenge might be:

```
SIP/2.0 401 Unauthorized.
Via: SIP/2.0/UDP 218.79.100.193:65030;branch=z9hG4bK1ce21dab.
To: "IPTel844978" <sip:844978@iptel.org>;tag=794fe65c16edfdf45da4fc39
From: "IPTel844978" <sip:844978@iptel.org>;tag=1fd6218e.
Call-ID: 2d471abf-c0fbee95-bee93355-fea1736b@218.79.100.193.
CSeq: 88608141 REGISTER.
WWW-Authenticate: Digest realm="iptel.org", \
  nonce="3f9fc19cf91f65958f664122c1310d4c28cc61a2".
Content-Length: 0.
```

407 responses are used by SIP elements (mostly SIP proxy servers) that are not the final destination for the request and after authentication will forward the requests further. 401 responses are used by SIP elements that are the final destination for the request and after authentication will generate a final reply.

When including the digest response clients add “Authorization” or “Proxy-Authorization” header field that contains the digest response. The following example shows a REGISTER message containing digest credentials.

```
REGISTER sip:iptel.org SIP/2.0.
Via: SIP/2.0/UDP 195.37.78.121:5060.
From: sip:jan@iptel.org.
To: sip:jan@iptel.org.
Call-ID: 003094c3-bcfea44f-40bdf830-2a557714@195.37.78.121.
CSeq: 102 REGISTER.
User-Agent: CSCO/4.
Contact: <sip:jan@195.37.78.121:5060>.
Authorization: Digest username="jan",realm="iptel.org",
   uri="sip:iptel.org",response="dab81127b9a7169ed57aa4a6ca146184",
   nonce="3f9fc0f9619dd1a712b27723398303ea436e839a",algorithm=md5.
Content-Length: 0.
Expires: 10.
```

4.3.2.3. Basic Scenarios

We have described what digest authentication looks like and how digest challenges and responses are carried in SIP messages. In this chapter we will look at which SIP messages can be challenged and which can not. We will also describe two most common situations in which digest authentication is used.

When a SIP user agent receives a digest challenge, it is supposed to re-send the same request again, but this time with proper digest credentials. That also means that the user agent must increase the CSeq number in the request in order to avoid treatment the new request as a retransmission by the server.

Because challenging a request means that the request will be sent again with higher CSeq, it is not possible to challenge ACK and CANCEL requests. Both the requests must have the same CSeq as the original request and thus can not be challenged.

All other requests can be challenged, although from time to time there appear implementations that seem to have problems with the challenging of the not so common SIP requests.

There are two cases which are deployed most often and deserve further description: authentication of REGISTER messages and authentication of INVITE messages. We will describe them in separate chapters.

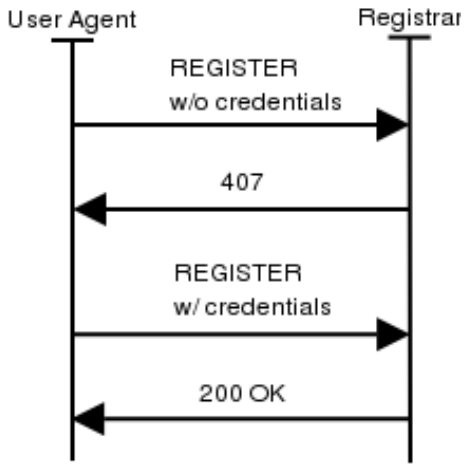
4.3.2.3.1. Registration Authentication

Authentication of REGISTER messages is very important and should be done by every SIP proxy server. By REGISTER messages SIP user agents are informing the server of their current location so the server knows where to send further requests.

If a server does not authenticate REGISTER requests then anyone can register any contact for any user, thus hi-jacking calls to that person. This is obviously extremely important to protect against and therefore authentication of REGISTER messages should always be enabled.

Figure 4.9 shows call flow of a typical SIP registration including digest authentication.

Figure 4.9. REGISTER Message Flow



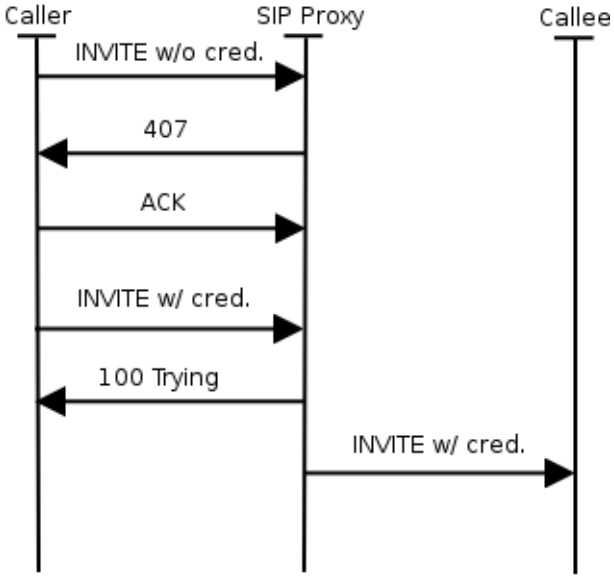
4.3.2.3.2. Invite Authentication

Authentication of INVITE requests is not really required, but it is a good practice to do so. A SIP proxy server can only challenge requests that are coming from users belonging to an administrative domain the proxy server is responsible for. This means that a proxy responsible for e.g. the iptel.org domain can challenge only requests that have iptel.org in the “From” header field.

Requests coming from foreign users can not be challenged because foreign users usually do not have a username and password registered at this server. Requiring authentication would make incoming calls from foreign users impossible.

Figure 4.10 is a call flow of challenged INVITE.

Figure 4.10. INVITE with authentication



4.4. Examples

This section lists some examples on how a zone setup could look like depending on the requirements.

4.4.1. Example 1: Simple, use IP telephony like legacy telephony

Assumption: An institution currently using a PBX with internal numbers of four digits length. There telephone numbers from 6000 to 6999 are available for IP telephony. There are no requirements regarding authentication. Because only calls into the PSTN shall be billed the PBX is the only place where billing shall take place. There are no special requirements regarding availability and there is no demand for IP telephony research.
Components: Any kind of IP telephony server (H.323 gatekeeper or SIP proxy)- even productions using proprietary protocols are usable. The gateway must be able to translate signaling between the protocol that the server uses and the PBX. The protocol to the PBX usually uses one of the protocols described in Section 5.1.1.
Structure: See Figure 4.11.
Call Routing: The PBX is configured to route every call to a number starting with 6 to the gateway. The IP telephony server either has the gateway as a default route for unknown/unregistered targets or is configured to route every call to a number that does not start with 6 to the gateway too. The gateway can either be configured to always route a call from one side to the other or needs to be get a configuration similar to the IP telephony server.
Authentication: Authentication on the IP side is either done using the H.323 or SIP authentication mechanisms or can be done on the link layer. In the latter case a telephone number is bound to a specific port or MAC address.
Billing: The billing mechanisms that were already in use for PBX calls can be used for IP telephony as well as all outgoing calls pass the PBX.

The described solution allows an easy integration of IP telephony into a PBX world. The gain of this solution compared to just more legacy phones is that IP telephony allows more flexibility regarding the endpoints - allowing hard- and software phones that may even be connected by wireless LAN (depending on the authentication mechanisms used).

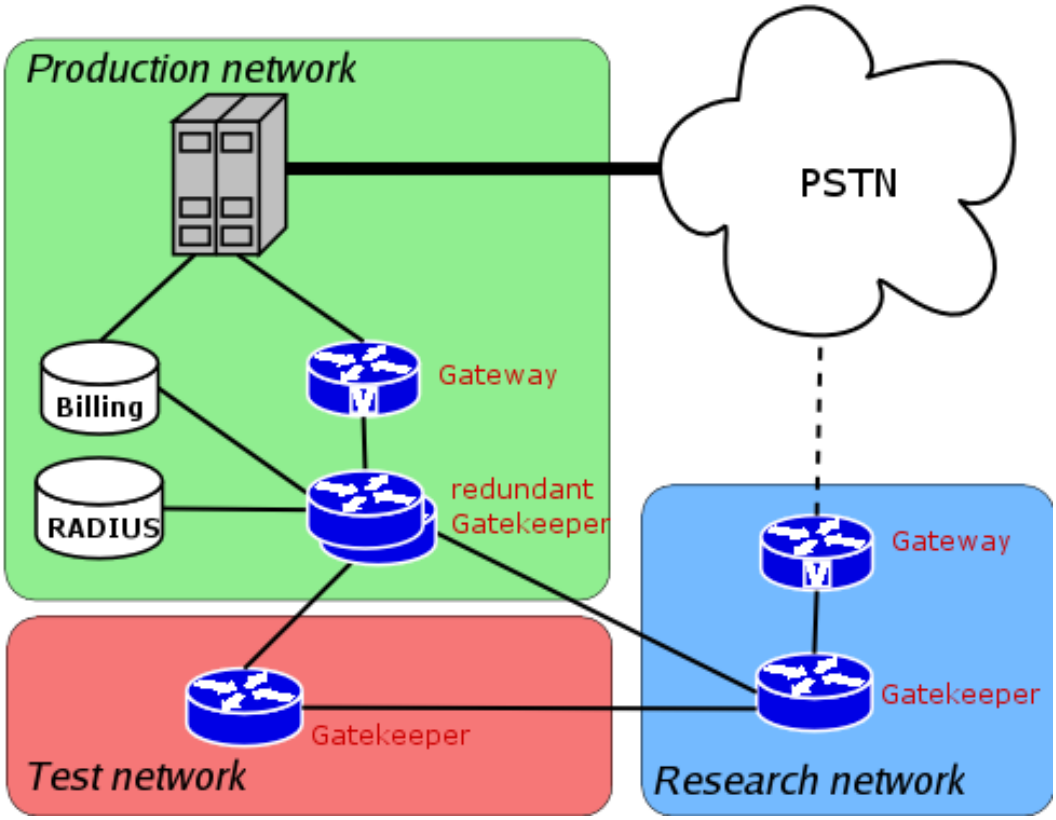
The problems of this solution are that it heavily relies on the PBX that remains the core element of the infrastructure. If once there is demand for more IP telephony accounts more number blocks must be available - to free such a block requires giving legacy phone users to numbers. The solution also is not prepared to make use of the Internet for long distance calls or select an IP telephony service provider.

4.4.2. Example 2: Complex, full featured

Assumption: A university with multiple locations, a shared unstructured dialing space and need for borth SIP and H.323. It should be possible to test new IP telephony server firmwares before installing them in the production network. To stretch this idea further an additional requirement is that the IP telephony system has to be divided into three *logical networks*: a production network (the telephone system for 90% of all employees), a *testing network* to run new firmware versions before deploying them in the production network, and a *research network* for IP telephony related research work. Obviously the networks differ in reliability - having high reliability requirements in the production network and nearly none in the research network. A daring user might decide to participate in the testing network - without changing his phone number or using a second phone.
Components: To be able to do IP telephony research on standardized protocols the research network runs either a H.323 gatekeeper or a SIP proxy. The production network runs a redundant server that supports H.323 as well as SIP. The testing network uses the same server model - without redundancy. The gateway is either a H.323/PSTN or SIP/PSTN gateway. A RADIUS server stores all valid users (names and numbers) along with their password. The billing records can be written by the PBX and the IP telephony server - e.g. using a SQL server.
Structure: Figure 4.12 describes how the servers are organized. There is a H.323 gatekeeper or SIP proxy for each logical network. Which logical network an endpoint belongs to is simply defined by which server it is registered with and is independent from the physical network structure. To participate in testing of new features, the endpoint of the user need only be configured to register on the server using the new firmware version.
Call Routing: Routing decisions are either made using a shared database (see Section 4.1.1.1.3) or by routing calls to external targets via the server in the production network to the PBX gateway. A server whose user dials an internal number tries other locally registered endpoints first, before asking the peer server using the LRQ mechanism of H.323.
Authentication: To achieve authentication the mechanisms described in Section 4.3 are used. The authentication backend is provided by a RADIUS server that stores logins and passwords.
Billing: Because external calls are routed through the PBX the existing billing solution may be used. If the pro-

duction network gatekeeper is able to write billing records as well, it will become the production billing server when sometime in the future the university selects an IP telephony provider instead a PSTN Telco.

Figure 4.12. Example of a multi-server IP telephony zone



This scenario is quite complex, but it is the most flexible. It allows individual users to move from the legacy telephony world to IP telephony, eventually reducing the PBX to a minimal state. It is made robust by using redundant servers where necessary. Because routing decisions are made on the IP side, this solution qualifies for communication with external targets via the Internet or through use of an IP telephony provider. The decision for open standards (SIP, H.323) allows soace for research initiatives and prevents dependencies on specific vendors.

All these possibilities come at a price. Several servers must be bought and the complex structure makes it harder to trace errors.

4.5. Setting up H.323 services

When setting up H.323 services, the basic component to install is a gatekeeper, in order to provide initial functionality to an installed base of H.323 clients. This basic functionality entails:

- in-zone calling among endpoints
- out-of-zone calling (incoming/outgoing)
- access to local services (e.g. gateways, multi-point conference servers)
- name resolution during calls, by H.323 alias or E.164 number
- zone management (authentication, bandwidth restriction, etc.)

In this section, we will be presenting guides for running the three most popular gatekeeper implementations that are available today. A comparison of these gatekeeper implementations based on their capabilities and requirements follows here:

Figure 4.13. Gatekeeper features examples

	Cisco MCM	Radvision ECS	GNU GK
h/w requirements	Cisco routers only	-	-
s/w requirements	IOS special images	Windows	Windows/Linux/MacOSX
availability	commercial product	commercial product	open-source
routing modes	direct, proxy	direct, call, control sig.	direct, call, control sig.
multiple zone support	yes (limited)	yes	yes
interzone routing	neighbour, DNS	neighbour, child-parent	neighbour, child-parent
failover support	HSRP	alternate gatekeeper	alternate gatekeeper
LDAP support	no	yes	yes (under dev.)
H.350 support	no	under dev.	under dev.
RADIUS support	yes	no	yes

Guides for basic operation of the above three gatekeepers follow, but official documentation for these products should be consulted when advanced functionality and features are required.

4.5.1. Using a Cisco Multimedia Conference Manager (MCM gatekeeper)

The Cisco MCM is a software gatekeeper that runs only on Cisco router hardware with special IOS images (H.323 feature set). One the one hand, this makes it easy to find a hardware platform for running it within most organizations that use Cisco hardware, without regard for underlying operating system support. On the other hand, it does not allow the flexibility of installing it on any available PC-based server. It is a commercial grade implementation, mostly geared towards VoIP gateway services and less towards an open H.323 community of endpoints that possibly spans organization borders. The MCM supports either direct mode dialing, or full routing mode, through the use of an included H.323 proxy server. Multiple H.323 zones can be configured and controlled on one MCM installation, but only in combination with subnet restriction rules for groups of endpoints. The MCM has good interzone routing features with DNS gatekeeper discovery as extra and performs great in a homogeneous "Cisco" environment, but has only basic support for RADIUS based authentication (by H.323 alias or E.164 and proprietary piggy-back password mechanism) and no support for LDAP H.350 authentication. "Cisco MCM VC: Configuring H.323 Gatekeepers" is available here[6].

4.5.1.1. Installation

Since the Cisco MCM gatekeeper is only an IOS feature (IOS being the Cisco router operating system), basic IOS installation procedures are sufficient, assuming a correct IOS image with MCM functionality has been chosen from the Cisco support site. Two tools can help you choose an appropriate IOS for your available router, but they are only available to registered users on the Cisco web site:

- Cisco IOS Upgrade Planner[7]
- Cisco Software Advisor[8]

Look for "High-Performance Gatekeeper" under features and for "IP/H.323" under feature sets. IOS versioning is a subject difficult to follow in itself. Add to this the fact that the MCM has been undergoing changes during IOS development and gatekeeper features available on different IOS versions vary significantly, and finding the right IOS-MCM combination to use for a specific hardware configuration can become time consuming. Always prefer the latest available IOS release for your hardware, assuming enough RAM is available to accommodate it.

4.5.1.2. Configuration

Working with the Cisco IOS command line interface does require some experience with basic commands, modes of operation, loading software images and configuration files, none of which procedures will be described here in detail. If you are not familiar with Cisco IOS basic commands, make sure you read an introductory guide by Cisco. To configure the Cisco MCM, you must establish command line access (telnet) to the router that runs the "IP/H.323" feature set and enter privileged (enable command) mode, indicated by the # at the prompt, before you

[6] http://www.cisco.com/en/US/customer/products/sw/iosswrel/ps1835/products_configuration_guide_chapter09186a00800d80a52.html
[7] http://www.cisco.com/pcgi-bin/Software/Iosplanner/Planner-tool/iosplanner.cgi
[8] http://www.cisco.com/pcgi-bin/Support/CompNav/Index.pl

can enter configuration commands. Enter configuration mode (config command) and then specify the "gatekeeper" section. In this section, you will need to enter the MCM configuration commands, as in the sample below, which merely initializes the gatekeeper operation:

```
gkp#config
Configuring from terminal, memory, or network [terminal]?
Enter configuration commands, one per line.  End with CNTL/Z.
gkp(config)#gatekeeper
gkp(config-gk)#zone local gkp.mydomain.org mydomain.org
gkp(config-gk)#no shutdown
gkp(config-gk)#^Z
```

The above sample is sufficient to start gatekeeper services on the router, but a more detailed configuration with comments for a basic gatekeeper set-up follows. We have dropped the command line prompt for simplicity. The following commands can be typed at the configuration interface, as shown above. Note that all user-specified fields are indicated as enclosed in brackets and you must customize/replace them appropriately for your site.

```
!
! This section goes outside the gatekeeper configuration section
! as it relates to general AAA settings and RADIUS server communication.
! H.323 endpoint RAS registration will be checked against
! local IOS usernames first and then RADIUS defined usernames.
! Accounting records will be sent to RADIUS server.
!
aaa new-model
aaa authentication login h323 local group radius
aaa accounting connection h323 start-stop group radius
!
radius-server host [radius.mydomain.org] auth-port [1812] acct-port [1813]
radius-server key [radius-server-key-as-defined-in-radius-host]
radius-server authorization permit missing Service-Type
!
! Gatekeeper section
!
gatekeeper
 !
 ! Local zone info, as controlled by this gatekeeper
 ! The zone name "myzone" is for config purposes only and plays no role,
 ! while the domain is important for endpoints registering by e-mail alias
 ! The zone prefix is important for recognizing
 ! in-zone calls and endpoints, e.g anything beginning with 0030234
 !
 zone local [myzone] [mydomain.org]
 zone prefix [myzone] [0030234*]
 !
 ! To set-up connectivity with other zones and gatekeepers,
 ! specify the ip of the neighbouring gatekeeper with a "zone remote"
 ! and the prefix it services with a "zone prefix" for that zone.
 ! For example, if you know a neighbour gatekeeper handles
 ! all calls with prefix 0030248, include the following two lines.
 !
 zone remote [neighb1] [neighb1-domain.com] [neighb1-gkp-ip] 1719
 zone prefix [neighb1] [0030248*]
 !
 ! The VideNet gatekeeper for example is the largest global network
 ! of H.323 zones, to which you can connect as shown below.
 ! Any calls beginning with 00, are routed to the VideNet gatekeeper.
 ! In order to accept calls from VideNet as well,
 ! you have to make your gatekeeper well known to the VideNet
 ! hierarchy of gatekeepers (see https://videnet.unc.edu/)
 !
 zone remote videnet3 videnet 137.44.172.248 1719
 zone prefix videnet3 00*
 lrq forward-queries add-hop-count
 !
```

```
! To force endpoints to register with a specific h323-id and password
! you can use H.235 (few endpoints support it) or the h323-id/password
! mechanism that the MCM provides.
!
accounting
security h323-id
security password separator /
```

4.5.1.3. Operation

Immediately after configuration, the MCM may service endpoints, and you can verify this by making a couple of endpoints point to the gatekeeper for registration. As soon as the endpoints register, they can be listed with the following command:

```
> show gatekeeper endpoints
```

You may proceed with calling between the two endpoints by dialling from the one the registered aliases (name or number) of the other. The ongoing call can be listed with the following command:

```
> show h323 gatekeeper calls
```

As an administrator of the gatekeeper you may disconnect the call, or even unregister an endpoint.

```
> clear gatekeeper call call-id . . .
> unregister . . .
```

A view of the operational status of the gatekeeper, such as zones defined, endpoints registered, neighbour gatekeepers defined etc. may be displayed by the following command:

```
> show gatekeeper status
```

Debug logs of the gatekeeper operations may be monitored with the following sequence of commands:

```
> terminal monitor
> debug gatekeeper main 10
> debug h225 asn1
> debug h245 asn1
```

The first command makes your terminal capable of displaying console style logs and debugging output. The second command produces debugging output regarding basic gatekeeper actions. Obviously, the last two commands display info on H.225 and H.245 protocols and the output can be overwhelming, but it may be the only debugging option when faced with an otherwise intractable problem. Each debugging option can be stopped by its equivalent "no debug #" and all debugging output can be stopped with the "no debug all" command.

4.5.1.4. Endpoint authentication

The MCM gatekeeper implements H.235 authentication, but its use is limited to gatekeeper-to-gatekeeper and gatekeeper-to-gateway authentication, because of the very limited deployment of H.235 capable endpoints. Cisco has implemented an alternative method for endpoint authentication, which allows for an H.323 or E.164 alias to carry (piggy-back) both alias info and a password, separated by an administrator defined special character, e.g. /. A configuration for this feature is provided above and once activated, endpoints must be configured to use alias/password combinations to register with the gatekeeper. There are shortcomings to this method and stem mostly from the fact that it is a proprietary solution, which in some cases exposes clear text passwords to neighbour

devices (MCUs, gateways, gatekeepers). Of course, the MCM includes RADIUS support, which might allow for an IP address + alias identification method to be implemented on the RADIUS server side, but such a solution imposes restrictions to endpoint mobility.

4.5.1.5. Advanced features

The Cisco MCM supports RADIUS authentication and accounting to a remote RADIUS server. With the extensive support of RADIUS servers to a number of back-ends such as databases and directory services, this can be an important feature when seeking a method of integrating H.323 access control with already deployed services (e.g. dial-up, LDAP), or a simple way of storing call-accounting information in a database. Also, the exchange of standard and vendor-specific attributes during the RADIUS negotiation process allows very fine control of some delicate parameters such as "call duration" which would otherwise be inaccessible to an external-to-the-gatekeeper application. Of course, only experienced RADIUS administrators and middleware developers can exploit the full potential of the RADIUS configuration files and its back-end interfaces. The Cisco MCM supports an alternative method to neighbour discovery than static neighbour entries in the IOS configuration. A DNS-based gatekeeper discovery mechanism is in place that allows the MCM to find gatekeepers responsible for a specific domain by checking for the existence of a TXT record in the domain's DNS zone info. This can be useful if a large community of users in separate zones employs e-mail addresses for dialling. The gatekeepers serving them do not need to have static knowledge of each other, but can discover destination gatekeepers responsible for a domain through DNS. Multiple zone support is implemented on the MCM in a way that allows multiple instances of the gatekeeper to run within one router. This would have been an excellent feature, if it could have avoided a major handicap: endpoint registration to a specific gatekeeper has to be guided by administratively preset IP address subnet restrictions. Interestingly enough, Cisco gateways can utilize this functionality by indicating on their RRQ messages (by gatekeeper ID and not by IP address) which gatekeeper they request to be registered with.

4.5.2. Using a Radvision Enhanced Communication Server (ECS gatekeeper)

The Radvision ECS is a software only gatekeeper that runs on the WinNT or Win2000 operating systems, a fact that ties it to specific remote management techniques used with all other Windows based servers. It is a commercial grade implementation and it is considered top-of-the-line for the features it provides and its compatibility even with the latest H.323 specs. It is servicing large organizations with a great number of endpoints and some of the VideNet global root gatekeepers most notably. The ECS supports all three modes of routing: direct, Q.931 routing, and both Q.931 and H.245 routing. The ECS has good interzone routing features with DNS gatekeeper discovery and neighbour gatekeeper LDAP support as extra. Authentication is very flexible, with ability for "predefined" endpoint settings enforced at registration time and LDAP H.350 support, but no RADIUS support.

"ECS gatekeeper product description" is available here[9].

"ECS gatekeeper specifications" are available here[10].

4.5.2.1. Installation

Installing the ECS gatekeeper is a very simple task, since it involves merely the execution a GUI setup wizard, which requires no configuration options. The only potential source of installation problems lies with the fact that the Windows SNMP service must be already installed, before any service packs and the ECS installer are applied. If this advice, which is listed in the ECS documentation, is ignored, the ECS installer refuses to proceed and the only option is to reinstall the operating system itself. Also, the administrator of the host must make sure that port 80 is free, since the ECS installs an HTTP service on this default port for configuration management over a web interface. The documentation also calls for an FTP server to be running at the same host, but it only serves for downloading ECS log files, which is not a required functionality.

4.5.2.2. Configuration

Once installed, the ECS is ready to run with default configuration options. The administrator can access the management interface by launching a browser and requesting the local web server (http://localhost). The interface presents a login page, where the default username and password can be entered (admin/null-no-password). After

[9] http://www.radvision.com/NBU/Products/viaIP+Custom+Solutions/Gatekeeper+%28ECS%29.htm
[10] http://www.radvision.com/NR/rdonlyres/7D510F55-F4EA-408C-8721-4ACA39ADDA3E/320/ECSDatasheet1.pdf

successful login, the administrator is made aware of the fact that the management tool can supervise the operation of a whole hierarchy of ECS gatekeepers ("Global" picture), as well as the single ECS installation residing on this host ("Local" picture). Proceed with the "Local administrator" interface.

There are four commands to allow configuration management. The "Refresh" button fills in the web interface forms with configuration data from the currently running ECS gatekeeper configuration. The "Upload" button takes all the changes made on the web interface and applies them to the currently running ECS configuration. The "Import" and "Export" buttons are used to store and retrieve snapshots of the configuration at different points in time.

The rest of the interface is fairly straightforward, with an array of configuration Tabs (sections), the most important of which are listed below:

- Status Tab: allows view of the current status of the gatekeeper by indicating the number of ongoing calls and registered endpoints, as well as bandwidth usage statistics for in-zone and out-of-zone calls.
- Settings Tab: this is where most of the configuration options are specified, logically separated into a series of thematic categories (Basics, Calls, Dial Plan, Supplementary Services, Logs, LDAP, DNS, Security, Alternate Gatekeeper, Advanced).
- Endpoints Tab: allows view and control of the currently registered endpoints with details on their aliases (name and number), IP addresses and online time. This Tab can be used to "predefine" endpoints, i.e. assign specific aliases to endpoints that may later be used for endpoint identification during authentication.
- Services Tab: allows view and configuration of the currently declared services. By default, four services exist at installation time and they are not activated, since their prefix setting is null. Therefore they merely exist as templates for defining basic services functionality.
- Call Control Tab: allows view and control of the calls in progress and the setting up of gatekeeper initiated calls between arbitrary endpoints, with the "Make call" option, assuming the gatekeeper runs in fully routed mode (see Signaling Models in this book and the Settings Tab, category Calls in the ECS interface).
- Forwarding Tab: allows set-up of forwarding rules based on source and destination, for three cases: forward on busy, forward on no answer, unconditional forward.
- Hierarchy Tab: allows set-up of a parent gatekeeper in order to forward Location Requests for cases of unresolved destinations. User assigned filters may also be applied to specify and control the extent of the cases referred upstream, to the parent gatekeeper.

Even though the ECS gatekeeper runs out-of-the-box, you may want to inspect some of its basic settings and decide whether they fit the needs of your application. There are three Tabs that should be at least browsed through before proceeding with operation.

Under the Settings Tab, in the category "Basic", make a note of the name of the gatekeeper (gatekeeper ID). Also, be aware of the setting "who can register", where the choices are "everyone" for no authentication control, "predefined endpoints only" for some authentication control and "no endpoints" to turn down all endpoints for maintenance reasons only. The choice between "dial plan v.1" and "dial plan v.2" may not be obvious, but keep in mind that the second option allows more flexibility in hierarchically connected gatekeeper environments. Once chosen, it dynamically enables extra configuration sections. The option for "DHCP environment" may be used for authentication control, as it instructs the gatekeeper to identify endpoints by previously seen IP addresses and H.323 aliases (names) and authenticate them based on this information. The last choice, "merge predefined and on-line aliases upon registration" is an interesting feature, because it allows the gatekeeper to apply extra aliases to well-known and identified endpoints. E.g. an endpoint may register with a name alias only, but the gatekeeper will attach an E.164 number to this endpoint as well.

Under the Settings Tab, in the category "Calls", be aware of the "routing mode" selection, as it varies the operation of the gatekeeper dramatically. "Direct" mode employs minimal communication between endpoints and gatekeeper (RAS messages only), while "Call set-up routing" mode forces call set-up messages to be routed through the gatekeeper as well (Q.931). The third mode, forces all previous messages, as well as call control messages to be routed through the gatekeeper and not directly between the endpoints. The setting of "accept calls" can be used for maintenance reasons to turn off all calling between endpoints.

Under the Settings Tab, in the category "Dial plan", assuming you have chosen dial plan version 2, you will be able to specify stripping of zone prefixes from destination info of incoming calls. This feature may allow a more user-friendly dial plan, where in-zone endpoints use shorter dial numbers for dialling and out-of-zone endpoints use full length dial numbers.

In quick passing, check the category "Logs" if you would like to enable logging for debugging purposes and the

category "Billing" to enable usage statistics and accounting. Category "DNS" will allow discovery of neighbour gatekeepers through specially crafted DNS TXT records, but it seems to be compatible only with other RADVISION gatekeepers. Category "LDAP" will allow endpoint alias data and neighbour data to be retrieved from LDAP directory services, as well as LDAP-enabled endpoint authentication.

4.5.2.3. Operation

Immediately after installation, the ECS may service endpoints, and you can verify this by making a couple of endpoints point to the gatekeeper for registration. As soon as the endpoints register, they appear at the Endpoints Tab. You may proceed with calling between the two endpoints by dialling from the one the registered aliases (name or number) of the other. The ongoing call will appear in the Call Control Tab. As an administrator of the gatekeeper you may disconnect the call, or even unregister an endpoint from the respective Tab sections. Logs of the gatekeeper operations may be started through the Settings Tab, Logs subsection and can be inspected as text files from the `C:\Program Files\Radvision\ECS\Gatekeeper\Logs` directory, where they are maintained and rotated, after they reach a certain size.

4.5.2.4. Endpoint authentication

The ECS gatekeeper implements H.235 authentication, but its use is limited to gatekeeper-to-gatekeeper and gatekeeper-to-gateway authentication, because of the very limited deployment of H.235 capable endpoints. The ECS implements a method of storing informational data for well-known endpoints (predefined endpoints). This feature allows for an IP address + alias identification method to be implemented, but such a solution imposes restrictions to endpoint mobility.

4.5.2.5. Advanced features

The ECS gatekeeper is able to support hierarchies of gatekeepers (child-parent relationships) in cases where many levels of prefixes must be supported by prefix stripping or prefix substitution. For example, a country-level (parent) gatekeeper may need to know all dialed destinations by their 12-digit number, while an organization-level (child) gatekeeper may be able to operate with just 4-digit numbers most of the time. In order for the child gatekeeper to support both long and short dial strings, it needs to implement prefix stripping.

The H.450 protocol provides the implementation framework for supporting in H.323 a number of features common to conventional PBX systems. The ECS implements the H.450 protocol specifications, thus enabling many different types of forwarding: forward on busy, forward on no answer, forward on reject, etc. These features are supported only when the gatekeeper is in the full-routing mode (both call and control signal routing).

The ECS already has support for retrieving endpoint and neighbour data from LDAP, but it does so in a proprietary way. New developments in LDAP-enabled voice-over-IP services have given rise to H.350, the standardized protocol for storing and retrieving user settings and preferences regarding H.323 and SIP services. Radvision, being an active partner in the committee that developed the H.350 standard (previously known as H.LDAP or CommObject), has made the commitment to implement it in the ECS gatekeeper.

Until very recently, gatekeepers used to be single points of failure for voice-over-IP services, as endpoints in H.323 can only be registered with one gatekeeper. The ECS implements a special feature called "Alternate Gatekeeper", where two identical ECS gatekeepers on two different nodes can act in tandem, providing resilience in gatekeeper services transparently to the endpoints. This is achieved by constant exchange of information and status checking between a master and a slave gatekeeper, so that the second one can assume the role of the first in case of failure. In this case, some of the calls in progress may be disconnected, but at least redialing should be successful, without requiring the endpoints to register to a new gatekeeper.

4.5.3. Using an OpenH323 Gatekeeper - GNU Gatekeeper

The GNU GK is the most popular and active in development of the open-source gatekeeper projects that stem from the OpenH323 project efforts. Being an open-source effort, it benefits from availability for many different operating systems and from flexibility in configuring a multitude of features and interfaces that are not usually available in commercial products, and all these with no licensing cost. At the same time, its initial installation is made problematic by lack of quality documentation and good versioning vs. feature availability support, in contrast to a very active mailing list that users can seek help with. The GNU GK supports all three modes of routing:

direct, Q.931 routing, and both Q.931 and H.245 routing. It has only basic inter-zone routing features, but authentication is very flexible, with very configurable RADIUS support and LDAP H.350 support in the works.

"OpenH323 Gatekeeper - The GNU Gatekeeper Documentation" is available here[11].

4.5.3.1. Installation

Installing the GNU GK gatekeeper is not a simple task, if you decide to compile the source of the gatekeeper and the two libraries it requires. However, this may be your only option, if support of MySQL and LDAP is required, since the provided precompiled binaries are lacking it. To avoid compilation of the code please refer to the Pre-Built binaries downloads at the end of this section. In order to compile and build the GNU GK you will need both the PWLib libraries (version 1.2 or later) and the OpenH323 libraries (version 1.8 or later), if you are not familiar with those libraries please refer to their web site on how to build them.

These libraries are available here[12]. See the instructions on how to compile the code available here[13].

Recommended versions of the libraries are PWLib 1.4.11 or later and Openh323 1.11.7 or later. The order of compiling the packages is the following:

- PWLib (release + debug version)
- OpenH323
- OpenH323 test application (not needed, just to make sure everything works so far)
- The GNU Gatekeeper itself

To compile the GNU gatekeeper on Unix, do a "make debug" or "make opt" in the gatekeeper source directory to build debug or release versions, respectively. Use "make both" to build both versions. Note you have to use GCC 2.95.2 or later. Good practice is to do a "make debugdepend" or "make optdepend" in the gatekeeper source directory before starting actual compilation (make debug or make opt). On Windows just open and compile the provided project (gk.dsw) for Microsoft Visual C++ 6.0 or 7.0 (Visual C++ 5.0 is too old).

The Gatekeeper supports MySQL and LDAP back-end interfaces (support for LDAP is still under development). The make scripts will look for the MySQL and OpenLDAP libraries in standard places, but if they are not found, you will have to explicitly point to their source directories by config options. If you do not want MySQL support, you may set the NO_MYSQL environment before making:

```
$ NO_MYSQL=1 make both
```

To leave out LDAP support:

```
$ NO_LDAP=1 make both
```

Or disable both with

```
$ NO_MYSQL=1 NO_LDAP=1 make both
```

For gatekeepers with a large numbers of concurrent calls, the GNU GK has implemented an extended "fd_set" structure that enables the Gatekeeper to support thousands of concurrent calls in routed mode. To enable this feature, export the LARGE_FDSET environment variable to the maximum number of file descriptors. For example:

```
$ LARGE_FDSET=16384 make opt
```

The GNU GK includes implementation of a Radius protocol client that enables registration/admission authentic-

[11] http://www.gnugk.org/h323manual.html
[12] http://www.openh323.org/code.html
[13] http://www.openh323.org/build.html

ation and authorization using Radius servers. This featured is enabled by default. To disable compilation of these Radius modules, set the NO_RADIUS environment variable before making:

```
$ NO_RADIUS=1 make both
```

The GNU GK is able to do accounting. Currently, only RADIUS and plain text file accounting modules are available. The accounting is still considered an experimental feature, so it is not compiled in by default. To enable accounting, set the HAS_ACCT environment variable before making:

```
$ HAS_ACCT=1 make both
```

Moreover, there is no special installation procedure needed. After compilation, copy the executable to a directory of your choice and create a configuration file for it. There are several configuration examples in the etc/ subdirectory of the source tree. See the next section on Configuration for further explanations.

For example, to start the gatekeeper, a command like:

```
$ /usr/sbin/gnugk -c /etc/gnugk.ini -o /var/log/gnugk.log -ttt
```

should work if the configuration file (gnugk.ini) is correct.

Pre-Built binaries - If you do not wish to compile the gatekeeper from source, there are several pre-built packages available from here[14]. Not all versions will be made available as binaries therefore the reader will have to check what is available.

As regards the Red Hat, packages you will have to download the RPMs and enter the following command as root, substitute in the name of the file you wish downloaded.

```
$ rpm -Uvh gnugk-x.x.x.rpm
```

As regards the Debian packages, you can install the gatekeeper by using the following command as root:

```
$ apt-get install openh323gk
```

4.5.3.2. Configuration

The behaviour of the gatekeeper is completely determined by the command line options at run time and the specified configuration file. Some command line options may override settings in the configuration file. In order to avoid confusion it is common practice to keep all the configuration options in the configuration file and start the GNU GK with the following command:

```
$ [/usr/sbin/]gnugk -c /etc/gnugk.ini -o /var/log/gnugk.log -ttt
```

Here we provide a sample configuration file with the most important options for setting up basic services and their relative explanation. Note that all user-specified fields are indicated as beginning with "my" and you must customize/replace them appropriately for your site.

```
#Two lines in order to be able to telnet your GK on a specific port
#(the default is port 7000)
#(the authorization rules are detailed in the [GkStatus::Auth] section)
[Gatekeeper::Main]
Fourtytwo=42
```

[14] http://sourceforge.net/project/showfiles.php?group_id=4797

```
#name of your GK
Name=my-GnuGK

#Network information
#Specify the network interfaces of the gatekeeper
#By default the gatekeeper will detect the interfaces
#of your host automatically
Home=my-ip-address

#information about the parent GK in order to forward LRQ
#for out-of-zone calls
[RasSrv::Neighbors]
[neighbour-name]=my-ip-address:my-port;my-prefix-of-the-neighbour

#define some features on LRQ and LCF
[RasSrv::LRQFeatures]
#The gatekeeper replies with LCFs containing
#the destinationInfo and destinationType fields,
#the registered aliases and the terminal type of the destination endpoint
#The neighbor gatekeeper can then save the information
#to suppress later LRQs
#However, some vendors' gatekeepers misuse the information,
#thus resulting in interoperability problems
#set it to 0 if you encounter problems with a third-party GK
IncludeDestinationInfoInLCF=0
#Include a NonStandardParameter in LRQs
#to be compatible with Cisco gatekeepers
CiscoGKCompatible=1
#If hopCount has reached 0, the gatekeeper shall not forward the message
ForwardHopCount=10

#route mode section
[RoutedMode]
#Enable the gatekeeper routed mode, as opposed to the direct mode
GKRouted=1
#Route the H.245 control channel, only takes effect if GKRouted=1
H245Routed=1
#Some endpoints send h245Address in the UUIE of Q.931
#even when h245Tunneling is set to TRUE
#This may cause interoperability problems, avoid setting this option to 1
RemoveH245AddressOnTunneling=1
#The gatekeeper could tear down a call by sending
#RAS DisengageRequest to endpoints
#Some bad endpoints just ignore this command, with this option turned on,
#the gatekeeper will send
#Q.931 Release Complete instead of RAS DRQ to both endpoints
#to force them to drop the call
DropCallsByReleaseComplete=1
#Setting this parameter to 1 makes the gatekeeper
#to always send Release Complete to both endpoints
#before closing the call when it receives DRQ from one of the parties
SendReleaseCompleteOnDRQ=1

#Authorization rules  for telnet access to port
#(the default is port 7000)
[GkStatus::Auth]
#allow only specific addresses
rule=regex
# - we are allowing the IP addresses 192.168.1.*
regex=^(192\.168\.1\.[0-9]+)
default=forbid
#if you want to allow everybody, comment the previous lines and ...
#rule=allow
```

4.5.3.3. Operation

There are a number of ways to monitor the operation of the GNU GK. A command-line (telnet) interface is

provided, which is installed by default and allows monitoring of endpoints registrations and call requests. It also accepts unregistration commands for specific endpoints, call clearing and even reloading of the configuration file.

Having inserted the following lines in the configuration file:

```
[Gatekeeper::Main]
Fourtytwo=42
[GkStatus::Auth]
rule=allow
```

we can telnet to the GNU GK machine on the port specified in the configuration file (the default is port 7000):

```
me@mypc> telnet gnugk-ip-address 7000
```

There are a number of commands that can be issued in this telnet session: Type "help" to see a list of them. Most commands are easy and intuitive and there is no need to explain them further (for a detailed explanation see here[15]). To end the telnet session with the gatekeeper type "quit" and hit Enter.

Moreover, there are two Graphical User Interface (GUI) front-ends for the gatekeeper in order to monitor and visualize the operations.

- Java GUI: allows you to monitor the registrations and calls that go through the gatekeeper. A right-click on a button gives you a popup menu for each endpoint. This GUI works with Java 1.0 built into most web browsers. The program is available here[16].
- GkGUI: A new standalone Java program. It requires Java 1.4. The GkGUI is released under GNU General Public License, available here[17].

4.5.3.4. Endpoint authentication

The GNU Gatekeeper supports all three Radius, MySQL and LDAP backend interfaces (LDAP is still under development) for registration (RRQ) and admission (ARQ) authentication and authorization mechanisms. This is obviously a very complex as well as flexible environment in which to implement authentication and authorization methods. H.235 is supported, but more commonly ad hoc authentication methods are used, such as the IP address + alias identification method on the RADIUS server side. Special credit-time or duration restricted calling applications can be deployed on the GNU GK, assuming sufficient administrator man/hours can be spared. Please refer to [Gatekeeper::Auth] and following configuration sections on the manual web page for a more detailed configuration description of such features.

4.5.3.5. Advanced features

The GNU GK gatekeeper incorporates an excellent combination of the features of the Cisco MCM and the Radvision ECS, in a very flexible environment, being able to support hierarchies of gatekeepers (child-parent relationships) in cases where many levels of prefixes must be supported by prefix stripping or prefix substitution (please refer to the [Endpoint::RewriteE164] configuration section). Moreover the GNU GK implements resilience features such as "Alternate Gatekeeper" support (configuration available through the [Gatekeeper::Main] configuration section), where two identical GNU GK gatekeepers on two different nodes can act in tandem, providing resilience in gatekeeper services transparently to the endpoints. Since it is an open-source project, its value per cost ratio is very high, but the command-line interfaces it provides are not for the faint-hearted and if you do make the choice, be prepared to spend many hours over out-dated documentation and recompilations of new code-fixing releases.

4.6. Setting up SIP services

[15] http://www.gnugk.org/h323manual.html
[16] http://www.gnugk.org/h323gui.html
[17] http://www.gnugk.org/h323develop.html#java

4.6.1. SIP Express Router

SIP Express Router (SER) is an industrial-strength, free VoIP server based on the Session Initiation Protocol (SIP, RFC3261). It is engineered to power IP telephony infrastructures up to large scale. The server keeps track of users, sets up VoIP sessions, relays instant messages and creates space for new plug-in applications. Its proven interoperability guarantees seamless integration with components from other vendors, eliminating the risk of a single-vendor trap. It has successfully participated in various interoperability tests along with products of other leading SIP vendors.

4.6.1.1. Getting SIP Express Router

SIP Express Router is available for download from BerliOS[18]

The newest release can be found in the folder /latest.

4.6.1.2. Installation (From binary packages)

4.6.1.2.1. Supported architectures

The following architectures are supported by SER:

- Linux/i386
- Linux/armv4l
- FreeBSD/i386
- OpenBSD/i386
- Solaris/sparc64
- NetBSD/sparc64

(For other architectures the Makefiles might need to be edited) There are various configuration options defined in the `Makefile` and `Makefile.defs`.

4.6.1.2.2. Requirements

- gcc or icc : gcc >= 2.9x; >=3.1 recommended (it will work with older version but it might require some options tweaking for best performance)
- bison or yacc (Berkeley yacc)
- flex
- GNU make (on Linux this is the standard “make”, on FreeBSD and Solaris is called “gmake”)
- sed and tr (used in the make files)
- GNU tar (“gtar” on Solaris) and gzip if you want “make tar” to work.
- GNU install or BSD install (on Solaris “ginstall”) if you want “make install”, “make bin”, “make sunpkg” to work.
- “mysql” if you need MySQL support.
- “Apache (httpd)” if you want serweb support
- “PHP, MySQL-PHP” for serweb support

[18] ftp://ftp.berlios.de/pub/ser

- libmysqlclient and libz (zlib) if you want mysql support (the mysql module)
- libexpat if you want the jabber gateway support (the jabber module)

4.6.1.2.3. Install the packages:

Example:

```
/root>rpm -i ser-08.11-1.i386.rpm
```

Packages for other popular distributions are available, and can be installed using the appropriate package manager for that distribution.

On many platforms you can start the ser service using:

```
/etc/init.d/ser start
```

RedHat based systems will use:

```
/etc/rc.d/init.d/ser start
```

That will start the server with the default configuration. You can try to register your SIP user agent (for example MS Messenger) with the server, place first calls as well as send instant messages to other users registered with this server.

The default configuration is very limited, its purpose is to allow to start the server easily and to test the basic functionality. The default configuration does not include authentication, the persistence of the user location database and many other important features.

4.6.1.3. MySQL setup

To install support for a MySQL database you will need to download the package ser-mysql, which is available from the same location from which you downloaded SIP Express Router. This package contains scripts to create the required database and establish permissions for preconfigured accounts. A recent release of MySQL is recommended, you should definitely use version higher than 4.0. Earlier versions may have problems with the syntax required to set permissions in the database.

If you do not already have a copy of MySQL installed, download it from http://www.mysql.com or check out your linux distribution. Many popular linux distribution come with the MySQL server pre-packaged.

Once you have the MySQL server installed and running, execute

```
/usr/sbin/ser_mysql.sh create
```

That will create database “ser” and all the tables that are required by SER.

You can verify that the database has been created, and correct permissions assigned by using the MySQL management tool and these steps:

```
Mysql> select * from user;
| Host              | User  | Password         | Select_priv | ...
| %                 | ser   | 4e633cf914a735a0 | N           | ...
| localhost         | ser   | 4e633cf914a735a0 | Y           | ...
| %                 | serro | 7cb73a267cb7bd5f | N           | ...
```

```
| localhost          | serro | 7cb73a267cb7bd5f | Y           | ...
```

The above results show that the two users, ser and serro, have been created and granted the permissions needed to access the database. Note that in the above example the permissions have been modified to deny access to these accounts from any system(%) other than the local host.

```
mysql> connect ser;
Connection id:    294
Current database: ser

mysql> show tables;
+-----------------+
| Tables_in_ser   |
+-----------------+
| acc             |
| active_sessions |
| aliases         |
| config          |
| event           |
| grp             |
| location        |
| missed_calls    |
| pending         |
| phonebook       |
| reserved        |
| silo            |
| subscriber      |
| version         |
+-----------------+
14 rows in set (0.00 sec)

mysql> select * from subscriber;
| phplib_id                         | USERNAME | PASSWORD | FIRST_NAME | ...
| 4cefa7a4d3c8c2dbf6328520bd873a19 | admin     | heslo | first        | ...
```

The previous query shows that you have one user account defined and it has administrator privileges. Users with administrator privileges will be allowed to user the admin interface of Serweb.

We will need to add another account to be the administrator for your realm. We will do that later.

4.6.1.4. Configuration

4.6.1.4.1. Overview

This section demonstrates simple examples on how to configure the server's behaviour using the SER request routing language. All configuration scripts follow the SER language syntax, which dictates the following section ordering:

- *Global configuration parameters*: these values affect the behaviour of the server such as port number on which it will be listening, the number of spawned children processes, and log level used for the syslog.
- *Module loading*: these statements link external modules, such as transaction management (tm) or stateless UA server (sl) dynamically.

> **Note**
>
> If modules depend on each other, then the depending modules must be loaded after modules on which they depend. We recommend to load first modules tm and sl because many other modules (auth, usrloc, acc, etc.) depend on them.

- *Module-specific parameters*: determine the behaviour of modules. For example, it is possible to configure a database to be used by the the authentication module.
- One or more *route blocks*: the route blocks contain the request processing logic, which includes built-in actions as well as actions exported by modules.
- Optionally, if modules supporting reply processing (currently only tm) are loaded, one or more *failure_route blocks* containing logic triggered by received replies. Restrictions on use of actions within the failure_route blocks apply (see SER Administrator's Guide for more details).

4.6.1.4.2. Default Configuration Script

The configuration script, `ser.cfg`, is a part of every SER distribution and defines default behaviour of the server. It allows users to register with the server and have requests proxied to other users registered at the same server as well as to other SIP servers.

After performing routine checks, the script looks whether an incoming request is for the served domain (administrative domain). If this is true and the request is “REGISTER”, SER acts as a SIP registrar and updates the user location database. Optionally, it verifies user's identity first to avoid unauthorized contact manipulation.

Non-REGISTER requests for served domains are then processed using the user location database. If a contact is found for a Requested-URI, script execution proceeds to stateful forwarding, a negative 404 reply is generated otherwise. Requests targeted outside the served domain are always statefuly forwarded.

Note that the default configuration settings as set by this simple script have several limitations:

- By default, authentication is turned off to avoid dependency on MySQL. Unless it is turned on, anyone can register using any name and “hijack” someone else's calls.
- Even if authentication is turned on, there is no relationship between authentication username and address of record (see Section 2.2.2.2.3). That means, for example, that a user authenticating himself correctly with a “john.doe” id may register contacts for “gw.bush”. Site policy may wish to mandate that the authentication ID must be identical to the username claimed in the “To” header field. The auth module contains action called **check_to** that can be used to enforce such a policy.
- No dial plan is implemented. All users are supposed to be reachable via the user location database.
- The script assumes users will be using server's hostname as the domain part of the address of record. If users wish to use another name (domain name for example), this must be set using the `alias` options.
- If authentication is turned on by uncommenting related configuration options, the server will assume that the backend authentication database contains password in clear-text form (another option is storing HA1 strings for the digest authentication, but the strings must be generated for every administrative domain of the server separately).

Example 4.3. Default Configuration Script

```
#
# simple quick-start config script
#

# ----------- global configuration parameters ------------------------

debug=3         # debug level (cmd line: -dddddddddd)
fork=yes
log_stderror=no # (cmd line: -E)

/* Uncomment these lines to enter debugging mode
fork=no
```

```
log_stderror=yes
*/

check_via=no # (cmd. line: -v)
dns=no           # (cmd. line: -r)
rev_dns=no      # (cmd. line: -R)
port=5060
children=4
fifo="/tmp/ser_fifo"

# ------------------ module loading ----------------------------------

# Uncomment this if you want to use SQL database
#loadmodule "/usr/local/lib/ser/modules/mysql.so"

loadmodule "/usr/local/lib/ser/modules/sl.so"
loadmodule "/usr/local/lib/ser/modules/tm.so"
loadmodule "/usr/local/lib/ser/modules/rr.so"
loadmodule "/usr/local/lib/ser/modules/maxfwd.so"
loadmodule "/usr/local/lib/ser/modules/usrloc.so"
loadmodule "/usr/local/lib/ser/modules/registrar.so"
loadmodule "/usr/local/lib/ser/modules/textops.so"

# Uncomment this if you want digest authentication
# mysql.so must be loaded !
#loadmodule "/usr/local/lib/ser/modules/auth.so"
#loadmodule "/usr/local/lib/ser/modules/auth_db.so"

# ----------------- setting module-specific parameters ---------------

# -- usrloc params --

modparam("usrloc", "db_mode",   0)

# Uncomment this if you want to use SQL database
# for persistent storage and comment the previous line
#modparam("usrloc", "db_mode", 2)

# -- auth params --
# Uncomment if you are using auth module
#
#modparam("auth_db", "calculate_ha1", yes)
#
# If you set "calculate_ha1" parameter to yes (which true in this config),
# uncomment also the following parameter)
#
#modparam("auth_db", "password_column", "password")

# -- rr params --
# add value to ;lr param to make some broken UAs happy
modparam("rr", "enable_full_lr", 1)

# -------------------------  request routing logic -------------------

# main routing logic

route{

        # initial sanity checks -- messages with
        # max_forwards==0, or excessively long requests
        if (!mf_process_maxfwd_header("10")) {
                sl_send_reply("483","Too Many Hops");
                break;
        };
        if (msg:len >=  max_len ) {
                sl_send_reply("513", "Message too big");
                break;
        };
```

```
        # we record-route all messages -- to make sure that
        # subsequent messages will go through our proxy; that's
        # particularly good if upstream and downstream entities
        # use different transport protocol
        if (!method=="REGISTER") record_route();

        # subsequent messages withing a dialog should take the
        # path determined by record-routing
        if (loose_route()) {
                # mark routing logic in request
                append_hf("P-hint: rr-enforced\r\n");
                route(1);
                break;
        };

        if (!uri==myself) {
                # mark routing logic in request
                append_hf("P-hint: outbound\r\n");
                route(1);
                break;
        };

        # if the request is for other domain use UsrLoc
        # (in case, it does not work, use the following command
        # with proper names and addresses in it)
        if (uri==myself) {

                if (method=="REGISTER") {

# Uncomment this if you want to use digest authentication
#                 if (!www_authorize("iptel.org", "subscriber")) {
#                         www_challenge("iptel.org", "0");
#                         break;
#                 };

                        save("location");
                        break;
                };

                lookup("aliases");
                if (!uri==myself) {
                        append_hf("P-hint: outbound alias\r\n");
                        route(1);
                        break;
                };

                # native SIP destinations are handled using our USRLOC DB
                if (!lookup("location")) {
                        sl_send_reply("404", "Not Found");
                        break;
                };
        };
        append_hf("P-hint: usrloc applied\r\n");
        route(1);
}

route[1]
{
        # send it out now; use stateful forwarding as it works reliably
        # even for UDP2TCP
        if (!t_relay()) {
                sl_reply_error();
        };
}
```

4.6.1.4.3. Redirect Server

The redirect example shows how to redirect a request to multiple destinations using a 3xx reply. Redirecting requests as opposed to proxying them is essential to various scalability scenarios. Once a message is redirected, SER discards all related state information and is no more involved in subsequent SIP transactions (unless the redirection addresses point to the same server again).

The key SER actions in this example are **append_branch** and **sl_send_reply** (sl module).

The **append_branch** action adds a new item to the destination set. The destination set always includes the current URI. The **sl_send_reply** action, if passed SIP reply code 3xx, takes all values in the current destination set and adds them to the "Contact" header field in the reply.

Example 4.4. Redirect Server

```
#
# this example shows use of ser as stateless redirect server
#

# ------------------ module loading ----------------------------------

loadmodule "modules/sl/sl.so"


# -------------------------  request routing logic -------------------

# main routing logic

route{
        # for testing purposes, simply okay all REGISTERs
        if (method=="REGISTER") {
                log("REGISTER");
                sl_send_reply("200", "ok");
                break;
        };
        # rewrite current URI, which is always part of destination ser
        rewriteuri("sip:parallel@iptel.org:9");
        # append one more URI to the destination ser
        append_branch("sip:redirect@iptel.org:9");
        # redirect now
        sl_send_reply("300", "Redirect");
}
```

4.6.1.4.4. On-Reply Processing (Forward on Unavailable)

Many services depend on the status of messages relayed downstream: *forward on busy* and *forward on no reply* to name the two most well-known. To support implementation of such services, SER allows to return to request processing when forwarding of a request fails. When a request is reprocessed, new request branches may be initiated or the transaction can be completed at the discretion of the script writer.

The primitives used are **t_on_failure(r)** and **failure_route[r]{}.** If **t_on_failure** action is called before a request is statefuly forwarded and a forwarding failure occurs, SER will return to request processing in a **failure_route** block. Failures include: receipt of a SIP error (status code >= 300) from downstream and the absence of a final reply within the final response period.

The duration of the timer is governed by parameters of the tm module. `fr_timer` is the duration of the timer set for non-INVITE transactions and INVITE transactions for which no provisional response is received. If the timer hits, it indicates that a downstream server is unresponsive. `fr_inv_timer` governs time to wait for a final reply for an INVITE. It is typically longer than `fr_timer` because final reply may take a long time until the callee (finds a mobile phone in his pocket and) answers the call.

In Example 4.5, **failure_route[1]** is set to be entered on error using the **t_on_failure(1)** action. Within this reply block, SER is instructed to initiate a new branch and try to reach the called party at another destination

(sip:nonsense@iptel.org). To deal with the case when none of the alternate destinations succeed, **t_on_failure** is set again. If this case really occurs, **failure_route[2]** is entered and a last resort destination (sip:foo@iptel.org) is tried.

Example 4.5. On-Reply Processing

```
#
# example script showing both types of forking;
# incoming message is forked in parallel to
# 'nobody' and 'parallel', if no positive reply
# appears with final_response timer, nonsense
# is retried (serial forking); than, destination
# 'foo' is given last chance

# ------------------ module loading ----------------------------------

loadmodule "modules/sl/sl.so"
loadmodule "modules/tm/tm.so"

# ----------------- setting module-specific parameters ---------------

# -- tm params --
# set time for which ser will be waiting for a final response;
# fr_inv_timer sets value for INVITE transactions, fr_timer
# for all others
modparam("tm", "fr_inv_timer", 15 )
modparam("tm", "fr_timer", 10 )

# -------------------------  request routing logic -------------------

# main routing logic

route{
        # for testing purposes, simply okay all REGISTERs
        if (method=="REGISTER") {
                log("REGISTER");
                sl_send_reply("200", "ok");
                break;
        };
        # try these two destinations first in parallel; the second
        # destination is targeted to sink port -- that will make ser
        # wait until timer hits
        seturi("sip:nobody@iptel.org");
        append_branch("sip:parallel@iptel.org:9");
        # if we do not get a positive reply, continue at reply_route[1]
        t_on_failure("1");
        # forward the request to all destinations in destination set now
        t_relay();
}

failure_route[1] {
        # forwarding failed -- try again at another destination
        append_branch("sip:nonsense@iptel.org");
        log(1,"first redirection\n");
        # if this alternative destination fails too, proceed to ...
        t_on_failure("2");
        t_relay();
}

failure_route[2] {
        # try out the last resort destination
        append_branch("sip:foo@iptel.org");
        log(1, "second redirection\n");
        # we no more call t_on_negative here; if this destination
        # fails too, transaction will complete
        t_relay();
```

```
}
```

4.6.1.4.5. Accounting

In some scenarios, like termination of calls in the PSTN, SIP administrators may wish to keep track of placed calls. SER can be configured to report on completed transactions. Reports are sent by default to syslog facility. Support for RADIUS and MySQL accounting exists as well.

Note that SER is by no means call-stateful. It reports on completed transactions, i.e., after a successful call set up is reported, it drops any call-related state. When a call is terminated, a transactional state for the BYE request is created and forgotten again after the transaction completes. This is a feature and not a bug -- keeping the state information during transactions only allows to achieve significantly higher scalability. It is then up to the accounting application to correlate call initiation and termination events.

To enable call accounting, tm and acc modules need to be loaded, requests need to be processed statefuly and labeled for accounting. This means that if you want a transaction to be reported, the initial request must have taken the path "**setflag(X)**, **t_relay**" in the SER script. X must have the value configured in the `acc_flag` configuration option.

Also note, that by default only transactions that initiate a SIP dialog (typically INVITE) visit a proxy server. Subsequent transactions are exchanged directly between end-devices, do not visit proxy server and cannot be reported. To be able to report on subsequent transactions, you need to force them to visit the proxy server by turning record routing on.

Example 4.6. Configuration with Enabled Accounting

```
#
# example: accounting calls to numerical destinations
#

# ------------------ module loading ----------------------------------

loadmodule "modules/tm/tm.so"
loadmodule "modules/acc/acc.so"
loadmodule "modules/sl/sl.so"
loadmodule "modules/maxfwd/maxfwd.so"
loadmodule "modules/rr/rr.so"

# ----------------- setting module-specific parameters ---------------

# -- acc params --
# set the reporting log level
modparam("acc", "log_level", 1)
# number of flag, which will be used for accounting; if a message is
# labeled with this flag, its completion status will be reported
modparam("acc", "log_flag", 1 )

# -------------------------  request routing logic -------------------

# main routing logic

route{

        /* ********* ROUTINE CHECKS  ********************************** */

        # filter too old messages
        if (!mf_process_maxfwd_header("10")) {
                log("LOG: Too many hops\n");
                sl_send_reply("483","Too Many Hops");
                break;
```

```
        };
        if (len_gt( max_len )) {
                sl_send_reply("513", "Wow -- Message too large");
                break;
        };

    #  Process record-routing
    if (loose_route()) { t_relay(); break; };


        # labeled all transaction for accounting
        setflag(1);

        # record-route INVITES to make sure BYEs will visit our server too
        if (method=="INVITE") record_route();

        # forward the request statefuly now; (we need *stateful* forwarding,
        # because the stateful mode correlates requests with replies and
        # drops retransmissions; otherwise, we would have to report on
        # every single message received)
        if (!t_relay()) {
                sl_reply_error();
                break;
        };

}
```

4.6.1.4.6. Reporting Missed Calls

SER can report missed calls via the syslog facility or to MySQL. Mysql reporting can be utilized by SER's complementary web interface, Serweb.

Reporting of missed calls is enabled by the acc module. There are two cases, on which you want to report. The first case is when a callee is offline. The other case is when a user is online, but call establishment fails. There may be many failure reasons (call cancellation, inactive phone, busy phone, server timer, etc.), all of them leading to a negative (>=300) reply sent to the caller. The acc module can be configured to issue a missed-call report whenever a transaction completes with a negative status.

The following configuration fragment reports a missed call in both cases. The top half of the condition reports on calls missed due to offline callee status, using the **acc_request** action. The action is wrapped in transactional processing (**t_newtran**) to guarantee that reports are not duplicated on receipt of retransmissions.

The bottom half of the condition marks transactions to online users in order to be reported on failure. That is what the **setflag(3)** action is responsible for, along with the configuration option `log_missed_flag`. This option configures SER to report on all transactions, which were marked with flag 3.

```
loadmodule("modules/tm/tm.so");
loadmodule("modules/acc/acc.so");
....
# if a call is labeled using setflag(3) and is missed, it will
# be reported
...
modparam("acc", "log_missed_flag", 3 );
if (!lookup("location")) {
     # call invitations to off-line users are reported using the
     # acc_request action; to avoid duplicate reports on request
     # retransmissions, request is processed statefuly (t_newtran,
     # t_reply)
     if ((method=="INVITE" || method=="ACK") && t_newtran() ) {
          t_reply("404", "Not Found");
          acc_request("404 Not Found");
          break;
     };
     # all other requests to off-line users are simply replied
```

```
     # statelessly and no reports are issued
    sl_send_reply("404", "Not Found");
    break;
} else {
     # user on-line; report on failed transactions; mark the
     # transaction for reporting using the same number as
     # configured above; if the call is really missed, a report
     # will be issued
     setflag(3);
     # forward to user's current destination
     t_relay();
     break;
};
```

4.6.1.4.7. User Aliases

Frequently, it is desirable for a user to have multiple addresses in a domain. For example, a user with username “john.doe” wants to be reachable at a shorter address “john” or at a numerical address “12335”, so that PSTN callers with numeric-only key-pads can reach him as well.

With SER, you can maintain a special user location table and translate existing aliases to canonical usernames using the **lookup** action from the usrloc module. The following script fragment demonstrates the use of **lookup** for this purpose.

Example 4.7. Configuration of Use of Aliases

```
if (!uri==myself) { # request not for our domain...
  route(1); # go somewhere else, where outbound requests are processed
  break;
};
# the request is for our domain -- process registrations first
if (method=="REGISTER") { route(3); break; };

# look now, if there is an alias in the "aliases" table; do not care
# about return value: whether there is some or not, move ahead then
lookup("aliases");

# there may be aliases which translate to other domain and for which
# local processing is not appropriate; check again, if after the
# alias translation, the request is still for us
if (!uri==myself) { route(1); break; };

# continue with processing for our domain...
...
```

The table with aliases is updated using the serctl tool. The command serctl alias add <alias> <uri> adds a new alias, the command serctl alias show <user> prints an existing alias, and the command serctl alias rm <user> removes it.

```
[jiri@cat sip_router]$ serctl alias add 1234 sip:john.doe@foo.bar
sip:john.doe@foo.bar
200 Added to table
('1234','sip:john.doe@foo.bar') to 'aliases'
[jiri@cat sip_router]$ serctl alias add john sip:john.doe@foo.bar
sip:john.doe@foo.bar
200 Added to table
('john','sip:john.doe@foo.bar') to 'aliases'
[jiri@cat sip_router]$ serctl alias show john
<sip:john.doe@foo.bar>;q=1.00;expires=1073741811
[jiri@cat sip_router]$ serctl alias rm john
```

```
200 user (aliases, john) deleted
```

Note that the persistence of records needs to be turned on in the usrloc module. All changes to aliases would be otherwise lost on server reboot. To enable the persistence, set the `db_mode` usrloc parameter to a non-zero value.

```
# ....load module ...
loadmodule "modules/usrloc/usrloc.so"
# ... turn on persistence -- all changes to user tables are immediately
# flushed to mysql
modparam("usrloc", "db_mode",   1)
# the SQL address:
modparam("usrloc", "db_url","mysql://ser:secret@dbhost/ser")
```

4.6.1.5. Operation

4.6.1.5.1. User Management

There are two tasks related to the management of SIP users: maintaining user accounts and maintaining user contacts. Both these jobs can be done using the serctl command-line tool. The complimentary web interface, Serweb, can be used for this purpose as well.

If user authentication is turned on, which is highly advisable, user accounts must be created before users can log in. To create a new user account, use the **serctl add** utility with the username, password and email as parameters. It is important that the environment variable `SIP_DOMAIN` is set to your domain and matches the realm values used in your script. The realm value is used for calculation of credentials stored in the subscriber database, which are bound permanently to this value.

```
[jiri@cat gen_ha1]$ export SIP_DOMAIN=foo.bar
[jiri@cat gen_ha1]$ serctl add newuser secret newuser@foo.bar
MySql Password:
new user added
```

serctl can also change the user's password or remove existing accounts from the system permanently.

```
[jiri@cat gen_ha1]$ serctl passwd newuser newpassword
MySql Password:
password change succeeded
[jiri@cat gen_ha1]$ serctl rm newuser
MySql Password:
user removed
```

User contacts are typically automatically uploaded by SIP phones to the server during the registration process and administrators do not need to worry about them. However, users may wish to append permanent contacts to PSTN gateways or to locations in other administrative domains. To manipulate the contacts in such cases, use the serctl ul tool. Note that this is the only correct way to update contacts -- direct changes of the back-end MySQL database do not affect server's memory. Also note, that if persistence is turned off (usrloc “db_mode” parameter set to “0”), all contacts will be lost on server reboot. Make sure that the persistence is enabled if you add permanent contacts.

To add a new permanent contact for a user, call serctl ul add <username> <contact>. To delete all user's contacts, call serctl ul rm <username>. The command serctl ul show <username> prints all current contacts of this user.

```
[jiri@cat gen_ha1]$ serctl ul add newuser sip:666@gateway.foo.bar
sip:666@gateway.foo.bar
200 Added to table
```

```
('newuser','sip:666@gateway.foo.bar') to 'location'
[jiri@cat gen_ha1]$ serctl ul show newuser
<sip:666@gateway.foo.bar>;q=1.00;expires=1073741812
[jiri@cat gen_ha1]$ serctl ul rm newuser
200 user (location, newuser) deleted
[jiri@cat gen_ha1]$ serctl ul show newuser
404 Username newuser in table location not found
```

4.6.1.5.2. Access Control (PSTN Gateway)

It is often important to exercise some sort of access control. A typical case is when SER is used to guard a PSTN gateway. If a gateway could not be well guarded, unauthorized users would be able to use it to terminate calls to the PSTN, inflicting high costs.

There are few issues you need to understand when configuring SER for this purpose. First, if a gateway is built or configured to accept calls from anywhere, callers may easily bypass your access control server and communicate with the gateway directly. You then need to enforce at transport layer that signaling is only accepted if coming via SER and deny SIP packets coming from other hosts and port numbers. Your network must be configured not to allow forged IP addresses. Also, you need to turn on record routing to assure that all session requests will travel via SER. Otherwise, caller's devices would send subsequent SIP requests directly to your gateway, which would fail because of transport filtering.

Authorization (i.e., the process of determining who may call where) is facilitated in SER using the *group membership* concept. Scripts make decisions on whether a caller is authorized to make a call to a specific destination based on the user's membership in a group. For example, a policy may be set up to allow calls to international destinations only to users, who are members of “int” group. Before user's group membership is checked, his identity must be verified. Without cryptographic verification of the user's identity, it would be impossible to confirm that a caller really is who he claims to be.

The following script demonstrates, how to configure SER as an access control server for a PSTN gateway. The script verifies user identity using digest authentication, checks user's privileges, and forces all requests to visit the server.

Example 4.8. Script for Gateway Access Control

```
loadmodule "modules/sl/sl.so"
loadmodule "modules/tm/tm.so"
loadmodule "modules/acc/acc.so"
loadmodule "modules/rr/rr.so"
loadmodule "modules/maxfwd/maxfwd.so"
loadmodule "modules/mysql/mysql.so"
loadmodule "modules/auth/auth.so"
loadmodule "modules/auth_db/auth_db.so"
loadmodule "modules/group/group.so"
loadmodule "modules/uri/uri.so"

# ----------------- setting module-specific parameters ---------------

modparam("auth_db", "db_url","mysql:ser:heslo@localhost/ser")
modparam("auth_db", "calculate_ha1", yes)
modparam("auth_db", "password_column", "password")

# -- acc params --
modparam("acc", "log_level", 1)
# that is the flag for which we will account -- don't forget to
# set the same one :-)
modparam("acc", "log_flag", 1 )

# ------------------------  request routing logic -------------------

# main routing logic
```

```
route{

    /* ********* ROUTINE CHECKS  ********************************** */

    # filter too old messages
    if (!mf_process_maxfwd_header("10")) {
        log("LOG: Too many hops\n");
        sl_send_reply("483","Too Many Hops");
        break;
    };
    if (len_gt( max_len )) {
        sl_send_reply("513", "Wow -- Message too large");
        break;
    };

    /* ********* RR ********************************** */

    /* grant Route routing if route headers present */
    if (loose_route()) { t_relay(); break; };

    /* record-route INVITEs -- all subsequent requests must visit us */
    if (method=="INVITE") {
        record_route();
    };

# now check if it really is a PSTN destination which should be handled
# by our gateway; if not, and the request is an invitation, drop it --
# we cannot terminate it in PSTN; relay non-INVITE requests -- it may
# be for example BYEs sent by gateway to call originator
    if (!uri=~"sip:\+?[0-9]+@.*") {
        if (method=="INVITE") {
            sl_send_reply("403", "Call cannot be served here");
        } else {
            forward(uri:host, uri:port);
        };
        break;
    };

    # account completed transactions via syslog
    setflag(1);

    # free call destinations ... no authentication needed
    if ( is_user_in("Request-URI", "free-pstn")  /* free destinations */
            |  uri=~"sip:[79][0-9][0-9][0-9]@.*"  /* local PBX */
            | uri=~"sip:98[0-9][0-9][0-9][0-9]") {
        log("free call");
    } else if (src_ip==192.168.0.10) {
    # our gateway does not support digest authentication;
    # verify that a request is coming from it by source
    # address
        log("gateway-originated request");
    } else {
    # in all other cases, we need to check the request against
    # access control lists; first of all, verify request
    # originator's identity

        if (!proxy_authorize(    "gateway" /* realm */,
            "subscriber" /* table name */))  {
        proxy_challenge( "gateway" /* realm */, "0" /* no qop */ );
        break;
        };

    # authorize only for INVITEs -- RR/Contact may result in weird
    # things showing up in d-uri that would break our logic; our
    # major concern is INVITE which causes PSTN costs

        if (method=="INVITE") {

        # does the authenticated user have a permission for local
```

```
        # calls (destinations beginning with a single zero)?
        # (i.e., is he in the "local" group?)
        if (uri=~"sip:0[1-9][0-9]+@.*") {
            if (!is_user_in("credentials", "local")) {
                sl_send_reply("403", "No permission for local calls");
                break;
            };
        # the same for long-distance (destinations begin with two zeros")
        } else if (uri=~"sip:00[1-9][0-9]+@.*") {
            if (!is_user_in("credentials", "ld")) {
                sl_send_reply("403", " no permission for LD ");
                break;
            };
        # the same for international calls (three zeros)
        } else if (uri=~"sip:000[1-9][0-9]+@.*") {
            if (!is_user_in("credentials", "int")) {
                sl_send_reply("403", "International permissions needed");
                break;
            };
        # everything else (e.g., interplanetary calls) is denied
        } else {
            sl_send_reply("403", "Forbidden");
            break;
        };

        }; # INVITE to authorized PSTN

    }; # authorized PSTN

    # if you have passed through all the checks, let your call go to GW!

    rewritehostport("192.168.0.10:5060");

    # forward the request now
    if (!t_relay()) {
        sl_reply_error();
        break;
    };

}
```

Use the serctl tool to maintain group membership. The command serctl acl grant <username> <group> makes a user member of a group, the command serctl acl show <username> shows groups of which a user is member, and the command serctl acl revoke <username> [<group>] revokes a user's membership in one or all groups.

```
[jiri@cat sip_router]$ serctl acl grant john int
MySql Password:
+------+-----+---------------------+
| user | grp | last_modified       |
+------+-----+---------------------+
| john | int | 2002-12-08 02:09:20 |
+------+-----+---------------------+
```

4.6.2. Asterisk

In this section we will describe an example configuration of the Asterisk PBX. We will focus mainly on the configuration of the SIP part.

4.6.2.1. Getting Asterisk

Asterisk can be downloaded from http://www.digium.com

4.6.2.2. Installation

Download the tarball and untar it using:

```
tar xvfz asterisk-0.5.0.tar.gz
```

Compile the sources:

```
# cd asterisk-0.5.0
# make
# make install
# make samples
```

4.6.2.3. Configuration

The configuration files can be found in the /etc/asterisk directory. The most important files are: `sip.conf` which contains configuration of SIP user agent and `extensions.conf` which defines dialing plan.

In our simple example we will configure asterisk to act as a simple back-to-back user agent. It will allow SIP user agents to register to it and make calls which will be routed to an outbound proxy.

The file `sip.conf` contains the following settings:

```
;
; SIP Configuration for Asterisk
;
[general]
port = 5060                     ; Port to bind to
bindaddr = 0.0.0.0              ; Address to bind to
context = from-sip              ; Default for incoming calls
;
register => asterisk:password@iptel.org/jan     ; Register with a SIP provider

[iptel]
type=friend
username=asterisk
secret=password
fromdomain=iptel.org
host=iptel.org

[jan]
type=friend
username=jan
;secret=blah
host=dynamic
canreinvite=no
```

Section [general] contains some generic settings. We configure asterisk to listen on port 5060 and to listen on all available interfaces. We also specify context to be “from-sip”. The same context must be later configured in `extensions.conf`!

The line beginning with “register” instructs asterisk to act as a user agent and register with the iptel.org server as user “asterisk” with password “password”. The /jan part indicates that all incoming calls to user asterisk will be forwarded to user “jan” registered at the asterisk server.

Section [iptel] contains configuration of a peer -- in this case it is the iptel.org proxy server, because we will be using this server as an outbound proxy. In this section we specify the parameter “fromdomain” because we want all outgoing messages to have this domain in the “From” header field.

The last section [jan] contains credentials and data for a user that will be able to register with the asterisk server. In this case we will configure one SIP phone with username “jan” and with an empty password and with a phone

which will be registered with the asterisk server to receive calls for username “jan”.

The file `extensions.conf` contains the following settings:

```
[from-sip]
exten => jan,1,Dial(SIP/jan)
exten => jan,2,Hangup
exten => _3.,1,SetCallerID(jan)
exten => _3.,2,SetCIDName(Jan Janak)
exten => _3.,3,Dial(SIP/${EXTEN:1}@iptel)
exten => _3.,4,Playback(invalid)
exten => _3.,5,Hangup
```

The first line describes the context which we have configured in `sip.conf` already. The following lines describe the dialing plan. The “exten” directive means that the extension of the call will be processed. The first parameter after the => is the extension. If the extension starts with an underscore then it will be treated as an expression, otherwise only exact match will be accepted.

In our example, the first two lines match the extension “jan”. The rest of the lines will match any extension starting with digit 3.

The second parameter is the preference of the line in the dialing plan. The last parameter is the action to be executed when the extension matches. The first line says that calls with extension “jan” will be routed to SIP and peer “jan”.

Any extensions beginning with 3 will be routed to the iptel.org server using SIP and username “jan” will be set as the caller ID. If a call fails then asterisk will reply with an error message.

4.6.3. VOCAL

4.6.3.1. Overview

The Vovida Open Communication Application Library (VOCAL) is an open source project targeted at facilitating the adoption of Voice over IP in the marketplace. VOCAL provides the development community with software and tools needed to build Voice over IP features, applications and services. The VOCAL system is a distributed network of servers that provides Voice Over Internet Protocol (Voice over IP) telephony services. VOCAL supports devices that communicate using the Session Initiation Protocol (SIP, RFC3261[21]). VOCAL also supports analog telephones via residential gateways. VOCAL supports on-network and off-network calling. Off-network calling enables subscribers to connect to parties through either the Internet or the Public Switched Telephone Network (PSTN).

The basic software in VOCAL includes a set of SIP based servers (Redirect Server, Feature Server, Provisioning Server and Marshal Proxy Server). This is the stable development branch of the VOCAL server. Moreover, even if the following applications are not included in the current release (1.5.0), their source code remains available in the CVS archive http://www.vovida.org/cgi-bin/fom?file=556.

- SIP to MGCP translator
- Policy server
- Conference proxy server
- SIP to H.323 translator
- JTAPI feature server
- SIP User Agent (replaced by SIPset)
- SNMP/NetMgnt

[21] http://www.ietf.org/rfc/rfc3261.txt

For a more detailed overview of VOCAL system please refer to http://www.vovida.org

4.6.3.2. Installation

If you have a previous installation of VOCAL that uses the “vocalstart” executable to run, you must stop all servers with “vocalstart stop”. Note that “vocalstart” is no longer used as of version 1.4.0. In order to perform this action and to install VOCAL you must be logged in as root in your Linux system.

There are two options for downloading and installing VOCAL:

- *Installing from RPM:*

  Download vocalbin-version-x.i386.rpm from http:/www.vovida.org[24]

  Install the RPM as root, typing “rpm -U vocalbin-version-x.i386.rpm”.

- Installing from source:

  Type the following sequence of commands:

```
./configure
make
make install
```

  You must become root before executing make install.

If you want to have more information about compiling and installing VOCAL please refer to the file BUILD.txt.

4.6.3.3. Configuration

To set up a basic configuration you have to use the configuration script indicated here:

/usr/local/vocal/bin/allinoneconfigure/allinoneconfigure

Running the “allinoneconfigure” script, you will be asked a number of questions. For basic services setup answer all questions with the default answers.

After such a default configuration an Apache Web Server has been reconfigured on your Linux machine to provide basic web-based configuration (provisioning in VOCAL terms) and must be restarted for this to take effect.

In order to restart the Apache Web Server you should run:

/etc/rc.d/init.d/httpd restart

When the Apache Web Server is restarted you will be able to use the web-based provisioning of the VOCAL system. In order to start provisioning your system you will have to point a web browser to.

http://your.server.name/vocal/

You will be prompted for a password. The username is “vocal”. During configuration, you were asked to enter a password or choose to have one generated for you. If you have forgotten the password from that step, you can regenerate one by running the command:

allinoneconfigure -r

After you have run the allinoneconfigure script, make sure that your VOCAL system is running typing the following command:

[24] http://www.vovida.org

/usr/local/vocal/bin/vocalctl status

You have to make sure that you are able to see all of the necessary processes, as follows:

```
fs 5080 14957
fs 5085 14955
ms 5060 15002
...
```

If instead of such a list of actively running servers with the details of the ports they are listening to, you see “vocald is not running”, then your VOCAL system is not running because something went wrong in the configuration.

If your VOCAL system is running, you can verify your installation by running the “verifysip” command.

Passing it the -a option causes “verifysip” to create two test users, test1000 and test1001, and make a call from test1000 to test1001. After testing, “verifysip” will remove the two users. You should be able to run it with a command like this:

/usr/local/vocal/bin/verifysip -a

If your installation is OK, you should see the following text:

VOCAL basic call test passed.

4.6.3.4. Operation

VOCAL 1.5.0 comes with two provisioning systems. Both work on the same configuration data. The first, described in uses a Java application to configure the system. The second uses web-based CGI scripts, which run on the web server to configure and control VOCAL. It can be accessed through your web browser. The web-based provisioning system provides simple access to the most commonly used provisioning options for an all-in-one system. The Java-based application provides complete view to the server's features, and dial plan provisioning.

Both provisioning systems can be used on the same system, but they should not be run at the same time. The web-based provisioning system is more limited than the Java based one, thus some tasks require using the latter. For tasks other than those requiring the Java provisioning system, we recommend using the web-based provisioning system. The provisioning systems allow the system administrators to work with user and server data. Administrators can choose to Add, View, Edit and Delete Users, as well as insert details of the clients / users allowed access to the VOCAL services.

Server data can be used to configure services, as well as install more advanced features.

4.6.3.5. Endpoint authentication

The server in charge of authentication in the VOCAL system is the Marshal Proxy Server. The Marshal Proxy Server supports these authentication options:

- No authentication.
- Access control authentication - verification of IP address.
- HTTP Digest authentication - verification of username and password.

4.6.3.6. Advanced Features

There is a lot of advanced features that can be deployed on VOCAL architecture. In this section we will present an overview of these features. For a more detailed explanation please refer to the VOCAL web pages.

Each Marshal Proxy Server sends the start and stop time of a call to the Call Detail Record (CDR) Server. The

CDR Server may forward the data to 3rd party billing systems using the RADIUS accounting protocol.

Redirection of calls, integration with PSTN gateways and conferencing are configured and exploited using other special kinds of Marshal servers.

Support for both SIP to H.323 gateways and SIP to MGCP gateways is under development and the actual implementation is not yet ready to be used in a production environment, even if improvements are planned in the near future.

Dial plans are easily managed by configuring entries in the provisioning system GUI. Redirection services and backup servers, as well as more features like call forwarding and call blocking are carried out using specialized servers. Last but not least, a voice mail server is used to deliver mails with attached voice files containing message registered to the users.

4.7. Firewalls and NAT

Firewalls and *Network Address Translation* (NAT) affect IP telephony signaling protocols, making it impossible to call targets outside the private or protected network. While often firewalls and NATs go hand in hand, they impose two different problems which shall be described here.

4.7.1. Firewalls and IP telephony

Both SIP and H.323 calls use a number of different ports, out of which only the signaling ports are well defined - TCP port 1720 for H.323 and TCP port 5060 (early versions of SIP used 5060 UDP as well). To be able to place and receive calls to/from outside the protected network opening these ports is a minimal requirement.

After signaling has started, further channels are required. H.323 often uses a separate TCP connection for capability exchange (H.245), which uses dynamically assigned port numbers. Likewise the RTP media stream uses dynamically assigned port numbers on each side. The only restriction that applies to these ports is that they are in the port range > 1023.

As a result, a firewall protected IP telephony zone needs either a firewall that does not protect ports > 1023 or a firewall that is *IP telephony aware* - meaning that it monitors all SIP and H.323 messages in order to open and close the required ports on the fly. A third alternative is to deploy an H.323 or SIP proxy outside the protected zone protected by the firewall, perhaps in a DMZ, and configure the firewall to allow communication of endpoints only with this proxy. This is a mid-level security approach, as it permits the relatively safe communication between protected endpoints and a trusted proxy server outside the firewall. This document does not intend to give an overview of suitable firewall solutions, so when installing IP telephony solutions ensure that your firewall supports it.

4.7.2. NAT and IP telephony

Another problem occurs if your IP telephony zone resides in a private network (no public IP addresses). SIP and H.323 use TCP for signaling, but the messages carried in the application layer contain IP addresses that are not recognized by the NAT. In addition, the H.323 RAS channel uses UDP for transport. This combination results in the following problems:

1. *Registration:* Consider an endpoint which lies inside a private network and registers at a public server. Without being aware of the NAT it would try to register with its private IP address. Eventually the server's reply would reach the endpoint, but no call that the server tries to put through using the registered address will.

2. *Call signaling:* Some call signaling messages contain IP addresses that are necessary for subsequent communication, e.g. the IP address (and port) to which media data shall be sent. Usually an endpoint transmits its local (private) IP address causing the communication partner to fail when trying to connect to that IP address.

One possible solution is the STUN protocol, defined in RFC 3489. An endpoint implementing this protocol connects to a public STUN server to be informed of his public IP address. The result of this query is used as the IP

address to register with at a remote server. While STUN seems to be more popular in the SIP world there is no H.323 endpoint that we are aware of that supports this feature.

Another possibility is to use a router that is aware of IP telephony protocols and rewrites the IP addresses within the application layer messages, as they are routed. But this requires full decoding and encoding support for SIP and/or H.323, which is simple for the text-based SIP protocol but quite complex for the ASN.1-based H.323 protocol. In addition, it is always possible that such a router might remove all message content it does not understand, so that trying to transport new protocol features through such a router may inexplicably fail. Such an IP telephony router may come in the form of an application running on the NAT router itself - like the OpenH323Proxy™ 3.

4.7.3. SIP and NAT

4.7.3.1. Overview

Since the beginning of deploying of SIP based devices for Internet telephony there have been problems with traversing NAT. There are several reasons why SIP does not work through NAT properly:

- Addresses used for the communication and that is changed by NATs.
- From it's beginning SIP has been designed without considering NAT scenarios. That allows to design a protocol that is highly scalable but imposes many restrictions later.
- SIP is very flexible so it is hard to implement SIP support into NAT devices.
- End-to-end communication for hosts behind two different NATs is often not possible.

Many proposals for NAT traversal have been created recently, but none of them works universally and is applicable to all real world scenarios. The proposals include Connection Oriented Media, STUN, TURN, SIP ALG and so on.

All the proposals have been collected into one single document which is called ICE. ICE stands for Interactive Connectivity Establishment. It is a methodology for traversing NAT, but it is not a new protocol, it is a collection of all previously mentioned attempts to traverse NAT which works universally. The methodology is quite complex and requires mutual cooperation of all endpoints involved in the communication.

Although it works universally, such a solution is hard to implement. In this section we will describe a solution that is based on ICE, but we will make a couple of assumptions and simplifications. That will result in a significantly simpler implementation, which is supported by almost any SIP devices available today and which works in most real-world scenarios.

The price for the simplifications will be using of an RTP relay in cases where it is not absolutely necessary, but such cases seem to be quite rare. Simpler solution will also impose higher requirements on SIP devices, fortunately most manufacturers seem to have implemented all necessary features already.

The scenario as presented in following subsections assumes that the SIP proxy is in the public internet and SIP user agents are behind NAT.

4.7.3.1.1. Symmetric Signaling

One of requirements laid on SIP devices that should work behind NAT is support for symmetric signaling. Normally each SIP user agent is listening on port 5060 for incoming SIP traffic. This port number is usually advertised in the Via header field so that replies will come back on proper port and not to the port from which request was sent, and the port number will be also registered in registrar so any subsequent messages will be sent to 5060 as well.

When sending SIP messages, the user agent is allowed to use a completely different socket, which means that the source port number of the outgoing SIP messages will be different--it will be not 5060, usually it is some high port number assigned by operating system.

3http://sourceforge.net/projects/openh323proxy/

That works well when there is no NAT device along the path, but it will not work when the user agent is behind a NAT The reason is that NATs usually do not allow any traffic to the private network unless there was a packet sent from the same IP and port to the public internet. That means if a host with private IP address 192.168.0.1 wants to receive a packet on port 5060 from a host 1.2.3.4 (which is in the public internet), it has to first send a packet with source port 5060 to host 1.2.3.4. The packet will create a binding in the NAT and the NAT box will forward replies from 1.2.3.4 to 192.168.0.1. The binding will expire if there was no traffic for some interval.

Having said that, it is clear that a SIP user agent behind NAT listening on port 5060 will not be able to receive any traffic to that port unless it sends a packet through the NAT with source port 5060.

That is the principle of *symmetric signaling*. In other words it means that SIP user agents using symmetric signaling send SIP messages from the same source port on which they receive incoming SIP messages (usually 5060). Only that way they will be able to create a binding in the NAT that will allow incoming SIP traffic through.

Fortunately vast majority of developers of SIP devices seems to get this so almost all SIP devices that are available today do support symmetric signaling. That includes Cisco phones, Windows Messenger, Mitel phones, Grandstream, Snom, X-lite and kphone (from version 3.13).

Support for symmetric signaling is a must, SIP user agents not supporting that will not work behind NATs.

4.7.3.1.2. Symmetric Media

Similar problem is with getting media streams through NATs Here again, the party behind NAT must send the first media packet to create a binding and open a pinhole in the NAT box.

We have the same problem here, user agent behind NAT must send media packet with source port that is same as the port on which the user agents expects media from remote party and which was advertised in SDP. Only that way media will be able to pass the NAT in both directions.

In addition to that, the remote party (which is in the public internet) must ignore contents of SDP and send the media not to the IP and port which it received SDP, but to the IP and port from which are media from the remote party coming. That will be the public IP of the NAT box.

This approach is called *Connection Oriented Media*. It is also known as *Symmetric Media*.

It will work when one party is behind a NAT and the other party is in the public internet, in that case the party behind NAT will send the first packet and the party in the public internet will use the first packet to determine IP and port to which it should send.

When both parties are behind two different NATs, then this approach will not work. The reason is very simple, since both SIP user agents are behind NAT, both of them need to send the first media packet to open pinholes in NAT. Both of them will use the data received in SDP, but that will not work because NATs might have changed the port number.

To solve the situation, an intermediary in the public internet will be necessary -- an RTP proxy. The RTP proxy will receive all the media traffic from both parties and send it to the other side. Because it will be located in the public internet it can wait for the first media packet from both sides and send subsequent media packets to IPs and port from which it received the first packets.

4.7.3.2. Support in SIP User Agents

In order to work behind NATs SIP user agents must support symmetric signaling and symmetric media. In addition to that they should also be able to use an outbound proxy because all SIP traffic has to go through a SIP proxy in the internet.

Vast majority of SIP user agents available today can work properly behind NATs.

4.7.3.3. Support in SIP Server

Most of the burden with traversing NATs is on the SIP server in the public internet. The SIP server must do the following:

- Detect if a SIP user agent is behind NAT
- Change the contents of the SIP message if necessary.
- Force using of RTP proxy if direct communication is not possible.
- Send periodically short packets to SIP user agents behind NAT to keep the bindings open.

All the necessary NAT traversal features are implemented in the SIP Express Router available from iptel.org[26]. We will now briefly describe how the NAT traversal support works in the server.

4.7.3.3.1. Registration

When the server receives a registration, it tries to find out if the sender is behind a NAT. It does so by comparing the IP address from which the request came with IP address in Via of the sender. If they differ then the user is behind NAT and the information is saved into user location database along with his contacts.

If the previous test fails then the proxy checks if Contacts contain private IP addresses. If so then the user agent is also behind NAT and the information is saved into user location database.

For user agents behind NAT registrar rewrites IP addresses and ports in Contact header fields with the IP and port from which the REGISTER came and saves the value into the user location database.

Later, when the proxy retrieves the information from the location database, it will get the rewritten values and it will send requests correctly to the IP of the NAT box which will in turn forward the request to the host in the private network.

4.7.3.3.2. Session Invitation

Session invitation is a little bit more complicated because we want to minimize using of RTP proxy.

When the server receives an INVITE message, it tries to find out if the caller is behind NAT It, again, checks if the IP from which the request came is different from IP in the topmost Via. If they are same then IP in Contact header field is searched for a private (RFC1918) IP address.

If the caller is behind NAT then the server checks for presence of Record-Route header fields in the message. Presence of the header fields indicates that there is another proxy between us and the caller and we do not have to rewrite the contents of Contact header field because the proxy might be behind the NAT and in that case it can route private IPs properly because it is in the same network.

Otherwise we will mark the transaction the INVITE created as “behind NAT” and rewrite Contact and/or SDP The mark will be used later.

After that the proxy server does all the processing as usual. At some point the server performs user location table lookup to find out the current destination of the callee. You might remember that we saved flags about presence of NAT when processing registration. If the callee is behind NAT (the flag in user location database is set), then we will mark the transaction as “behind NAT” if it is not marked yet. After this step the transaction will be marked as “behind NAT” if either caller or callee or both are behind NAT

When the server is just about to forward the request (i.e. no other changes will be made to the request), the server will check for presence of the “behind NAT” mark in the current transaction. If the mark is set and the callee is not in the public internet or does not support symmetric media, then the proxy will send a command to RTP proxy to create a session and then force using of the RTP proxy by rewriting contents of SDP We will do the same in 200 OK when it comes.

We have mentioned that we do not force using of RTP proxy when the callee is in the public internet and does support symmetric media, but how do we know that ? Indeed, there currently is no way of finding out whether a SIP user agent support symmetric media or not. That means we do not force RTP proxy only for destination for which we know that are symmetric, like voicemail or PSTN gateway.

4.7.3.4. RTP Proxy

[26] http://iptel.org/ser

4.7.3.4.1. Overview

The RTP proxy is a very simple packet forwarder. The SIP server can talk to the RTP proxy using UNIX sockets. When the SIP proxy receives a session invitation that will require using of the RTP proxy then the SIP proxy will ask the RTP proxy for a port number that will be used to forward the media.

When the RTP proxy receives first media packets from both sides, it records IPs and port from which the packets came and starts relaying the media packets.

4.7.3.4.2. Drawbacks

Using of RTP proxy has some drawback that are worth mentioning. First of all, it introduces another hop on the path from one user agent to another and that results in increased delay. How much the delay will be increased depends on the underlying network and location of the user agents.

Secondly, using of RTP proxy imposes more burden on the server, because all the media traffic has to go through the server. For example, for G.711 is it 64 kbit/s in each direction per call. Care should be taken when building a server for RTP proxy that will receive a lot of media traffic.

4.7.3.5. Real World Setup

In this section we describe how to set up SIP Express Router and RTP proxy for NAT traversal. Note that both proxies (SIP and RTP) must be in the public Internet to make it work.

4.7.3.5.1. SIP Express Router

> **Note**
>
> This section describe only settings necessary for NAT traversal. For complete configuration guide please refer to Section 4.6.1.

First of all, it is necessary to load the nathelper module and configure its parameters. Put the following into the configuration file to load the module:

```
loadmodule "/usr/local/lib/ser/modules/nathelper.so"
```

Then set the following parameters:

```
# We will you flag 6 to mark NATed contacts
modparam("registrar", "nat_flag", 6)

# Enable NAT pinging
modparam("nathelper", "natping_interval", 60)

# Ping only contacts that are known to be
# behind NAT
modparam("nathelper", "ping_nated_only", 1)
```

The first parameter tells the registrar module which flag should be used to mark contacts behind NAT. Second parameter is interval (in seconds) for keep alive messages (to keep the NAT bindings open), and the last parameter specifies that only contacts that are behind NAT should be pinged.

To check if the sender of a message is behind NAT, we will put the following test at the beginning of the main routing section of the configuration file (right after the test for too large messages):

```
# special handling for NATed clients; first, nat test is
# executed: it looks for via!=received and RFC1918 addresses
# in Contact (may fail if line-folding used); also,
# the received test should, if complete, should check all
# vias for presence of received
```

```
if (nat_uac_test("3")) {
    # allow RR-ed requests, as these may indicate that
    # a NAT-enabled proxy takes care of it; unless it is
    # a REGISTER

    if (method == "REGISTER" || ! search("^Record-Route:")) {
        log("LOG: Someone trying to register from private IP, rewriting\n");

        # This will work only for user agents that support symmetric
        # communication. We tested quite many of them and majority is
        # smart smart enough to be symmetric. In some phones, like
        # it takes a configuration option. With Cisco 7960, it is
        # called NAT_Enable=Yes, with kphone it is called
        # "symmetric media" and "symmetric signaling". (The latter
        # not part of public released yet.)

        fix_nated_contact(); # Rewrite contact with source IP of signalling
        if (method == "INVITE") {
            fix_nated_sdp("1");  # Add direction=active to SDP
        };
        force_rport();  # Add rport parameter to topmost Via
        setflag(6);     # Mark as NATed
    };
};
```

The test does exactly what was described in previous sections.

At the end of the processing we also need to perform similar test for the callee and force using of RTP proxy if necessary. Because there can be potentially many places in an average configuration script from which we send out messages (all occurrences of the **t_relay()** or **forward()** actions), we will put the whole test into a separate **route** section and call the section instead of **t_relay()**:

```
#
# Forcing media relay if necessary
#
route[1] {
    if (uri=~"[@:](192\.168\.|10\.|172\.16)" && !search("^Route:")){
        sl_send_reply("479", "We don't forward to private IP addresses");
        break;
    };
    if (isflagset(6)) {
        force_rtp_proxy();
        t_on_reply("1");
        append_hf("P-Behind-NAT: Yes\r\n");
    };

    if (!t_relay()) {
        sl_reply_error();
        break;
    };
}


onreply_route[1] {
    if (status =~ "(183)|2[0-9][0-9]") {
        fix_nated_contact();
        force_rtp_proxy();
    };
}
```

The route section checks if flag 6 (marks NAT) is set and if it is set then it will force using of the RTP proxy. We will also setup an **onreply_route** section which will process 200 OK messages (we also need to rewrite the “Contact” header field and SDP body in the reply).

4.7.3.5.2. RTP Proxy

Installation and running of the RTP proxy is very simple and straightforward. First of all download the proxy from PortaOne site[27].

Untar the archive and compile the proxy using:

```
make -f Makefile.gnu
```

That will generate a binary called rtpproxy. Start the proxy using **./rtpproxy** and restart SER.

SER and the RTP proxy use socket `/var/run/rtpproxy.sock` to communicate.

[27] http://www.portaone.com/~sobomax/rtpproxy.tar

Chapter 5. Setting up Advanced Services

Abstract

This chapter introduces the user to the concept of setting up advanced services. There are sections for configuration and basic operations of gatewaying functions (Section 5.1) (gateways configuration, SIP to H.323 and vice-versa, H.323 to PSTN and SIP to PSTN), supplementary services (Section 5.2) and multipoint conferencing (Section 5.3).

5.1. Gatewaying

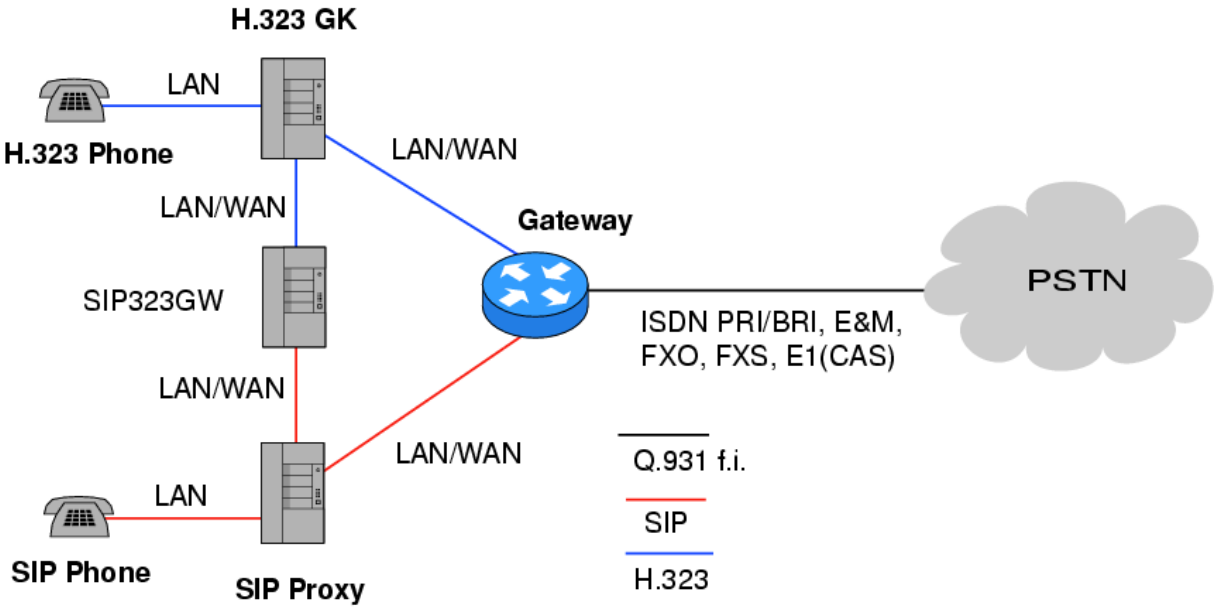
Please refer to Section 4.1 for a general architecture of SIP-H.323 and PSTN gatewaying. This section is dealing with analysis of characteristics of gateways and configuration principles for the gatewaying functions to be set up in an advanced environment. The topics detailed in this section are ranging from VoIP - PSTN gateways to SIP - H.323 gateways configuration, ending with short considerations on accounting.

5.1.1. Gateway interfaces

One of the most important interfaces in IP telephony is between a PBX and voice gateway (VoGW). It enables communication between PBX phones and IP phones (H.323 or SIP) and can also facilitate communication to PSTN, see Figure 5.1. When a PBX phone dials a number which is not another PBX extension, the PBX can forward to voice gateway either calls to numbers beginning with a specified prefix (so called access prefix that the users dials before the required number to get to IP telephony network) or all calls. Similarly, a voice gateway can forward to the PBX calls to PBX phones.

When a PBX phone dials a number in PSTN, the PBX can forward the call directly to PSTN (e.g. over ISDN) or it can forward the call to voice gateway, which forwards it to a selected PSTN operator over the Internet.

Figure 5.1. The role of PBX to voice gateway interface



There are different kinds of PBX to voice gateway interfaces with different features and cost. Your choice of the

interface type will probably depend on which features you require, acceptable cost and availability (whether there is already some interface present in PBX and voice gateway). In this section we provide some technical details on different kinds of PBX to voice gateway interfaces. We also shortly describe signaling systems, such as Channel Associated Signaling (CAS), E&M signaling methods, Q-signaling and Q.931 call control protocol and give examples of exchanged messages during correct communication.

5.1.1.1. Subscriber Loop

A subscriber loop, also called U-interface, is a 2-wire interface used primarily when connecting a telephone set to a Subscriber Line Module Analog (SLMA). SLMA is the name of the analog module in a PBX. A corresponding module in a Voice Gateway is called Foreign Exchange Station (FXS).

FXS and SLMA modules can also be interconnected to trunk modules. Trunk Modul Analog (TMA) is the name of the trunk module on the PBX side and Foreign Exchange Office (FXO) is the common name of the corresponding module in a Voice Gateway.

A subscriber loop can be used in any of the following configurations:

- Telephone to SLMA or FXS
- FXS to TMA
- FXO to SLMA

There are two operating modes:

- FXO/TMA - telephone emulation (as a common terminal equipment). This is a very simple mode, it only detects a ringing signal, provides digit dialing and switching between off-hook (to close the loop) an on-hook (to open the loop).

- FXS/SLMA - subscriber line circuit emulation. In this mode, the SLMA or FXS wait for a closed loop that will generate a current flow and a signaling tone of 425 Hz (with 10% tolerance). The Subscriber Lin Interface Circuit Emulation (SLIC) provides the functions of BORSHT (Battery, Overvoltage, Ringing, Supervision, Hybrid 2/4 wires and Testing).

The two most common methods for end-loop signaling are loop-start and ground-start signaling. DTMF (Dual Tone Multi-Frequency) is commonly used to transmit telephone number digits. DTMF tones identify numbers 0 through 9 and the * and # symbols. Digits are represented by a particular combination of two frequencies from the high group and the low group. Each group includes only four frequencies. Out of 16 possible combinations 12 are used on the keypad. DDI (Direct Dialing In) is possible only through a DTMF suffix, that is during the connection time when the caller normally already pays for the connection.

5.1.1.2. E&M interfaces

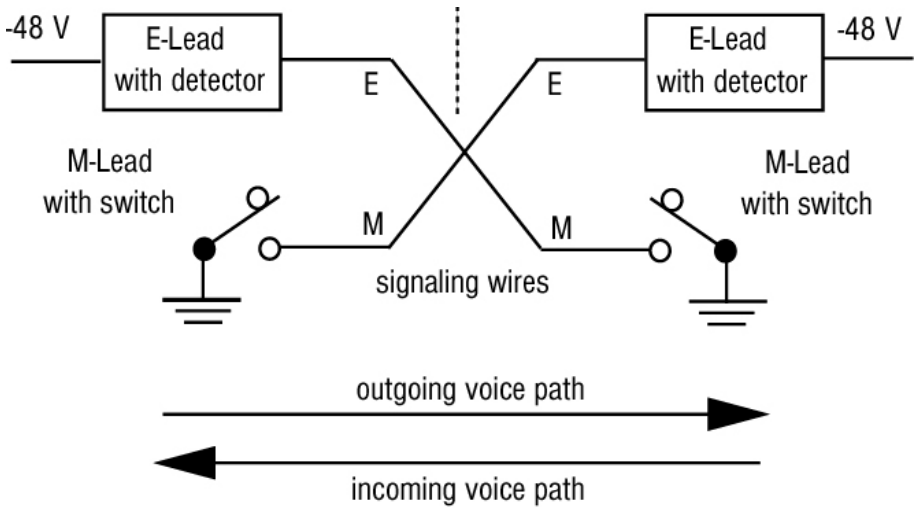
E&M is commonly alternately explained as Ear and Mouth or recEive and transMit. E&M interfaces allow DDI without restrictions before the conversation starts. There are several different types of E&M interfaces according to signalling and number of interconnecting wires. Type V (see Figure 5.2) is very popular in Europe. In the commonly used 6-wire interconnection the individual wires are used as follows:

- one pair of wires (wires E and M) is used for signaling
- one pair of wires is used for outgoing voice path
- one pair of wires is used for incoming voice path

This 6-wires connection can be reduced into 4-wires:

- one pair of wires (wires E and M) is used for signaling
- one pair of wires is used for voice path in both directions (which can cause a problem with echo cancellation and inhibits a possibility to use an amplifier)

Figure 5.2. E&M signaling, type V



Signaling is carried out with direct current via the E and M control wires for call set-up and tear down, pulse dialing and remote blocking. DTMF signals can alternatively be used for dialing. The E&M signaling can operate in several modes:

- continuous signal
- wink start signal
- delay dial
- immediate dial

5.1.1.3. E1/CAS trunk

CAS (Channel Associated Signaling) exists in many varieties that operate over analog or digital interfaces. A common digital interface with CAS signaling is called E1 (European version). The physical layer is working in accordance with the ITU recommendation G.703/G.704 for PCM30/32. The endpoints continually send Backward and Forward marks in 16 TSLs (Timeslot of PCM30/32, bits ABCD) as a supervision signal to indicate various states of the connection. Additionally, the MFC-R2 (Multi Frequency Compelled) signaling is used (in TSL 1-15 and TSL 17-31) to support for several features:

- malicious call tracing (used to transfer calling party numbers)
- override authorization
- free calls
- called party hold

5.1.1.4. ISDN Access Interfaces

ISDN (Integrated Service Digital Network) is a prefered current method of PBX to VoGW interconnection. We will describe it in more detail and give some illustrative examples of exchange of its Q.931 signalling messages. ISDN is a system of digital connections allowing to establish a call with end-to-end digital connectivity of nx64kbps. Original recommendations of ISDN were in CCITT Recommendation I.120 (1984), which described some initial guidelines for implementing ISDN. The first commercial implementation of ISDN was in PBX Hicom300 (Siemens AG). Several different signalling protocols have emerged. It was 1TR6 protocol in Germany, NI (National ISDN) in the USA and French National ISDN VN3 protocol in France. The absence of an international standard, led each European country to make its own version of ISDN, which meant incompatibility and increased costs. 26 communication organizations signed in 1992 the "Memorandum of Understanding on the

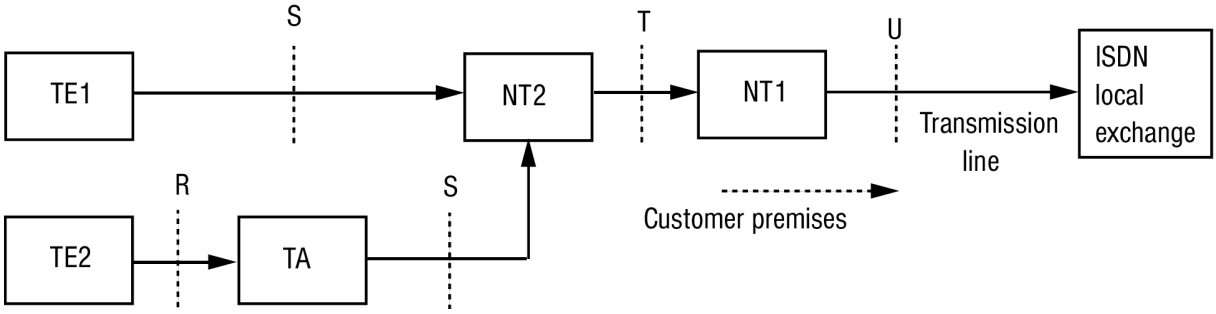
implementation of a European ISDN". The signing countries were obliged to offer a common technological substructure for the ISDN network development, connecting all Europe. As a result Q.931 signalling has been internationally standardized.

Two types of access methods exist for ISDN:

- BRI (Basic Rate Interface)
- PRI (Primary Rate Interface)

BRI delivers two 64 kbps B channels and one 16 kbps D channel. The reference configuration of ISDN defined in the ITU specification I.411 is illustrated in Figure 5.3

Figure 5.3. ISDN configuration



- Network Termination NT (divided into NT1 and NT2; NT1 is working in Layer 1 and NT2 in Layers 2 and 3, NT1 and NT2 are connected by a four-wire T interface)
- Terminal Equipment TE1 is ISDN compatible device (TE1 is connected to NT2 via a four-wire S interface)
- Terminal Equipment TE2 is a non-ISDN compatible device that requires terminal adapter interconnection
- Terminal Adapter provides an ISDN-compliant interface to NT and a standard interface to TE2 (such as RS-232, USB, X.21, etc.)

As a PBX can provide NT2 functions, the T interface is commonly used for interconnection of a PBX and a Voice Gateway. The PBX is working in the user-side operation mode and the Voice Gateway in the network-side operation mode.

5.1.1.4.1. Q.931

The L2 and L3 interface of ISDN is also referred to as the Digital Subscriber Signaling System No.1 (DSS1). The L2 protocol of ISDN is ITU Q.920/Q. 921 and the L3 protocol is ITU Q.930/Q.931. Q.932 enables general procedures for accessing and controlling supplementary services.

Q.931 provides call control capabilities. Some of the most important Q.931 messages are:

- Setup
- Setup acknowledge
- Call proceeding
- Progress
- Alerting
- Connect
- Connect acknowledge
- Disconnect
- Release complete
- Information

The destination digits can be sent in two forms during call-setup:

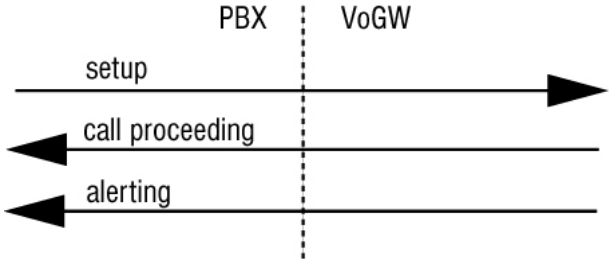
- complete called party number in the SETUP message, also known as the en-bloc signal

- one by one in separate messages, also known as the overlap signal

An example of Q.931 Call control messages in call-setup with the en-bloc signal is shown in Figure 5.4. This example corresponds to the following setup:

- Cisco AS5300 was used as a Voice Gateway, debugging was enabled with "debug isdn q931" command
- The call was initiated from Technical University in Ostrava to Czech Technical University in Prague, PBX was connected through ISDN/PRI, called number was sent as the en-bloc in the SETUP message
- Calling Party Number was 596991699
- Called Party Number was 224352979

Figure 5.4. Q.931 call control messages in call-setup with the en-bloc signal

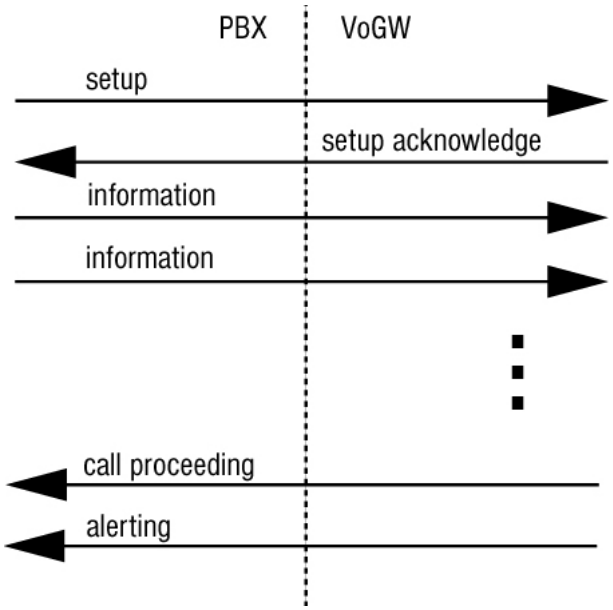



```
Jun 24 18:30:12.817: ISDN Se0:15: RX <- SETUP pd = 8  callref = 0x0002
        Sending Complete
        Bearer Capability i = 0x8090A3
        Channel ID i = 0xA9838E
        Calling Party Number i = 0x00, 0x83, '596991699', Plan:Unknown, ...
        Called Party Number i = 0x80, '224352979', Plan:Unknown, Type:U...
        High Layer Compat i = 0x9181
Jun 24 18:30:12.837: ISDN Se0:15: TX -> CALL_PROC pd = 8  callref = 0x8002
        Channel ID i = 0xA9838E
Jun 24 18:30:13.129: ISDN Se0:15: TX -> ALERTING pd = 8  callref = 0x8002
        Progress Ind i = 0x8188 - In-band info or appropriate now available
        Progress Ind i = 0x8182 - Destination address is non-ISDN
```

An example of Q.931 Call control messages in call-setup with the overlap signal is shown in Figure 5.5. This example corresponds to the following setup:

- Cisco AS5300 was used as a Voice Gateway, debugging was enabled with "debug isdn q931" command
- The call was initiated from Czech Technical University in Prague to PSTN (Public Switched Telephone Network), PBX in Prague was connected through ISDN/PRI, called number was sent as the digit by digit (overlap) in the SETUP and INFOMATION messages
- Calling Party Number is 224355406
- Called Party Number is 224324997

Figure 5.5. Q.931 call control messages in call-setup with overlap




```
Jun 24 18:31:43.092: ISDN Se1:15: RX <- SETUP pd = 8  callref = 0x540A
        Bearer Capability i = 0x8090A3
        Channel ID i = 0xA1838E
        Progress Ind i = 0x8183 - Origination address is non-ISDN
        Calling Party Number i = 0x31, 0x81, '224355406', Plan:ISDN, Type:.
        Called Party Number i = 0x81, '2', Plan:ISDN, Type:Unknown
Jun 24 18:31:43.104: ISDN Se1:15: TX -> SETUP_ACK pd = 8  callref = 0xD40A
        Channel ID i = 0xA9838E
Jun 24 18:31:43.808: ISDN Se1:15: RX <- INFORMATION pd = 8  callref = 0x540A
        Called Party Number i = 0x81, '2', Plan:ISDN, Type:Unknown
Jun 24 18:31:45.152: ISDN Se1:15: RX <- INFORMATION pd = 8  callref = 0x540A
        Called Party Number i = 0x81, '4', Plan:ISDN, Type:Unknown
Jun 24 18:31:46.536: ISDN Se1:15: RX <- INFORMATION pd = 8  callref = 0x540A
        Called Party Number i = 0x81, '3', Plan:ISDN, Type:Unknown
Jun 24 18:31:47.564: ISDN Se1:15: RX <- INFORMATION pd = 8  callref = 0x540A
        Called Party Number i = 0x81, '2', Plan:ISDN, Type:Unknown
Jun 24 18:31:48.896: ISDN Se1:15: RX <- INFORMATION pd = 8  callref = 0x540A
        Called Party Number i = 0x81, '4', Plan:ISDN, Type:Unknown
Jun 24 18:31:51.012: ISDN Se1:15: RX <- INFORMATION pd = 8  callref = 0x540A
        Called Party Number i = 0x81, '9', Plan:ISDN, Type:Unknown
Jun 24 18:31:52.696: ISDN Se1:15: RX <- INFORMATION pd = 8  callref = 0x540A
        Called Party Number i = 0x81, '9', Plan:ISDN, Type:Unknown
Jun 24 18:31:54.480: ISDN Se1:15: RX <- INFORMATION pd = 8  callref = 0x540A
        Called Party Number i = 0x81, '7', Plan:ISDN, Type:Unknown
Jun 24 18:31:54.604: ISDN Se1:15: TX -> CALL_PROC pd = 8  callref = 0xD40A
Jun 24 18:31:55.684: ISDN Se1:15: TX -> ALERTING pd = 8  callref = 0xD40A
        Progress Ind i = 0x8288 - In-band info or appropriate now available
```

5.1.2. Gatewaying from H.323 to PSTN/ISDN

One of the most useful H.323 services, is the ability of VoIP callers (H.323 world) to reach the PSTN (classic telephony world). This service is provided by H.323/PSTN gateways and the functionality they provide is:

- forwarding of incoming calls from PSTN and call signalling to H.323
- termination of incoming calls from H.323 and forwarding to PSTN
- accounting for calls utilizing the gateway

- optional support for H.320 (ISDN) capable conferencing endpoints

This section introduces the basic principles on how to perform gatewaying using H.323; basic configuration guidelines and operational principles of commercial and open source gatekeepers are described here in order to detail how to set up gatekeepers for being interconnected with PSTN. Moreover details on gateways configuration are given in order to give guidelines on how to configure gateways to be part of a H.323 network.

This section introduces the basic principles on how to perform gatewaying using H.323. Basic configuration guidelines and operational principles of commercial and open source gatekeepers are described here in order to detail how to set up gatekeepers for being interconnected with PSTN.

5.1.2.1. Using a RADVISION OnLAN 323 L2W-323 Gateway

The Radvision OnLAN 323 L2W-323 Gateway is a hardware-based H.323 to H.320 gateway, which allows H.323 endpoints to reach destinations on the PSTN or specialized H.320 (ISDN-based) endpoints and vice versa. The L2W-323, in its most common configuration, has four Ethernet (LAN) interfaces and two ISDN BRI (WAN) interfaces. It is designed for stand-alone use and thus integrates the functions of a gatekeeper, as well as a gateway, under the same hood. It is presently not available for sale but it has quite a large installed base, considering that the Cisco 3520 is essentially the same product marketed by Cisco.

5.1.2.1.1. Installation

Its installation is straight-forward, requiring merely power, a network interface, BRI ISDN interfaces and a PC on the same LAN for setting-up initial configuration parameters through a windows-based application. To manage the device, install the RADVision OnLAN Tools from the installation disks, by running the **setup.exe** program on the PC. Since the gateway has no network configuration to begin with, the PC running the software will have to be on the same LAN in order to perform initial network setup of the unit. After specifying an IP address for the Ethernet interface of the unit, the configuration application can then be run remotely.

5.1.2.1.2. Configuration

When running the configuration application, you will need to specify the IP address of the target device, or choose one from the list of detected devices on the same LAN. After entering the administrative password, you are presented with a window where you specify which configuration to edit. The options are to edit the currently loaded configuration of the device, or a previously saved-to-file configuration. The next step is to select which functionality to configure: the gatekeeper (select "Gatekeeper Setup") or the gateway (select "Unit Setup"). We are interested in the second option only, since the gateway can register with any of the gatekeepers detailed above, and the built-in gatekeeper can be turned off since it provides only basic functionality. Select "Unit Setup" and proceed with "Unit Identification" and "Date/Time" options, which are informational only.

The next screen, "Miscellaneous Parameters", presents a number of configuration options. The critical settings are the ones for "Default Gatekeeper" and for "Default Router IP", which you must set to the IP address of the gatekeeper controlling your zone, and the IP address of the router (network gateway in the IP protocol sense). The rest of the settings can be left at their default state.

The next screen, "LAN Port Settings", is responsible for configuring the Ethernet interfaces. There are four screens, one for each of the four ethernet ports. Configuring just one interface with an "IP Address" and "IP Mask" is sufficient. A summary screen with settings for all four ports follows.

The next screen, "Services Definition Table", is the most important one, since it defines the services that the gateway will make available to callers, by registering them with its controlling gatekeeper. Each service definition includes information about the "Prefix" it is called with, the "Call Type" which can be voice-only or H.320 (voice and video), and the "Maximum Bit Rate" which a call of this type will require. By defining different services, the callers are given the option to choose the type of call they can make, based on the service prefix they dial, assuming the prefixes are well known to the users, or at least that the gatekeepers have pre-selected default gateway services. For example, if the gateway defines a voice only service with a prefix of e.g. 9, the administrator of the gatekeeper will likely make this the default service to route all calls to the gateway. Any user requiring a voice and video call, will have to know the service prefix, e.g. 8, for that special type of call. The L2W gateway will already have template services defined, but you can add, edit or delete services as needed. There is certainly one thing you will need to customize and that is the service prefixes in order to make them match your local dialling scheme.

The next screens refer to "WAN Port Settings" which for an ISDN interface present relative settings. Country-specific ISDN protocol must be selected and the "Phone Number" connected to the ISDN port can be indicated. Also, specific services can be enabled or disabled for this WAN port, assuming more than one type of WAN ports are available, each with distinct capabilities. A summary screen with settings for all WAN ports follows.

At the end of the configuration wizard, you are given the option to save the new configuration settings in a file, before downloading them to the gateway itself. At this point, the gateway is reloaded and the new settings are applied.

5.1.2.1.3. Operation

Immediately after configuration and reload, the gateway attempts to register its services with the gatekeeper. This must be verified on the gatekeeper side, before attempting to use the gateway's services. Specifically, there are two things that must be checked: a) the registration of the gateway on the gatekeeper b) the registration of its service prefixes on the gatekeeper. For example on the Cisco MCM gatekeeper, the appropriate commands would be:

```
> show gatekeeper endpoints
                    GATEKEEPER ENDPOINT REGISTRATION
                    ================================
CallSignalAddr  Port  RASSignalAddr   Port  Zone Name        Type    Flags
194.257.8.150   1820  194.257.8.150   1024  gk.mydomain.org  CH320-GW
    H323-ID: GW4134522623
```

At least one entry should refer to the registration of the gateway, indicating its IP address, the type of registration (H320-GW) and the H.323 alias of the unit (GW4134522623).

```
> show gatekeeper gw-type-prefixes
GATEWAY TYPE PREFIX TABLE
=========================
Prefix: 92*
  Zone gk.mydomain.org master gateway list:
    194.257.8.150:1820 GW4134522623

Prefix: 93*
  Zone gk.mydomain.org master gateway list:
    194.257.8.150:1820 GW4134522623
```

A listing of all registered services at the gatekeeper should include an entry for each of the services defined on the gateway, indicating the prefix, and the IP address and H.323 alias of the gateway to forward calls to. If registration of services with the gatekeeper is confirmed, the gateway is ready to service calls, which can be traced through the gatekeeper tools.

The gateway can be monitored by a command-line interface only:

```
> telnet 194.257.8.150
Trying 194.257.8.150...
Connected to 194.257.8.150.
Escape character is '^]'.
VxWorks login: admin
Password:
->
```

At this prompt, no command can be entered to explicit logs, but passive monitoring of events is provided. Logs on this interface can be overwhelming, but the L2W gateway does not provide any other means of debugging, or monitoring. Please note that in the current release a bug makes the L2W gateway unregistering from the gatekeeper after some time, making its services unavailable to the zone. Check often for registration of gateway services and in case they are lost, reboot the gateway to force gatekeeper registration again. The most flexible way

to reboot the gateway is to use the telnet interface, login, and apply a Control-X command.

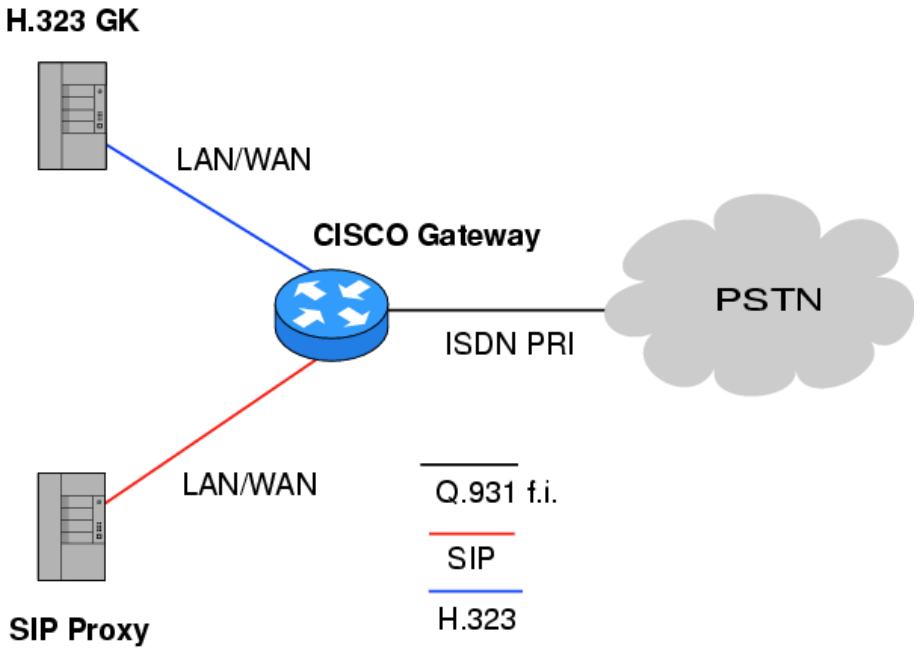
5.1.2.1.4. Authentication

In environments where gatekeeper registration is authenticated, the gateway has to provide authentication credentials in order for the gatekeeper to allow its registration. If H.235 authentication is required by the gatekeeper, the L2W gateway does not support it and administrative measures must be taken at the gatekeeper side to exempt it from authentication. If authentication by H.323 alias and password is required, e.g. for the Cisco MCM piggyback mechanism, the L2W gateway has a fixed H.323 alias that it tries to register with the gatekeeper (of the format GWnnnnnnnnnn, where n is a number generated by the gateway itself and it is not configurable). The administrator must make sure that this specific H.323 alias is allowed to register on the gatekeeper with no password. Similar measures must be taken in case an IP-address plus H.323 alias authentication method is applied on the gatekeeper side.

5.1.2.2. Gatewaying H.323 using CISCO

In this subsection we will present an example how a CISCO Voice Gateway should be configured for interconnection with a PBX. See Figure 5.6 for illustration of interconnection. The same procedure could be used for interconnecting CISCO Voice Gateway with PSTN/ISDN

Figure 5.6. CISCO voice gateway interconnection



Gateway configuration can be divided into three main parts:

- Protocol configuration
- Interface configuration
- Dial-peer configuration

The example corresponds to the following setup. As a Voice Gateway is used Cisco 2651 connected through PRI/ISDN to the PBX and is registered to a Gatekeeper. Configuration examples were cut off from Cisco CLI command show running-config output and commented.

5.1.2.2.1. Protocol configuration

Protocol specific part of configuration is not necessary the first. Gatekeeper peering and gateway id are set up in

this section. Cisco gateway could serve as voice gateway for both H.323 and SIP protocols at one time, but SIP protocol specific part is configured in completely different configuration section of the gateway (not sip-gateway, see Section 5.2.2). It is recommended to use h323-gateway command set in loopback interface configuration section.

```
h323-gateway voip interface
h323-gateway voip id GK1-CESNET2 ipaddr 195.113.113.131 1718 priority 100
h323-gateway voip id GK2-CESNET2 ipaddr 195.113.144.85 1718
h323-gateway voip h323-id VoGW-ZCU
h323-gateway voip tech-prefix 1#
```

- Voice gateway is registered to Gatekeeper GK1-CESNET2
- A second Gatekeeper GK2-CESNET2 is used as backup
- The h323-id of the Voice Gateway is VoGW-ZCU, it is checked on the Gatekeeper and is needed for successful registration

5.1.2.2.2. Interface configuration

Interconnection to PBX or PSTN has to be also properly set-up at interface level (router(config-if)#). Telephony interface configuration is independent on the used VoIP signaling protocol. In this section are configured ISDN parameters for signaling channel (sixteenth channel has number 15 by CISCO).

```
interface Serial0/0:15
  isdn switch-type primary-net5
  isdn overlap-receiving T302 5000
  isdn protocol-emulate network
  isdn send-alerting
  isdn sending-complete
  isdn outgoing-voice info-transfer-capability 3.1kHz-audio
```

- primary-net5 is the setting of DSS1 (EuroISDN) signaling protocol on the primary interface PRI (basic-net3 is used on the BRI)
- timer T302 is interval in milliseconds for overlap mode (the interface waits 5 seconds for possible additional call-control information)
- "isdn protocol-emulate network" is the configuration of the Layer2 and Layer3 of the ISDN protocol, the Voice Gateway is working as NT, the PBX is in the slave mode
- "isdn send-alerting" causes the Alerting message to be is sent out before the Connect message
- "isdn sending-complete" is an optional enhancement, where the sending complete information element is required in the outgoing call setup message the last command specifies the transfer capability for voice calls

5.1.2.2.3. Dial-peer configuration

Next configuration step is setting up the call rules. This is done by dial-peer command set from basic config mode (router(config)#) of the CISCO gateway. If the gateway should provide calls from both sides (to and from PSTN/PBX) set of minimum two dial-peers has to be configured.

```
dial-peer voice 1 pots
  destination-pattern 42037763....
  progress_ind alert enable 8
  direct-inward-dial
```

```
  port 0/0:15
```

These commands specify rules for calls to PSTN(PBX) from VoIP side.

- the called number prefix is 42037763 and must be followed with four digits of extension (four dots substitution pattern). Best match to destination pattern chooses the right dial peer.
- "progress_ind alert enable 8" is the transcription of the Progress element in the Alerting message, this transcription causes B-channels interconnection and allows to resolve a possible problem with ringback tone
- the rules are written into port interface 0/0:15 with DDI (Direct Dialing In)

```
dial-peer voice 112 voip
  destination-pattern 420.........
  session target ras
  no vad
```

These commands specify rules for calls leading to VoIP side from PSTN(PBX).

- the called number prefix is 420, it must be followed with nine digits
- the target of this session is the Gatekeeper accessible through RAS signaling
- the last command specifies that the Voice Activity Detection is not possible (higher voice quality), default configuration is VAD with CNG function (Comfort Noise Generation)

User is reminded that here listed configuration may not be sufficient to run voice gateway on CISCO routers. Commands may depend on the type of the router and used IOS version.

5.1.2.3. Gatewaying H.323 using GNU Gatekeeper

How to set up services using GNU Gatekeeper was already described in Section 4.5.3, here we describe enhanced configuration and operation for GnuGK to be used with a gateway in order to reach people who are using ordinary telephones.

The gatekeeper has to know which calls are supposed to be routed over the gateway and what numbers shall be called directly. Using the [RasSrv::GWPrefixes] section of the config file the administrator tell the gatekeeper the prefix of numbers that shall be routed over the gateway.

```
[RasSrv::GWPrefixes]
gw-PSTN=0
gw-MOBILE=3
```

These entries tell the gatekeeper to route all calls to E.164 numbers starting with 0 to the gateway that has registered with the H.323 alias "gw-PSTN" and all calls to E.164 numbers starting with 3 to the gateway that has registered with the H.323 alias "gw-MOBILE". If there is no registered gateway with those alias the call will fail. Note that you must use the gateway alias - you can't just tell the gatekeeper the IP number of the gateway. Static configuration of gateway-ip-address/prefix is in principle possible using the [RasSrv::PermanentEndpoints] configuration section but we do not advice such a solution because it leads to errors when a network reconfiguration happens.

When using a gateway you often have to use different numbers internally and rewrite them before sending them over a gateway into the telephone network. You can use the RasSrv::RewriteE164 section to configure that.

```
[RasSrv::RewriteE164]
```

```
12345=08765
```

In these example we configured the gatekeeper to replace the numbers 12345 at the beginning of the E.164 number dialed to 08765 (Example: 12345-99 is rewritten to 08765-99). Please refer to "rewrite"[1] for the section syntax.

A gateway can register its prefixes with the gatekeeper by containing supportedPrefixes in the terminalType field of RRQ. The following option defines whether to accept the specified prefixes of a gateway.

```
AcceptGatewayPrefixes=1
#Default: 1
```

5.1.3. Gatewaying from SIP to PSTN/ISDN

5.1.3.1. Gatewaying SIP using CISCO

Configuration of SIP gateway is almost same as configuration of H.323 gateway and because H.323 gateway is already described in Section 5.1.2.2, we will not describe the same settings here again (for example interface configuration which is exactly the same). We will describe only configuration specific to SIP. Readers should read Section 5.1.2.2 first.

Dial-peer configuration is almost same as described in Section 5.1.2.2, the only difference is that “ras” will be replaced by “sip-server”. Example:

```
dial-peer voice 112 voip
    destination-pattern 420.........
    session target sip-server
    no vad
```

5.1.3.2. sip-ua Configuration

All the SIP parameters are configured in sip-ua section. An example configuration might look like:

```
sip-ua
 nat symmetric role passive
 nat symmetric check-media-src
 retry invite 4
 retry response 3
 retry bye 2
 retry cancel 2
 sip-server dns:iptel.org
```

The first parameters configures the gateway to be passive. That is good for Connection Oriented Media. Value passive means that if there is a direction=active parameter in SDP then the gateway will wait with sending of media until it receives first media packet from the remote party. This feature can be very useful for NAT traversal. It can be enable only when the gateway is in the public Internet.

The second line configures the gateway to check for the source of incoming media and send it's media there if it is in symmetric mode. This option is related to the previous one.

Retry parameters specify how many times various SIP messages should be retried.

And the last parameter specifies how to reach the SIP proxy. In this case the gateway will send all the SIP traffic to iptel.org and it will use dns to resolve it to IP

[1] http://www.gnugk.org/h323manual.html#rewrite

5.1.4. Gatewaying from SIP to H.323 and vice versa

SIP to H.323 gatewaying and vice versa is a complex matter since, up to now, only basic translations of services are possible using this gatewaying. A great development effort in this topic is useless since the gatewaying makes the users loosing the native protocol (H.323 / SIP) supplementary services. These kind of gateways, even if valuable for connecting SIP and H.323 worlds are not yet widely deployed because the adoption of proprietary protocols provides the users with more value added services. Anyway, for sake of thoroughness, this section deals with operations, descriptions and simple configuration of SIP-H.323 gateways.

A SIP / H.323 gateway allows users to make an audio call from SIP network to H.323 network and vice-versa. The entities making the calls using such a gateway can be:

- a SIP user agent;
- a H.323 terminal;
- a SIP proxy server;
- a H.323 gatekeeper;
- another gateway (e.g., SIP-PSTN).

Various types of operations can be performed using a SIP / H.323 gateway and can basically divided in five categories (detailed described in the following of this section):

- user registration;
- call from SIP to H.323;
- call from H.323 to SIP;
- media switching and capabilities negotiation;
- call termination.

There are different modes in which a SIP / H.323 gateway can operate; in this section we describe basic operation modes (for more detailed operations please refer to Interworking Between SIP/SDP and H.323[2]) in order to give the reader basic functionalities examples as well as configuration guidelines (actual configurations are relative to specific hardware/software and are out of the scope of this document).

A SIP / H.323 gateway may have a built-in H.323 gatekeeper or a built-in SIP registrar (optional presence or activation of such entities are dependent on software implementation choices), different operations are performed by the SIP / H.323 gatekeeper depending if the internal gatekeeper / internal SIP register are activated or not.

If a SIP / H.323 gateway is configured appropriately it should try to register both with a H.323 gatekeeper using RAS procedures and with a SIP registrar in order to perform gateway's address resolution from either side. A SIP entity can query the registrar, whereas a H.323 entity can query the H.323 gatekeeper to locate SIP / H.323 gateway. If internal gatekeeper is activated then no registration to external gatekeeper should be performed. If internal SIP registrar is activated then no registration to external SIP registrar should be performed. Anyway, at least one H.323 gatekeeper and at least one SIP registrar should be always specified / activated for operations to be successful.

If H.323 gatekeeper is not specified then a broadcast GRQ, Gatekeeper ReQuest, message should be sent to discover it, once the H.323 gatekeeper address is known a RRQ, Registration ReQuest, message is used to register to it; the SIP / H.323 gateway alias should be inserted in such a message. If no built-in SIP registrar is used then an external SIP registrar address should always be specified (SIP REGISTER message should always contain the destination address). The contact address inserted in the REGISTER message should be that of the SIP / H.323 gateway itself.

5.1.4.1. User Registration

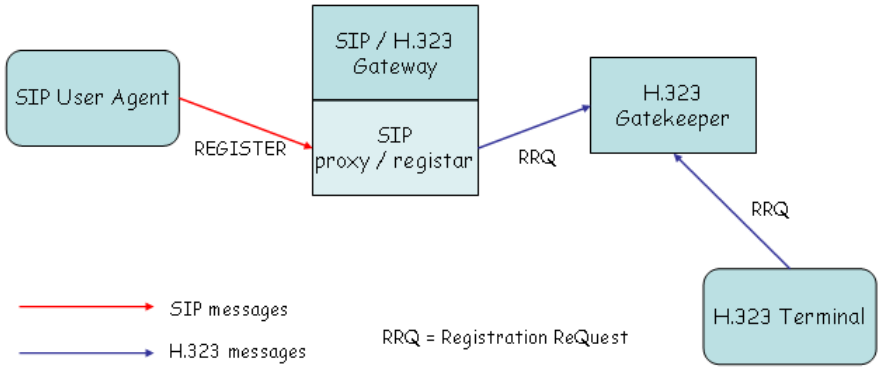
In this section, we give details on different architectures for user registration / address resolution. User registration servers are SIP registrars and H.323 gatekeepers. If SIP / H.323 gateway has direct access (either built-in or configured) to user registration servers this simplifies locating users independent of the signaling protocol. The user registration server forwards the registration information from one network, to which it belongs, to the other.

Depending on the internal architecture of the SIP / H.323 Gateway we can distinguish three different cases:

[2] http://www.cs.columbia.edu/~kns10/publication/iptel2000.pdf

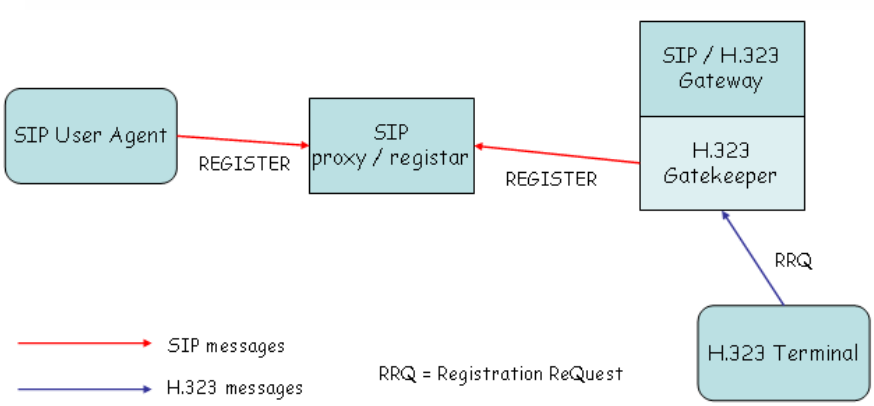
- SIP / H.323 gateway contains SIP proxy and registrar (Figure 5.7). The H.323 gatekeeper maintains the registration information and thus H.323 users register via the usual procedure. On the other hand, when a SIP REGISTER request is received by the SIP registrar server, it generates a registration request (RRQ) to the H.323 gatekeeper, the RRQ contains the translation of a SIP URI into H.323 Alias Address.

Figure 5.7. SIP / H.323 gateway containing SIP proxy and registrar



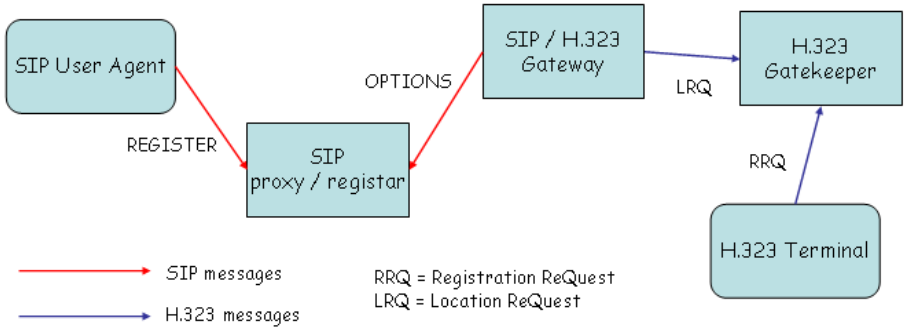
- SIP / H.323 gateway contains an H.323 gatekeeper (Figure 5.8). The user registration information from both networks is maintained by the SIP proxy server. The SIP registrar receives the forwarding of any H.323 registration request received by the H.323 gatekeeper; the SIP server stores the user registration information of both the SIP and H.323 entities.

Figure 5.8. SIP / H.323 gateway containing a H.323 gatekeeper



- SIP / H.323 gateway is independent of proxy or gatekeeper (Figure 5.9). User registration is done independently in the SIP and H.323 networks.

Figure 5.9. SIP / H.323 gateway independent



5.1.4.2. Call from SIP to H.323

We are going to describe the SIP / H.323 gateway operations only when it acts an active intervention. Since, when the SIP / H.323 gateway contains a SIP proxy and registrar, the SIP registration information is also available through the H.323 gatekeeper, any H.323 entity can resolve the address of SIP entities reachable via the SIP server, thus an active intervention is performed only in the case of calls originated from the SIP domain towards the H.323 one.

In such a case, if a SIP user agent wants to talk to another user located in the H.323 network, it sends a SIP INVITE message to the SIP server which, in turns, forwards an H.323 location request (LRQ) to the configured gatekeeper (if no gatekeeper was configured then a broadcast LRQ is sent). The gatekeeper responds with the IP address of the H.323 user making the SIP server able to route the call to the destination. Drawbacks of this approach are that the H.323 gatekeeper may be highly loaded because of all the registrations in the SIP network.

Of course when the SIP / H.323 gateway is independent of proxy or gatekeeper it must query the other network for user location acting an active intervention in the call.

5.1.4.3. Call from H.323 to SIP

Similarly to the previous case, calls from the H.323 domain to the SIP one requires an active intervention of the SIP / H.323 gateway only when it contains and H.323 gatekeeper and the call is originated from the H.323 domain since H.323 terminals appear as SIP URLs within the same domain to the SIP user agents (for further information on how H.323 addresses are translated to SIP URLs please refer to Interworking Between SIP/SDP and H.323[3]).

In this case, if an H.323 user wants to talk to a user located in the SIP network, it sends an admission request (ARQ) to its gatekeeper. The gatekeeper has to send the location request (LRQ) to all the gatekeepers it has configured as neighbours. If the SIP / H.323 built-in gatekeeper receives the LRQ, it tries to resolve the address of a SIP user. This address resolving is done by sending a SIP OPTIONS request. If this operation succeeds and the user is currently available to be called, the SIP / H.323 gateway replies to the H.323 network gatekeeper with the location confirmation (LCF), after the H.323 terminal knows that the destination is reachable. Drawbacks of this approach are, like in the previous case, that the SIP proxy has to store all H.323 registration information.

Of course when the SIP / H.323 gateway is independent of proxy or gatekeeper it must query the other network for user location acting an active intervention in the call.

5.1.4.4. Media Switching and Capability Negotiation

SIP / H.323 gateway should be configured in order to have media transport directly connected between the SIP and the H.323 entities; when, in some cases, this is not possible, the SIP / H.323 gateway should have a built-in media switching fabric activated to forward RTP and RTCP packets from one client to the other. A great interworking effort is carried out in capabilities negotiation: while SIP offers media with the INVITE message, the normal H.323 mode is to open a separate H.245 channel. In this case the SIP / H.323 gateway have to start a muted SIP call and re-negotiate the capabilities later unless the H.323 client support the use of FastStart procedure. If both these conditions can not be ensured, the gateway must use the default codecs for both sides and, eventually, perform media transcoding.

[3] http://www.cs.columbia.edu/~kns10/publication/iptel2000.pdf

5.1.4.5. Call termination

Since a call can be terminated from both H.323 and SIP clients, appropriate message translation / forwarding is required from the SIP / H.323 gateway: a SIP BYE message is mapped to a H.323 Release Complete message and vice-versa.

5.1.4.6. Configuration guidelines

In this section we are going to give the reader basic configuration principles abstracting them from a specific software in order to give general guidelines and not information becoming rapidly out-of-date. Apart from low impact configuration information (log files location, log level setting), in order to configure a SIP / H.323 gateway it is likely that there will be the need of configuring:

- enabling / not enabling of built-in SIP proxy / registrar and related configuration, please refer to section on setting up SIP services for information on how to configure a SIP proxy / registrar server;
- the remote address / port of the H.323 network gatekeeper (either if SIP built-in proxy / registrar is enabled and no broadcast requests has to be sent or if the SIP / H.323 gateway is independent of proxy or gatekeeper);
- enabling / not enabling of built-in H.323 gatekeeper and related configuration, please refer to section on setting up H.323 services for information on how to configure a H.323 gatekeeper;
- the remote address / port of the SIP network proxy / registrar (either if H.323 built-in gatekeeper is enabled or if the SIP / H.323 gateway is independent of proxy or gatekeeper);

5.1.5. Accounting Gateways

Accounting may be performed both on Gatekeepers and on Gateways. Even if, in principle, accounting done on Gateways is possible, the best solution is to centralize the accounting on Gatekeepers which are able to maintain all the call information (if configured in call signaling routed mode). Information on how to perform accounting may be found in Section 4.3.

5.2. Supplementary services

This section deals with supplementary services both using H.323 and using SIP. These supplementary services are intended to be used in addition to the basic one but still in a telephony-like environment. The supplementary services are protocol-specific and are intended to replicate all the wide range of services we are already used to in the PSTN networks. Section 5.2.1 will deal with the supplementary services using the H.323 protocol, while Section 5.2.2 will deal with supplementary services using the SIP protocol.

5.2.1. Supplementary Services using H.323

This section describe the supplementary services of the H.323 protocol as specified in the H.450.x supplementary services recommendations.

The supplementary services of H.323 are defined in H.450 series, which establish the signaling between endpoints necessary to implement the telephone-like services. Although some of these services have the same functionalities of the equivalent ones developed for the circuit-switched networks, it is relevant to note that the paradigm of the H.323 is completely different.

The peculiar characteristic of the supplementary services of H.323 is that the protocol actions needed for their control are performed using peer-to-peer signaling. In other words, the protocols are designed in such a manner that functional entities communicate with their peer entities (clients, gatekeepers, gateways etc.) directly without requiring network intervention.

The services are distributed in the endpoints, based on the suitability of the service at that endpoint. For example, an H.323 client maintaining the state of the calls is suitable for the implementation of service such as call transfer, call forwarding, call waiting and so on. Since the peer-to-peer model used by the supplementary services of H.323, the payload and the signaling of the services are transparently sent through the network without requiring the processing of any network element.

Considering also that the state of the calls is distributed to the endpoints involved in the calls, we can deduce that services for H.323 can be deployed by any manufacturer and sold directly to the end user for deployment. This feature of the service deployment leads to a low cost entry for the service provider and a service cost for customer characterized by an initial cost for the software implementing the service for unlimited use. This last aspect implies that new services can be distributed to the VoIP users in the same way as any other software is sold in the market today.

This scenario raises up problems related to the subscription control of these supplementary services. To this aim, the signaling necessary to these services should routed through the Gatekeeper (or other proxy elements) that, accessing to back-end services containing a service database description, permits, for example, the charging for the use of supplementary services or their blocking in the case there are not subscribed by the customers.

Another issue related to the peer-to-peer approach used in the H.323 supplementary services is the service incompatibility. In H.323, the clients exchange their capabilities and hence, they are able to only use those services that are common to both clients. Therefore, services that are present in one client are simply not used if they are not present in the other client involved in the call.

In the case of hybrid networks, e.g. PSTN and H.323, the supplementary service functionality and availability on the calls between legacy and H.323 networks depend on the capabilities of the gateway, which must perform the signaling translation necessary to guarantee the services. Moreover, the gateway can be used by the provider to charge the service.

Supplementary services in H.323 are specified in a multi-tier approach. Basic services consist of building blocks or primitives from which more complex services can be developed. Compound services are developed from two or more basic services. For example, in a consultation transfer service, the user needs to put a multimedia call on hold and retrieve it later, to call another person and possibly alternate between the two calls, and to transfer the two calls together. Hence, this service is simply obtained combining basic supplementary services. The basic supplementary services defined in the H.450 series are described in the following.

- *H.450.2 - Call Transfer.* It enables a user A to transform an existing call (user A - user B) into a new call between user B and a user C selected by user A.
- *H.450.3 - Call Diversion.* It is also known as call forwarding and comprises the services call forwarding unconditional, call forwarding busy, call forwarding no reply and call deflection. It applies during call establishment by providing a diversion of an incoming call to another destination alias address. Any of the above variants of call forwarding may operate on all calls, or selectively on calls fulfilling specific conditions programmed or manually selected by the user.
- *H.450.4 - Call Hold.* It allows the served user (holding user), which may be the originally calling or the called user, to interrupt communications on an existing call and then subsequently, if desired, re-establish (i.e. retrieve) communications with the held user. During the call interruption, the holding user provides some form of media for the held user, and may perform other actions while the held user is being held, e.g. consulting another user.
- *H.450.5 - Call Park and Pickup.* This is a service that enables the parking user A to place an existing call with user B to a parking position and to later pick up the call from the same or other terminal.
- *H.450.6 - Call Waiting.* It permits a served user while being busy with one or more calls to be informed of an incoming call from a new user by an indication. The served user then has the choice of accepting, rejecting or ignoring the waiting call. The user calling the busy party is informed that the call is a waiting call at the called destination.
- *H.450.7 - Message Waiting Indication.* It provides general-purpose notifications of waiting messages, including the number, type and subject of the messages.
- *H.450.8 - Name Indication.* This service permits to a user A to be informed of an incoming or existing call, or of the alerting state by a user B. The notification can be provided from a server or directly from user B.
- *H.450.9 - Call Completion.* It enables a calling user A to have the call toward a user B completed without having to make a new call attempt, when the first call procedure has been not completed since user B was busy or, though alerted, did not answer.
- *H.450.10 - Call Offer.* It permits to a calling user A, encountering a busy destination user B, to "camp-on" to the busy user. This means that the call is indicated to user B and kept in a waiting state until user B reacts to the indication, rather than being released due to the busy condition.
- *H.450.11 - Call Intrusion.* It enables a calling user A, encountering a busy destination user B, to establish communication with user B by breaking into an established call between user B and a third user C.
- *H.450.12 - Common Information Additional Network Feature (ANF-CMN).* This is an additional network feature that enables the exchange of Common Information between ANF-CMN endpoints. This Common In-

formation is a collection of miscellaneous information that relates to the endpoint or equipment at one end of a connection and includes one or more of the following: Feature Identifiers, Feature Values, or Feature Controls. This information, when received by an ANF-CMN endpoint, can be used for any purpose, e.g. as the basis for indications to the local user or to another network or in order to filter feature requests.

The following examples can be useful to understand how the supplementary services can be implemented. In the example, the attention will be focused mainly on the signaling procedure, considering that the peer-to-peer approach adopted by the H.323 supplementary services concentrates the problems related to the service implementation to the endpoint or gatekeeper used as service server.

5.2.1.1. Call Transfer Supplementary Service

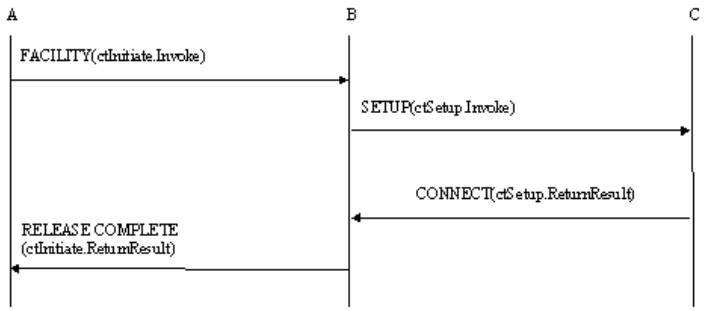
Supplementary Service Call Transfer (SS-CT) is a supplementary service which enables the served user (user A or transferring user) to transform an existing call with user B (primary call) into a new call between user B and a user C (transferred-to user) selected by user A.

It is relevant to note that the primary call between user A and user B must be answered before transfer can be initiated. On successful completion of SS-CT user B and user C can communicate with each other and user A will no longer be able to communicate with user B or user C.

The signaling necessary to use the service are implemented using a set of messages forming the Application Protocol Data Unit (APDU), which are transported within User to User information elements in call control and FACILITY messages as defined in the ITU-T Recommendation H.450.1.

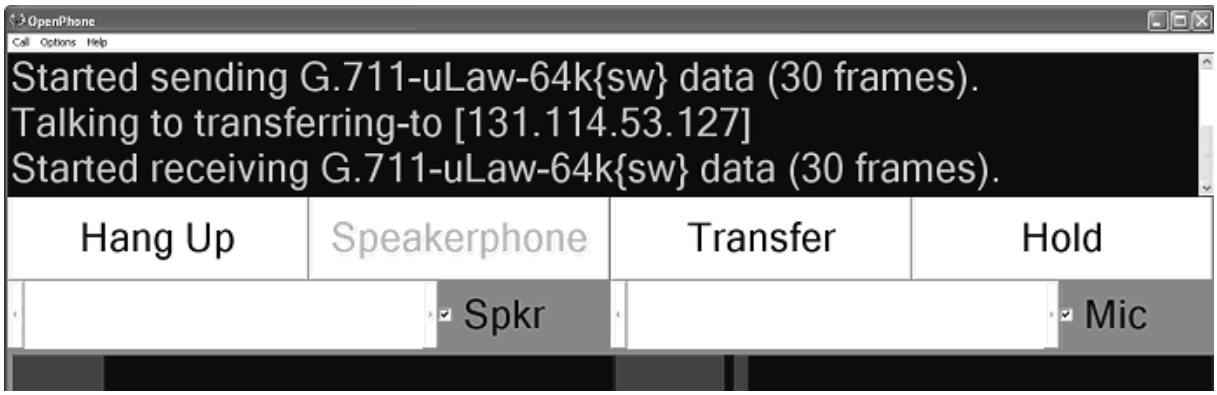
As an example, Figure 5.10 reports the signaling messages exchanged to perform the Call Transfer from the primary call between user A and user B to the new call involving user B and user C. In particular, when it decides to perform the Call Transfer, user A sends to user B the FACILITY message with the APDU denoted as CallTransferInitiate.Invoke containing the address of the user C. Using this information user B initiates the procedure to open the new call with user C transmitting the SETUP message with the CallTransferSetup.Invoke APDU. The user C accepts the call, sending the CONNECT message with the appropriate APDU. When user B receives the CONNECT message, it tears down the primary call with user A, transmitting the RELEASE COMPLETE message with the appropriate APDU.

Figure 5.10. Messages exchanged to implement the CT-SS without Gatekeeper



The example refers to the case where the Gatekeeper is absent or the direct signaling model is adopted. In these cases, only the software able to process and to generate the APDUs is necessary to implement the service. An example is given in Figure 5.11 where we report an Ohphone (an openh323 application) interface modified with addition of Call Transfer Supplementary services button. In this case simply pressing the “Transfer” button makes the application opening a dialog screen, letting us inserting the H.323 ID of the callee and generating the necessary APDUs.

Figure 5.11. Example of Call Transfer Supplementary Service without Gatekeeper - Ohphone modified interface



5.2.1.1.1. Gatekeeper role in the Call Transfer

In the case of the gatekeeper routed model, the gatekeeper shall either transparently transport or perform the operations necessary to offer the service. In particular, the Gatekeeper can provides CT-SS, when one or both endpoints are unable to support the service.

Hence, in order to provide the service the Gatekeeper can decide to perform the actions applicable to the transferred endpoint, to the transferred-to endpoint or both. When the Gatekeeper handles the Call Transfer signaling on behalf of an endpoint, it should perform further procedures as defined for the transferred and the transferred-to endpoints.

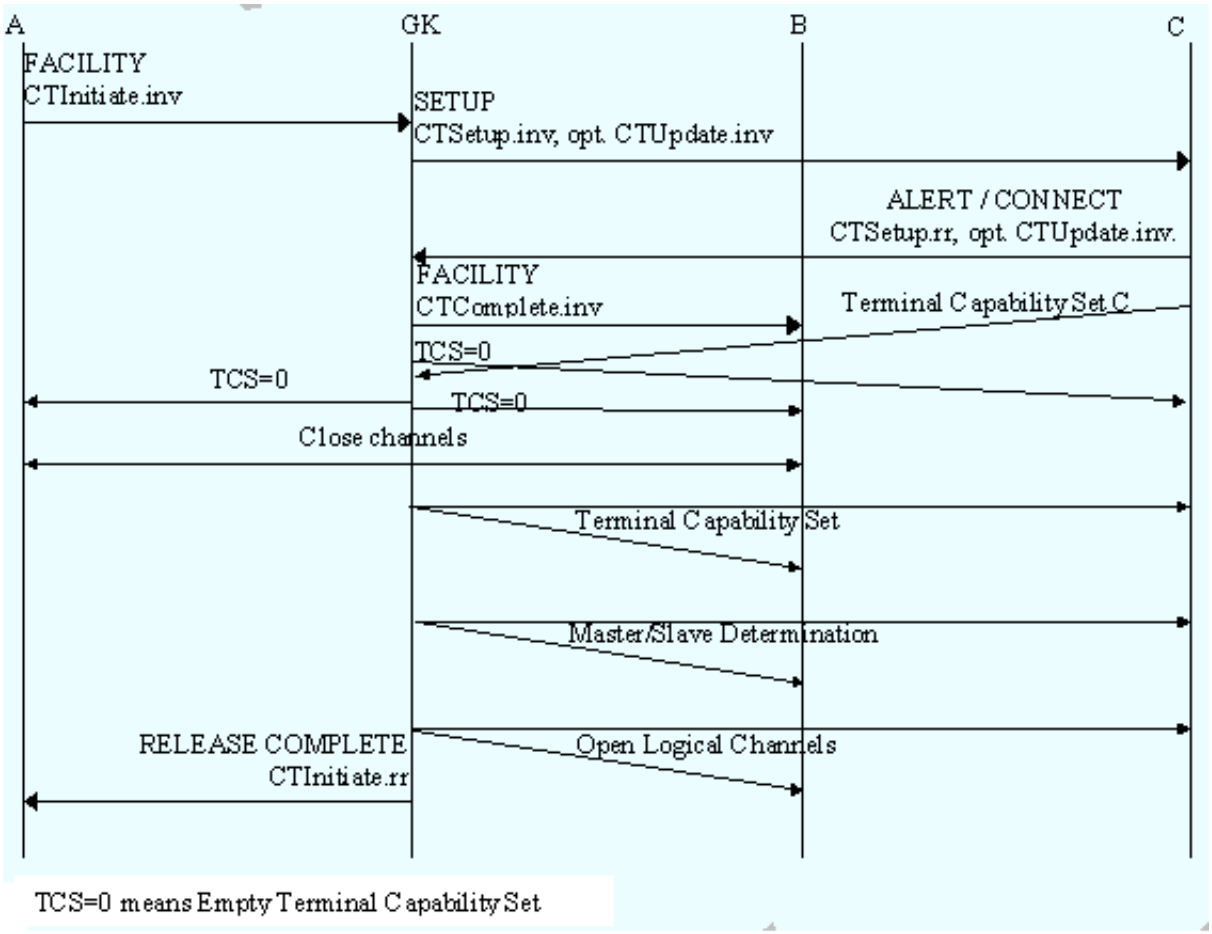
These further procedures are the sending of a FACILITY message with an appropriate APDU used to inform the transferred user (or the transferred-to user when it manages the signaling on behalf of transferred-to user) that it has been transferred. Furthermore, the Gatekeeper should instruct the endpoint (or endpoints), that it manages the signaling on behalf, of the new set of media channels to connect.

To accomplish this, the Gatekeeper uses the H.323 procedures for third party re-routing. These procedures require the gatekeeper to send an empty terminal capability set (one which indicates that the remote entity has no receive capabilities) to endpoints A and B, causing A and B to close their logical channels. Then, the Gatekeeper exchanges the H.245 command "end session" with endpoint A and sends a RELEASE COMPLETE message containing the result APDU to release the call signaling channel.

When it receives a non-empty terminal capability set from endpoint C, the Gatekeeper forwards the capability set to endpoint B to cause it to reset its H.245 associated state to that which it is in when H.245 has just completed (the first) terminal capability set exchange in the initial call establishment sequence. Then the Gatekeeper takes part in master/slave determination; and opens appropriate logical channels with endpoint C.

To better understand the signaling procedure used, Figure 5.12 illustrates the messages exchanged for the CT-SS when the Gatekeeper manages the signaling on behalf of the transferred user, i.e. B user.

Figure 5.12. Messages exchange for Gatekeeper managed CT-SS



After the reception and the processing of the FACILITY message indicating that user A wants transfer the current call, the Gatekeeper sends a SETUP message with a callTransferSetup.invoke APDU to the transferred-to endpoint C. Then the reception of the ALERTING or CONNECT message with the appropriate APDU transmitted by the user C, enables the Gatekeeper to send a FACILITY message with the APDU informing the endpoint B that it has been transferred ("joining").

To instruct the endpoint for the new set of media channels, the Gatekeeper sends an empty terminal capability set (it is defined as a terminalCapabilitySet message that contains only a sequence number and a protocol identifier) to endpoints A and B, causing A and B to close their logical channels, and to endpoint C, causing this endpoint to enter in the "transmitter side paused" state.

While in this state, an endpoint does not initiate the opening of any logical channels, but accepts the opening and closing of logical channels from the remote end based on the usual rules and continues to receive media on open logical channels opened by the remote endpoint. This procedure allows Gatekeepers to re-route connections from endpoints that do not support supplementary services and endpoints to receive announcements (e.g. pre-connect call progress) where the announcing entity does not wish to receive media from the endpoint.

The "transmitter side paused" state is left by the endpoint on reception of any terminalCapabilitySet message, other than an empty capability set. On leaving this state, an endpoint resets its H.245 state to that which it was in just after the H.245 transport connection was made at call establishment time (i.e. the beginning of phase B), but it preserves state information relating to any logical channels that are open.

This puts the endpoint in a known H.245 state after the pause, allowing an endpoint to be connected to a different endpoint when it is released from the paused state. After leaving the "transmitter side paused" state, an endpoint proceeds with normal H.245 procedures: it takes part in master/slave determination signaling and proceeds with normal open logical channel signaling procedures.

After these procedures necessary to set the new call between B and C, the Gatekeeper sends a RELEASE COMPLETE message containing callTransferInitiate return result APDU to release the call signaling channel of the primary calls, i.e. between A and B.

5.2.1.2. Call Diversion Supplementary Service

The signaling procedures and the protocols needed to implement the Call Diversion Supplementary Service are defined in the Recommendation H.450.3. The service permits to a user, denoted as Served user, receiving a call from an Originating user to redirect it to another user, denoted as Diverted-to user. This operation leads to connect the Originating user with the Diverted-to user, and it is possible only when Served and Originating user are not in a call. There are four kinds of Call Diversion service.

Supplementary Service Call Forwarding Unconditional (SS-CFU) permits a Served user, independently from its status, to forward incoming calls addressed to the Served user's number to another number. CFU is provided on a per number basis. On the contrary, in the case of the Supplementary Service Call Forwarding Busy (SS-CFB) the call forwarding is made by the Served User only when it is busy. A bit different is the Supplementary Service Call Forwarding No Reply (SS-CFNR), where the call forwarding begins when the Served user does not establish the connection within a defined period of time.

All these kinds of services may be either permanently activated or activated/deactivated under user control. The user control can be provided locally, at the Served endpoint, remotely by another endpoint or both. Other than the activation/deactivation procedures, the H.450.3 defines the interrogation and the registration procedures.

The interrogation procedure provides information to the interrogating endpoints such as the activated or deactivated state of the supplementary service, and, if the service is activated, the diverted-to number, whether activated for all basic services or an individual basic service and the identity of the individual basic service. The registration permits to provide the information related to the activated service, and it is performed on the activation procedure.

Indeed, to activate these services, the Served user shall supply the diverted-to number and optionally further parameters, depending on the capabilities of the specific implementation. Verification that the diverted-to number exists may be carried out before accepting the activation request.

In the same recommendation is defined also the Call Deflection (SS-CD), which permits a Served user to respond to an incoming call offered by the Served endpoint by requesting diversion of that call to another number specified in the response. Hence, the service does not require activation/deactivation/information/registration procedures. The diversion request is only allowed before the called user has answered the call and, in particular, when the Served user is in the alerting state.

On acceptance of the CD request, the served endpoint performs the diversion towards the indicated diverted-to number. The original call at the served user remains in the alerting state and the served user is still able to accept the call until the diverted-to endpoint enters an alerting state. When the diverted-to endpoint enters the alerting state the call to the served user is cleared.

5.2.1.2.1. Gatekeeper role in the Call Diversion

The Gatekeeper can be either transparent to the Call Diversion Service or directly manage it. In the first case, the implementation of the service is directly on the user endpoint or in proxy server.

When the service is managed by the Gatekeeper, the messages related to the Activation/Deactivation/Interrogation/Verification of the Diverted-to number are received and processed by it, which acts as Served endpoint. Thus, when the Originating user sends the ARQ message to contact the forwarding terminal for sending the Activation/Deactivation/Interrogation/Verification messages, the Gatekeeper responds returning the ACF message containing its own call signaling address, rather than the call signaling address of the forwarding terminal.

The invocation of the service depends on the kind of Call Diversion invoked. In particular, when the SS-CFU is activated in the Gatekeeper for an incoming call destined for user B (being the Served user), the Gatekeeper acts as Served endpoint and invokes the call forwarding unconditional for this call. Furthermore, if the option "served user notification" applies the Gatekeeper sends the notification to the called terminal.

In the case of the SS-CFB, the Gatekeeper continues the H.225 call establishment procedure to endpoint B, where the user B (being the served user) is. If the endpoint B results busy, i.e. if a RELEASE COMPLETE message is received from endpoint B containing Cause value "user busy", the Gatekeeper acts as served endpoint and invokes the call forwarding on busy for this call. Also in this case, the Gatekeeper sends the notification to the called terminal, if the option "served user notification" applies.

For the provision of the SS-CFNR service by the Gatekeeper, when arrives a call destined for user B (being the served user), the Gatekeeper starts a local "no-response" timer and continues H.225 call establishment procedure to endpoint B. If the local "no-response" timer expires during the alerting phase of the call, the Gatekeeper acts as served endpoint, invokes the call forwarding on no reply for this call and, if the option "served user notification" applies, sends the notification to the called terminal.

5.2.1.3. Call Waiting Supplementary Service

The Supplementary Service Call Waiting (SS-CW), defined in the H.450.6 Recommendation, permits a busy user B to be informed of an incoming call while being engaged with one or more other calls.

In other words, SS-CW operates in case of an incoming call (from user C, Calling user) when a busy condition within the endpoint is encountered. To note that in general, a busy condition may also be encountered if the user is busy with workflow applications (e.g. writing e-mails).

When the SS-CW is invoked, user B is given an appropriate indication of the waiting call and the calling user C may be informed about SS-CW being invoked at the destination by being provided with an appropriate indication. After receiving the call waiting indication, user B has the choice of accepting, rejecting or ignoring the waiting call. During the call waiting condition, the calling user C has the option to release the call or to invoke other supplementary services, e.g. message waiting callback.

In the case of endpoints able to simultaneously manage different calls, the SS-CW is invoked only when an attempt for a new call (i.e. the H.225.0 SETUP message from user C arrives to endpoint B) is made to exceed the maximum number of calls the endpoint B can simultaneously hold. This condition leads the served endpoint B to return an ALERTING message towards the calling user C containing the appropriate APDU.

Meanwhile, the endpoint B locally provides a call waiting indication to the user B. The busy User B can free resources to accept a waiting call by:

- releasing an existing call according to the procedures of recommendation H.225.0;
- using the Call Hold Supplementary Service on an existing call according to the procedures of Recommendation H.450.4;
- using the Call Park Supplementary Service on the existing call according to the procedures of Recommendation H.450.5.

If the served user B accepts the waiting call, the served endpoint sends the CONNECT message to the calling user and proceeds with normal call establishment procedures.

After the reception of the ALERTING message containing the APDU indicating the invocation of the SS-CW by the endpoint B, the calling endpoint provides a call waiting indication to the calling user, which has the following options:

- wait until the waiting call gets accepted (connected) by the served user B;
- release the call;
- invoke other supplementary services, e.g. message waiting callback;
- perform other actions (e.g. sending e-mail).

5.2.1.3.1. Gatekeeper role in the Call Waiting

In the case of the Gatekeeper routed model, the Gatekeeper passes on SS-CW operations transparently. When a Gatekeeper has appropriate knowledge about the served endpoint B status, it may act on behalf of the served endpoint by means of inserting the APDU indicating the invocation of the CW-SS into an ALERTING message received from endpoint B before sending on the ALERTING message towards the calling endpoint.

5.2.1.4. Supplementary services (H.450) support in popular gatekeepers

The Cisco MCM gatekeeper does not support any H.450 features, even though it is itself capable of H.323 proxying. However, Cisco gateways do support an extensive range of H.450 features and can play a significant role in supporting supplementary services for H.450 capable endpoints that would like to reach the PSTN.

The Radvision ECS gatekeeper, being based on a H.323 v.4 protocol stack, implements a number of the H.450 features. On the ECS configuration interface, check the Settings Tab, under the category "Supplementary Services", where you can enable the following:

- Unconditional forward (H.450.3): endpoints may select to redirect calls to other endpoints in all cases.
- Forward on busy (H.450.3): endpoints may select to redirect calls to other endpoints when they are themselves busy.
- Forward on no response (H.450.3): endpoints may select to redirect calls to other endpoints when they could not respond to a call after x minutes.

After enabling the above services, the gatekeeper administrator can define forwarding rules by utilizing the Forwarding Tab. Rules for specific endpoints (Standard), as well as mass-application (Wildcard) rules for whole prefixes can be applied for all three forwarding conditions: unconditional, on busy, on no answer.

For all supplementary services to function, the administrator must enable full routing mode (Settings Tab, category Calls). Make note that the ECS administrator can also initiate calls between endpoints, by using the Call Control Tab, and the Make Call button.

The GNU gatekeeper does not support any H.450 features either. However, some dynamic routing functionality can be implemented by utilizing the "virtual queues" or "CTI agents" feature of the GNU gatekeeper, which allows the operation of a simple model of aliasing a name and the forwarding of calls to a dynamic queue of "agent" endpoints.

5.2.2. Supplementary Services using SIP

This section demonstrates the use of legacy telephony features in SIP. Note that the general SIP design concept has been to leave the description of such features out of protocol specifications. It is responsibility of implementations to introduce new services on top of well defined protocol specification. There are only few well defined protocol elements such as REFER method[4] (RFC 3515). Such elements can be used for a variety of services without having to undergo the burdensome standardization process again and again. With the particular REFER example, the REFER method may be used for several variations of call transfer, click-to-dial application as described in the next section, interaction with conference bridges, etc. The application logic may be completely hidden in a SIP element, or it can use some established service building mechanisms such as Call Processing Language [5] or SIP-CGI (RFC 3050)[6].

To show the proper use of protocol elements for building frequently used services, the IETF issued an informational document Session Initiation Protocol Service Examples [7]. The document presents call flows for many services, including on-hold, transfers, conferencing. Another essential document is A Call Control and Multiparty usage framework for the Session Initiation Protocol (SIP)[8] by Rohan Mahy et al. The document shows the concepts of building more complex services based on SIP and also presents multiple design choices for many of them. Yet another usage document related to implementation of telephony services is Third-party call control[9], which shows how to implement complex multi-stage multi-party telephony conversations using external SIP-based call control.

The rest of this section demonstrates the most frequently used SIP services, gives a brief overview on provisioning of such services, and also describes how the signalling looks like.

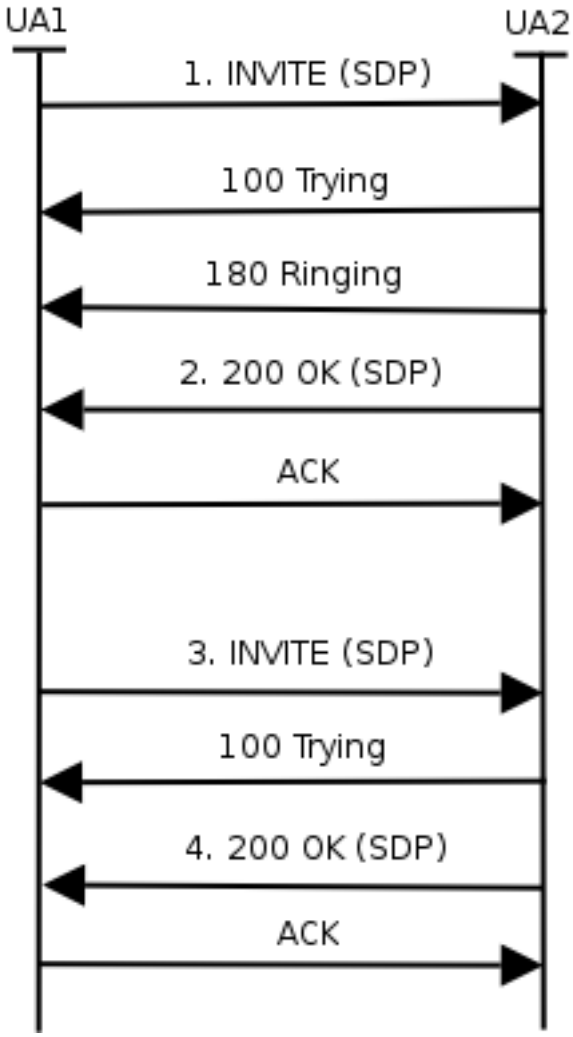
5.2.2.1. On Hold

With SIP, on-hold is implemented as a peer-to-peer feature. A phone putting the other telephone on hold does so by sending a specially-formed re-INVITE request. In response to such a request, the other phone stops sending media to save bandwidth and it indicates to user that he is on hold. (It may be silent, display status using user interface, play local music, etc.) There are subtle differences in how "on-hold" is signaled in earlier and current SIP version. RFC 2543 used dummy IP address 0.0.0.0 to indicate on-hold condition whereas RFC 3261 uses SDP sendonly attribute. Call-flow on Figure 5.13 presents on-hold signalling implemented according to RFC 2543. The key message is re-INVITE, number 3, which puts other party on hold. The SDP payload of numbered

[4] http://www.ietf.org/rfc/rfc3515.txt
[5] http://www.iptel.org/ietf/callprocessing/interfaces/#draft-ietf-iptel-cpl
[6] http://www.ietf.org/rfc/rfc3050.txt
[7] http://www.iptel.org/ietf/callflows/#draft-ietf-sipping-service-examples
[8] http://www.iptel.org/ietf/callprocessing/#draft-ietf-sipping-cc-framework
[9] http://www.iptel.org/ietf/callprocessing/3pcc/#draft-ietf-sipping-3pcc

SIP messages is shown in detail (SIP headers have been removed because they are not important in this case).

Figure 5.13. On-hold Call Flow



The following message is regular INVITE establishing a call

```
1. INVITE

[SIP headers not shown]

v=0
o=Cisco-SIPUA 997 27044 IN IP4 192.168.2.32
s=SIP Call
c=IN IP4 192.168.2.32
t=0 0
m=audio 21112 RTP/AVP 0 8 18 101
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
```

The previous INVITE is replied using the following 200 OK

```
2. 200 OK
```

```
[SIP header not shown]

v=0
o=CiscoSystemsSIP-GW-UserAgent 2451 1894 IN IP4 195.37.77.110
s=SIP Call
c=IN IP4 195.37.77.110
t=0 0
m=audio 18202 RTP/AVP 0
c=IN IP4 195.37.77.110
a=rtpmap:0 PCMU/8000
a=direction:passive
```

The following message puts the remote party on hold. Note the 0.0.0.0 on the line beginning with “c=”.

```
3. re-INVITE

[SIP headers not shown]

v=0
o=Cisco-SIPUA 4919 16082 IN IP4 192.168.2.32
s=SIP Call
c=IN IP4 0.0.0.0
t=0 0
m=audio 21112 RTP/AVP 0 8 18 101
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
```

The following message confirms the on hold state. There is nothing special in the SDP

```
4. 200 OK

[SIP headers not shown]

v=0
o=CiscoSystemsSIP-GW-UserAgent 2451 1895 IN IP4 195.37.77.110
s=SIP Call
c=IN IP4 195.37.77.110
t=0 0
m=audio 18202 RTP/AVP 0
c=IN IP4 195.37.77.110
a=rtpmap:0 PCMU/8000
a=direction:passive
```

5.2.2.2. Call Transfer

The call transfer is one of the most frequently used telephony services. The protocol element used in SIP to implement call transfer is the REFER method. The REFER method is a very powerful mechanism, which allows anybody to ask a SIP device to initiate a call to a specific destination. Call transfer is only one application which relies on REFER -- “click to dial” and conference management can utilize REFER as well. Even call transfer may be implemented in a variety of ways in SIP telephones.

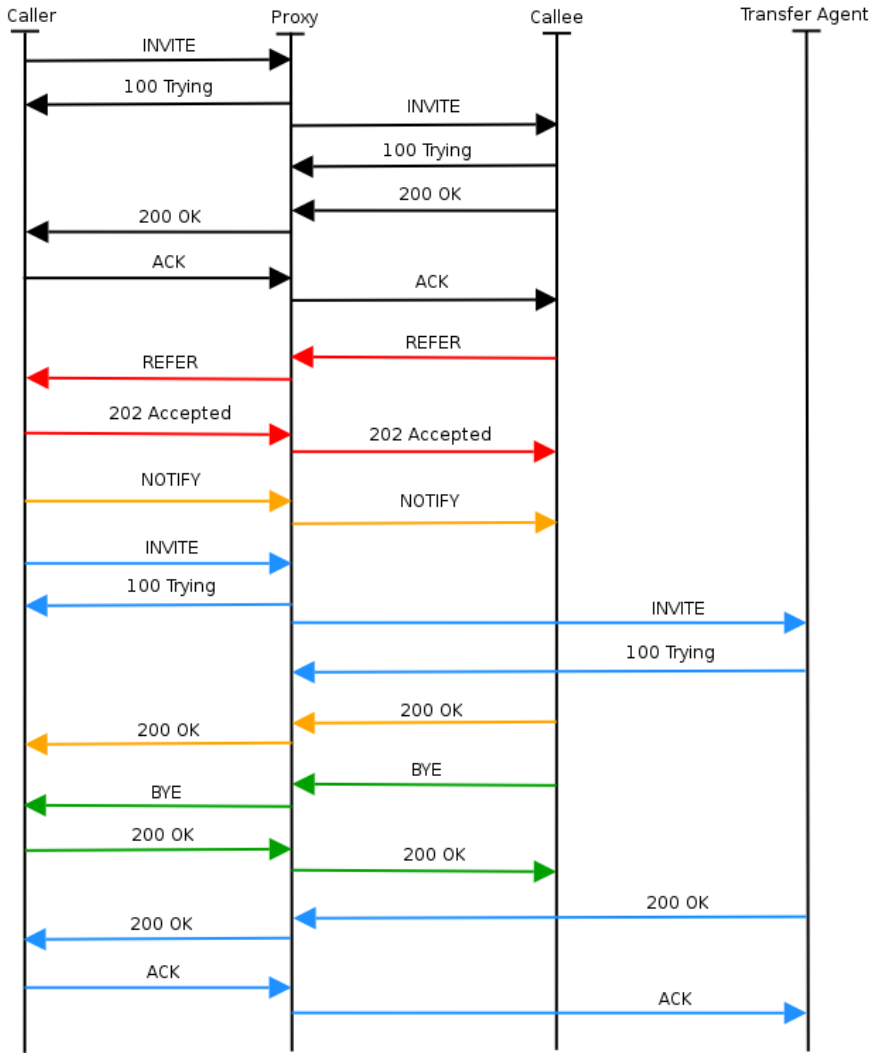
“Unattended transfer” (also called “Blind Transfer”) transfers a call participant to another party, whereas the transfer originator drops the initial conversation.

In “Attended transfer” (also called “Transfer with consultation”), the transferor first initiates a short conversation with the transfer target before connecting it to the transfered party.

Other variations may include a short introductory conference during the call transfer or keeping the original conversation active until the transfer succeeds. (the NOTIFY request is used to report on success of call transfer in SIP.)

The following call flow demonstrates the use of REFER for unattended call transfer. The key requests are REFER (red) by which the callee suggests the caller to establish a conversation with the transfer target. The caller does so by sending the INVITE (blue). The callee exits the original conversation by sending BYE (green) without awaiting the completion of the call transfer.

Figure 5.14. Call Transfer Call Flow



The following REFER message (red in the call flow) asks the caller to establish a call to the transfer agent. Note that the message contains the “Refer-To” header field containing SIP URI of the transfer agent and “Referred-By” header field which contains the SIP URI of the initiator (the callee).

```
REFER sip:195.37.77.101:5060;lr SIP/2.0
Via: SIP/2.0/UDP 192.168.2.32:5060
From: <sip:callee@iptel.org>;tag=00036bb90fd300
To: <sip:caller@iptel.org>;tag=992d8f53-ca7e
Call-ID: 90bdee69-eb4f@192.168.0.130
CSeq: 103 REFER
Contact: sip:callee@192.168.2.32:5060
Route: <sip:caller@193.175.135.38:40012>
Content-Length: 0
Refer-To: sip:transfer_agent@195.37.77.101
Referred-By: <sip:callee@iptel.org<
```

The following INVITE message (blue in the call flow) establishes a call from the caller to the transfer agent. Note that the value from the “Refer-To” header field of the REFER message has been put into the Request-URI and “To” header field.

```
INVITE sip:transfer_agent@195.37.77.101 SIP/2.0
Via: SIP/2.0/UDP 192.168.0.130
From: <sip:caller@iptel.org>;tag=32e189db-ec7e
To: <sip:transfer_agent@195.37.77.101>
Contact: <sip:caller@192.168.0.130>
Call-ID: 205e377f-7042-b00c-0@192.168.0.130
CSeq: 20142 INVITE
Max-Forwards: 70
Content-Type: application/sdp
Content-Length: 258

v=0
o=caller 0 0 IN IP4 192.168.0.130
s=-
c=IN IP4 192.168.0.130
t=0 0
m=audio 5004 RTP/AVP 0 8 4 18 2 15
a=ptime:20
a=rtpmap:0 PCMU/8000
```

The following BYE message (green in the call flow) terminates the call between caller and callee.

```
BYE sip:195.37.77.101:5060;lr SIP/2.0
Via: SIP/2.0/UDP 192.168.2.32:5060
From: <sip:callee@iptel.org>;tag=00036bb90fd3000
To: <sip:caller@iptel.org>;tag=992d8f53-ca7e-c73f
Call-ID: 90bdee69-eb4f-3e1c-9833-499857a5b2b2@192.168.0.130
CSeq: 104 BYE
Content-Length: 0
Route: <sip:caller@193.175.135.38:40012>
```

5.2.2.3. Unconditional Call Forwarding

User's ability to determine call processing is a great strength of IP telephony. With SIP in particular, users are able to associate any number of devices with their address of record. SIP telephones register their whereabouts automatically using the SIP REGISTER request. Users may also introduce additional contacts by the means of provisioning. Upon receipt of an incoming call request (INVITE method), a proxy server forwards the request to all registered contacts. Eventually, all registered phones start ringing in parallel and the conversation begins with the device on which the user picks up.

The ability to manipulate contacts allows users to determine handling of incoming calls. For example, a user may wish to decide that all incoming calls will be ringing his cell phone as well. The following set of examples shows how the manipulation of registered user contacts can be used for forwarding purposes. The examples use SER and its command line control tool, serctl, to manipulate contacts of a user. Typically, this job is done through web interface such as Serweb because it is more convenient. The commands used are “ul add USER SIP_CONTACT” for introducing a new forwarding address and “ul show USER” for displaying currently registered contacts.

Let us begin with inspecting the current status of registered contacts. The command reveals that a SIP phone registered a contact from IP address 212.202.42.68.

```
jiri@fox:~$ serctl ul show jiri
<sip:212.202.42.68:55723<;q=0.00;expires=272
```

To enable forwarding of incoming calls also to cell phone, introduce contact to the PSTN gateway. If a call arrives, the proxy server will ring both -- previously registered SIP phone and manually provisioned cell phone

contact.

```
jiri@fox:~$ serctl ul add jiri sip:123456@iptel.org
sip:123456@iptel.org
200 Added to table
('jiri','sip:123456@iptel.org') to 'location'
jiri@fox:~$ serctl ul show jiri
<sip:123456@iptel.org>;q=1.00;expires=1073741820
<sip:212.202.42.68:55723>;q=0.00;expires=21
```

5.2.2.4. Conditional Forwarding

Callees frequently wish to redirect incoming calls to an alternative destination if primary destination fails to answer. The reasons for failure are multi fold. They may include busy callee, disconnected callee's phone, user who currently does not answer, or user denying the incoming call. The alternative destination is typically a voicemail system but it may be also another human or some other SIP device.

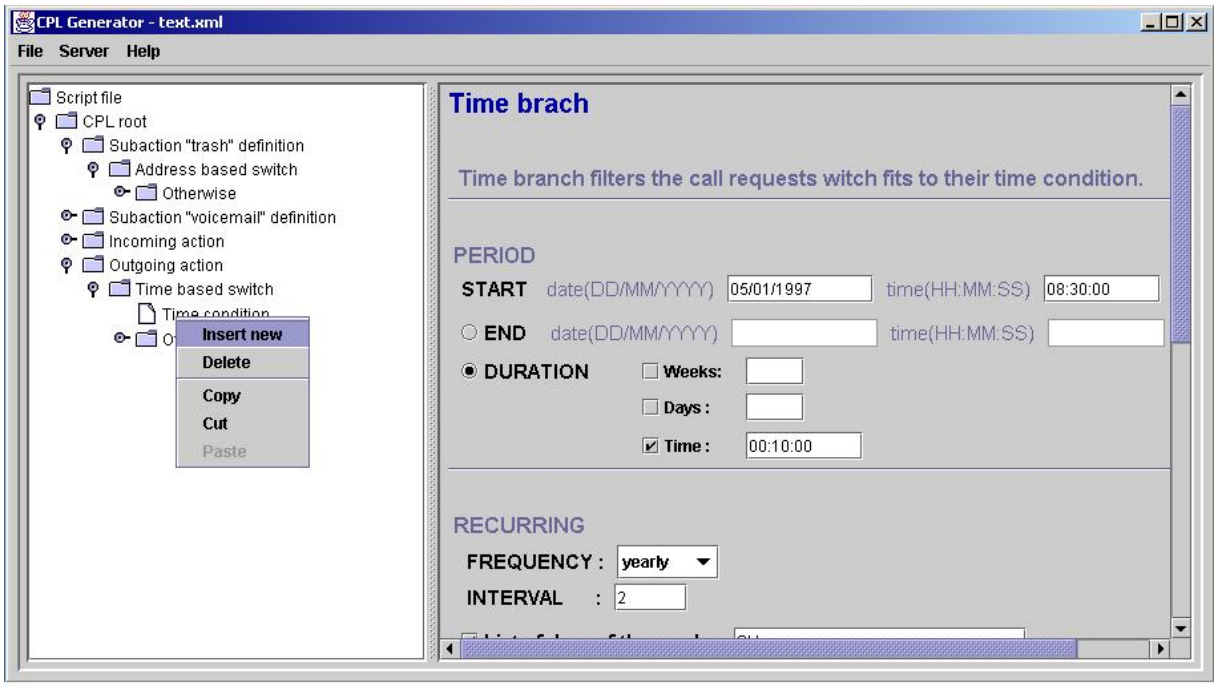
The following configuration fragment demonstrates how set up such a feature using SER:

```
# if invitation recipient off-line, forward to voicemail
if (!lookup("location")) {
    rewritehostport("voicemail.iptel.org");
    t_relay_to_udp("voicemail.iptel.org", "5060");
    break;
};
# user on-line, forward INVITE to him; also set up a failure
# handler so that we can redirect to voicemail if the
# call is not established successfully
if (method=="INVITE") {
    t_on_failure("1");
};
t_relay();

....

# alas, the call was not established successfully; well,
# let's retry with voicemail then
failure_route[1] {
    revert_uri(); # resend to voicemail with original request URI
    rewritehostport("voicemail.iptel.org");
    append_branch();
    t_relay_to_udp("voicemail.iptel.org", "5060");
}
```

Whereas this example demonstrates a global site policy which is applied to each user, some subscribers may wish to formulate their specific call processing preferences. How the preferences are formulated and provisioned in the SIP server is not necessarily governed by standards. SIP servers may have their own proprietary user provisioning interfaces, typically with a web front-end. On the other hand, a standardized way of call handling allows easier service portability. The IETF standard for describing callee's preferences is called Call Processing Language, CPL shortly. It is a simple XML-based language, that allows subscribers to determine how the server handles calls for them. It is a special-purpose language that support specification of most common types of call processing. CPL does not allow script writers to affect the behaviour of the server in a way which could compromise the security of the server.

Whereas it is simple enough, most users will not be willing to write SER scripts. It is envisioned that CPL scripts will be generated and edited by applications with a convenient user interface. The following picture shows a screen snapshot of iptel.org's CPL editor.

Figure 5.15. CPL Editor



5.3. Multipoint Conferencing

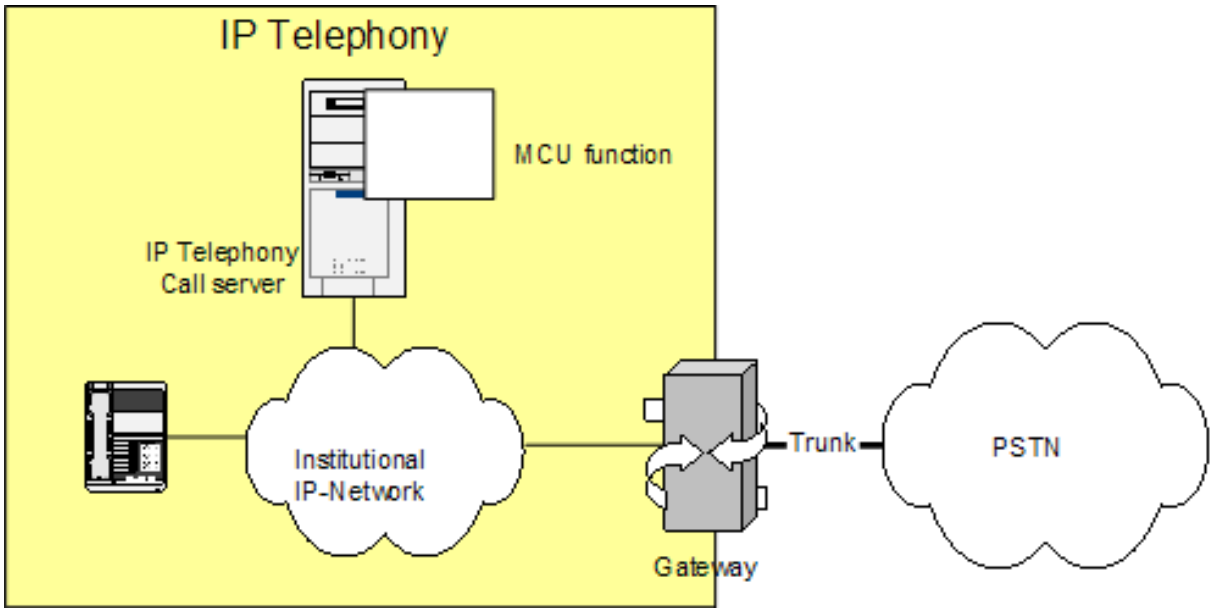
Multi-party conversations are known from the PSTN world. The function is often provided by a company's PBX. It is also possible to use a commercial service. A central part of the service is the Multipoint Conference Unit (MCU). In order to use the service, a session leader must make a reservation for the session at the MCU. Every MCU has a different interface to do so. MCUs in the IP-Telephony world usually offer a web based form for this.

At the time the conference is planned, each user calls the phone number of the MCU. After that, a number must be dialed to denote the session that the user wants to join, because an MCU can support multiple sessions at the same time. A password is required to prevent uninvited parties to join the conference. Some MCUs can initiate the set up of the conference itself by dialing all the parties. It needs to know everyones phone number in advance, of course.

The main function of the MCU starts at this point in the conference: it receives the audio signals of every party in the session, mixes the sources and copies the result to everyone except for the source party. This happens in real time, so everyone will hear everyone. This way of conversating has its specific ways of interaction between the parties. If a party wants to speak, it should be clear that the previous party has ended its part of the conversation. When collisions occur, it is useful if the session leader gives the word to one of those who wish to speak. These aspects have been investigated in the social-cultural area, but are not part of this cookbook.

Now that the functionality is known in general, more details on the case of IP Telephony MCUs can be given. An MCU can be obtained either in hardware or in software. Many Gatekeepers are equipped with built-in MCU software functionality. In case of a hardware MCU, the main interface is, of course, an Ethernet connection. From a functional point of view, users can not approach the MCU directly over IP. No matter whether H.323 or SIP are used setting up regular two-party calls, a user can only dial in to the MCU through a Gatekeeper or SIP proxy. Parties that use a PSTN phone can also join a conference by means of the IP-to-PSTN gateway.

Figure 5.16. MCU function in Gatekeeper



Modern MCUs can support both audio and video. The calling parties must support the audio and video codec that the MCU has on board. Some MCUs have the possibility to transcode between codecs, so enabling users with different codecs to join the same conference. If video is also distributed by an MCU, the video streams can not be mixed of course. One way this distribution mechanism is implemented is that only the video signal of the source with the loudest audio signal is transmitted to all users at that point of time. This in audio switching mode. There are other options, such as chair-controlled, in which the chair can lock the video (and possibly audio too) on one participant, or presentation mode, where one participant is chosen and both audio and video locked on the presenter and the rest of the audience can only listen. Some MCUs offer a Continuous Presence mode, in which video signals is displayed in a matrix that shows all users to every user. Modern MCUs support other layouts as well.

An alternative to using an MCU in the IP world is to use IP Multicast. In this case, all parties transmit their audio (and video) over a Multicast channel. All users must tune in to the channels of everyone else. This means that the total amount data traffic increases with every user that joins the conference. Unfortunately, few networks support Multicast.

Chapter 6. Setting up Value-Added Services

This chapter introduces reader to the concept of added value services and their implementation in IP telephony products. Value added services are services that integrate VoIP with other protocols and services. Seamless integration with Web, Email and potential other Internet services provides convenience which is a key feature of Internet telephony.

6.1. Web Integration of H.323 services

A natural extension to any internet service nowadays is its integration with the web. Unfortunately, H.323 is more difficult to integrate with web interfaces than SIP, due to the complexity of the protocol and its roots in the telecom world. Web applications that would be much needed in the area are the following:

- Presence: web-based applications that allow a group of users to indicate their availability for accepting calls and check availability of others. Analogous to the classic ICQ-type application, H.323 presence applications could stir-up significantly more users if they were widely available.
- Click-to-dial: web-based applications that allow users the ease to click in order to make a call to a target endpoint, service, or user. Obviously, H.323 endpoint applications are harder to set-up than e.g. MP3 client applications that download media streams, and this has a limiting effect on their use. Achieving the ease of use of a click-to-dial interface would have a dramatic effect on the deployment of H.323.
- IPPBX management: web-based applications that allow administrators of an IP-based PBX system to monitor and control calls being made. The simplicity of use of such web interfaces would allow the equivalent of a switchboard operator to: transfer calls, initiate multi-party calls, answer calls on busy, and terminate calls.

The applications above are mostly found in commercially available integrated solutions and they employ proprietary methods that are extremely difficult to integrate with custom made web interfaces. If you are interested in setting up your own web-based application and integrate it with your H.323 services, you must find the means of connecting it to the gatekeeper you have deployed. The methods are many and vary from gatekeeper to gatekeeper, depending on the available APIs and supported interfaces.

In this section, we will simply list some the options available for interfacing with common gatekeepers. For more information please refer to Appendix B.

6.1.1. RADIUS-based methods

If you are simply interested in making a presence application, then any gatekeeper with RADIUS support can be interfaced with, through proper RADIUS server set-up. For example, the FreeRadius server can be configured to execute an external script each time an authentication is made. This method can be used to keep a list of currently registered endpoints on the gatekeeper and make it available over the web as a primitive means of advertising availability. A slightly better method, would be to configure FreeRadius with MySQL back-end and maintain a database table with currently registered endpoints, by updating the table for each RADIUS event (authentication and accounting).

6.1.2. SNMP-based methods

An alternative to the above method, is to use SNMP to interface with your gatekeeper, assuming it has SNMP support. The Cisco MCM has SNMP support but with limited functionality, while the Radvision ECS supports the H.341 standard for SNMP access. You can explore the options available to an external application interfacing with SNMP by carefully studying the supported MIBs for your gatekeeper.

6.1.3. Cisco MCM GK API

The Cisco MCM gatekeeper actually implements a very flexible API for receiving events for triggers that the administrator is interested in processing. The external application can choose to further process the event, or just log it for informational purposes. The API interface to the MCM is proprietary and based on the GKTMP ad-hoc protocol for informing the external application of RRQ,URQ,ARQ,LRQ,DRQ,BRQ and related (confirm/reject) events. The API allows the external application to issue respective confirm and reject (xCF,xRJ) commands for specific events, based on external application logic. Cisco does provide a library in C and template code for an external application, but it not fully working code and requires significant resources to be applied before a demo application can be built. Also, there exist commercial solutions from Cisco partners which are based on this same API.

- GK API Guide Version 4.2[1]

6.1.4. GNU GK Status Interface

The GNU gatekeeper provides a very useful command line interface for monitoring and control of gatekeeper operations. It is called the "Status Interface" and allows telnet connections from remote administrative nodes to connect and monitor RCF/RJ,UCF/RJ,ACF/RJ,LCF/RJ,DCF/RJ,BCF/RJ events with detailed info. Additionally, it allows monitoring of call detail records (CDR) for accounting applications and "RouteRequest" messages for interfacing with the "Virtual Queues" feature, proprietary to the GNU gatekeeper. The fact that many different nodes can connect at the same time over this administrative interface and process different events of the gatekeeper, allows for a distributed and flexible implementation of monitoring services. Indeed, a number of tools have been developed that build on this interface and provide interesting functionality:

- OpenH323 Gatekeeper Java GUI[2]: this interface allows the monitoring of registrations and calls on the gatekeeper and provides endpoint information as well. Source code is available to modify for added functionality, if needed.
- Sample ACD application[3]: this interface allows the definition and management of groups of endpoints (so called agents) who will handle a large volume of calls for a single alias. The ACD will check which of the agents is qualified and available (not in another call and not logged off from ACD work) and informs the gatekeeper which agent will receive the call. If no agent is available the ACD will tell the gatekeeper to reject the call. All call routing logic is kept out of the gatekeeper to ensure stable operation while routing logic can be changed frequently.
- PHP GNUgk Status Monitor - v0.4[4]: this application allows monitoring of registered endpoints and calls in progress through a PHP web interface. Call disconnection is possible and further functionality is being developed. Source code is available.

6.2. Web Integration of SIP Services

In this chapter we will provide an overview of some added value services implemented using web interface. Serweb, the web interface for SIP Express Router described in Chapter 4, will be used as an example implementation of web-based added value services for SIP

Integration of SIP services with web interface is the most common scenario. Web interface is often used for provisioning of SIP products and for implementation of advanced services and web browser are available in vast majority of existing operating systems.

6.2.1. Click-to-Dial

Click-to-Dial, in other words, it a method of establishing a call between participants using web interface. It greatly simplifies dialing in that callers do not have to dial lengthy addresses and they keep their phonebooks separately from SIP phones. In its simplest form a user has a web page where he can enter SIP addresses of two users and SIP user agents of those two users get connected. We will focus on REFER-based click-to-dial.

REFER based click-to-dial scenario is based on the paradigm of intelligent end devices and dumb network. One

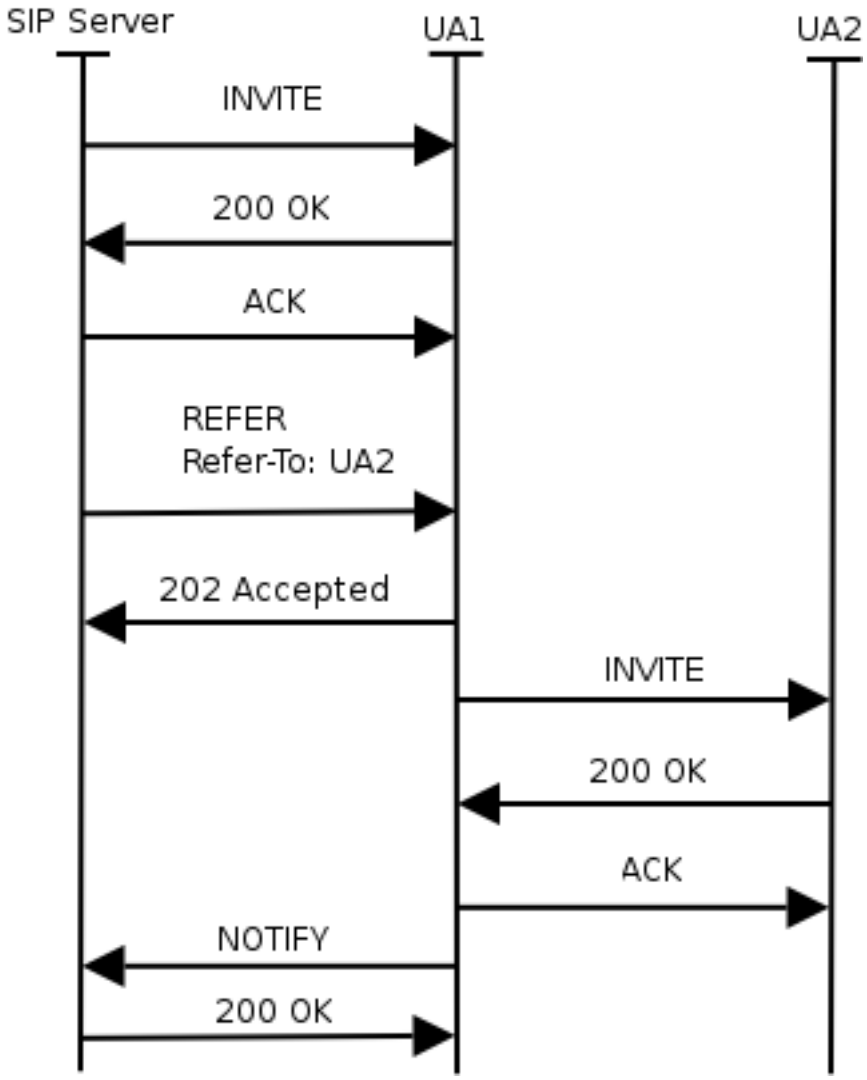
[1] http://www.cisco.com/en/US/customer/products/sw/iosswrel/ps1839/products_programming_reference_guide09186a0080153ef7.html
[2] http://www.gnugk.org/h323gui.html
[3] http://www.gnugk.org/h323-acd.html
[4] http://www.acefone.com/acefone/dlds/PHPgnugkstatus04.zip

of involved SIP user agent is asked to connect to the other and report to the server when it is done.

Drawback of this approach is that one of involved SIP user agents must support REFER SIP method which has been standardized recently, see RFC3515[5]. Big advantage that balances the previous drawback is that it is extremely easy to implement REFER based click to dial in the server.

Call-flow for REFER based click-to-dial is depicted on Figure 6.1.

Figure 6.1. REFER Based Click-to-Dial



First the SIP server sends an INVITE to one of the phones because phones usually do not accept REFER without prior invitation. The invite contains 0.0.0.0 as IP address in SDP because there is no remote phone (the message is sent by user agent within the SIP server which doesn't deal with media).

After that the server sends a REFER method which will ask the phone to send INVITE somewhere else. URI of the callee is passed to the phone in Refer-To header field of the REFER method.

The phone sends a NOTIFY method back to the server once the connection is established.

The click-to-dial feature allows creation of many advanced features, like phone-book in which you can click on an entry and your phone and phone of the person represented by the entry get connected. You can implement list of missed call in exactly the same way, clicking on an entry in the list will connect your phone with that person,

[5] http://www.ietf.org/rfc/rfc3515.txt

and many others possible scenarios.

6.2.2. Presence

It is also possible to display presence of SIP users in a webpage (for example in a phone-book). Displaying on-line status of subscribers allows callers to determine availability and willingness to have a conversation conveniently. The status may be shown for example in caller's phonebook or on callee's homepage. Linking on-line users to click-to-dial application greatly integrates telephony with web and introduced convenience to users.

When a SIP phone registers, the SIP server records this information into a database. The web server can then access the database to see if a user is online or offline.

Go to iptel.org[6], create a new account and insert some entries into your phone-book to see how does it work.

6.2.3. Missed Calls

Missed calls is a feature that allows users to display the list of call attempts that were made during the period when he was not online (registered in SIP). The list can be presented on a web page.

Recording missed calls has a lot of common with accounting. Servers doing accounting log some information when a call is successfully established and torn down. When a caller gets a negative final response to his call or doesn't receive any reply at all (timeout), then the server also records this event with a flag telling that it was a missed call. This information can be later used to compile the list of missed calls.

6.2.4. Serweb

Serweb is a web frontend for for SIP Express Router, see Section 4.6.1 for more details. Serweb creates a user interface for users of the proxy server, where they can manage their account, change their configuration and do many advanced things.

6.2.4.1. Installation

Serweb is a set of php scripts. To run it you will need apache web server with php and mysql support. Because SIP Express Router and serweb talk together using FIFO interface, the SIP proxy and the web server must be running on the same machine.

Get serweb from http://developer.berlios.de/projects/serweb and untar the archive, it is recommended not to untar it to the document root of your web server. Alternatively you can get serweb using CVS:

```
lexport CVSROOT=:pserver:anonymous:@cvs.berlios.de:/cvsroot/serweb
cvs login
cvs co iptel
```

6.2.4.2. Configuration

All the configuration of serweb is in config.php file in html subdirectory. You will need to configure the following:

- Host on which MySQL server is running: $this->db_host="localhost";
- Path of the user interface on the web server: $this->root_path="/";
- Root URI of the web server: $this->root_uri="http://www.foobar..."
- Path of serweb images on the web server: $this->img_src_path = $this->root_path."iptel_img/";
- Path of java script files of serweb: $this->js_src_path = $this->root_path."iptel_js/";
- Path of ccs files of serweb: $this->style_src_path = $this->root_path."iptel_styles/";

It is necessary to create some aliases in the configuration file of apache web server:

[6] http://iptel.org

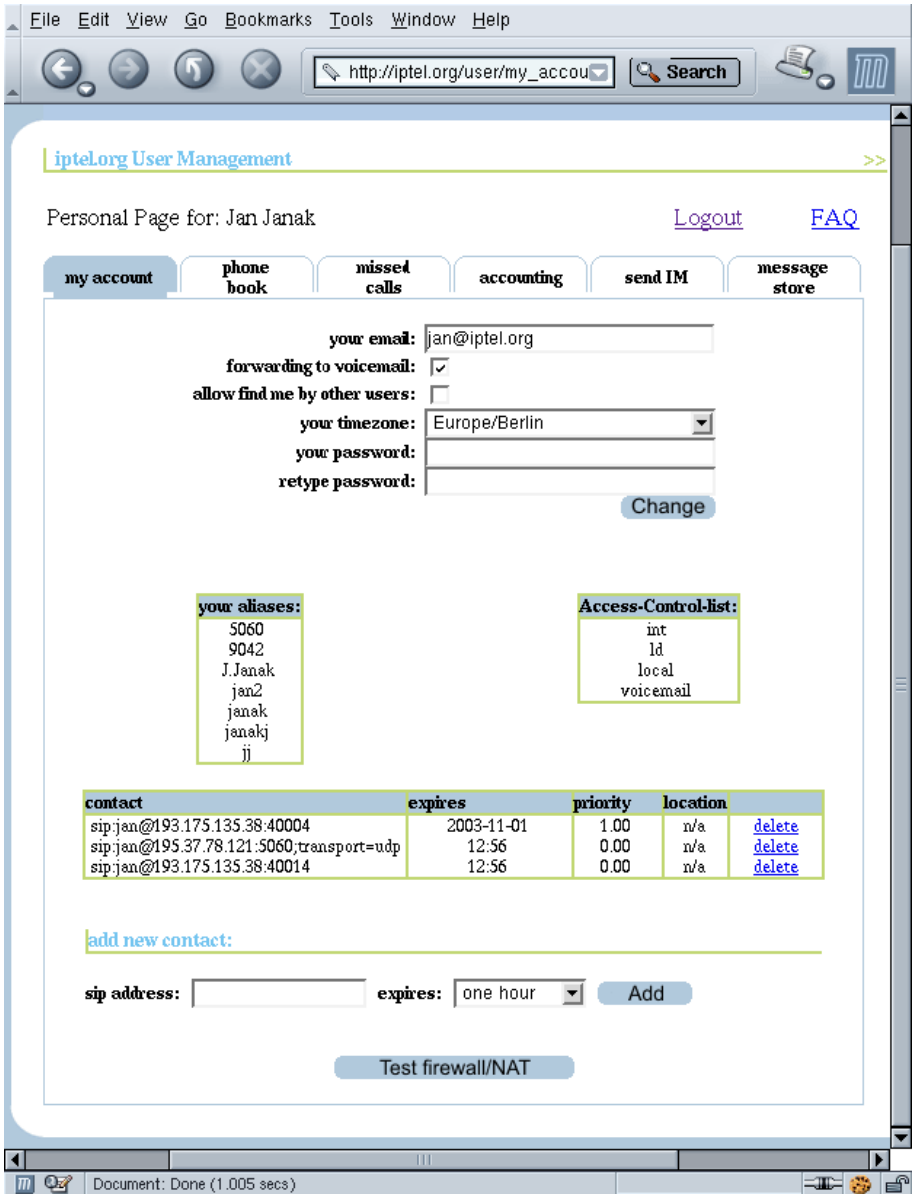
```
Alias /iptel_img "/var/www/iptel/html/img"
Alias /iptel_styles "/var/www/iptel/html/styles"
Alias /user "/var/www/iptel/html/user_interface"
Alias /admin "/var/www/iptel/html/admin"
```

Do not forget to update the directory path according to your real settings and make sure that you have `register_globals` and `short_open_tag` set to On in your `php.ini` file.

6.2.4.3. Operation

To login into serweb open http://<your_server>/user in your web browser. You will be prompted for username and password. The username and password is same as the one you are using in your SIP user agent to register at the server.

Figure 6.2. Serweb - My Account

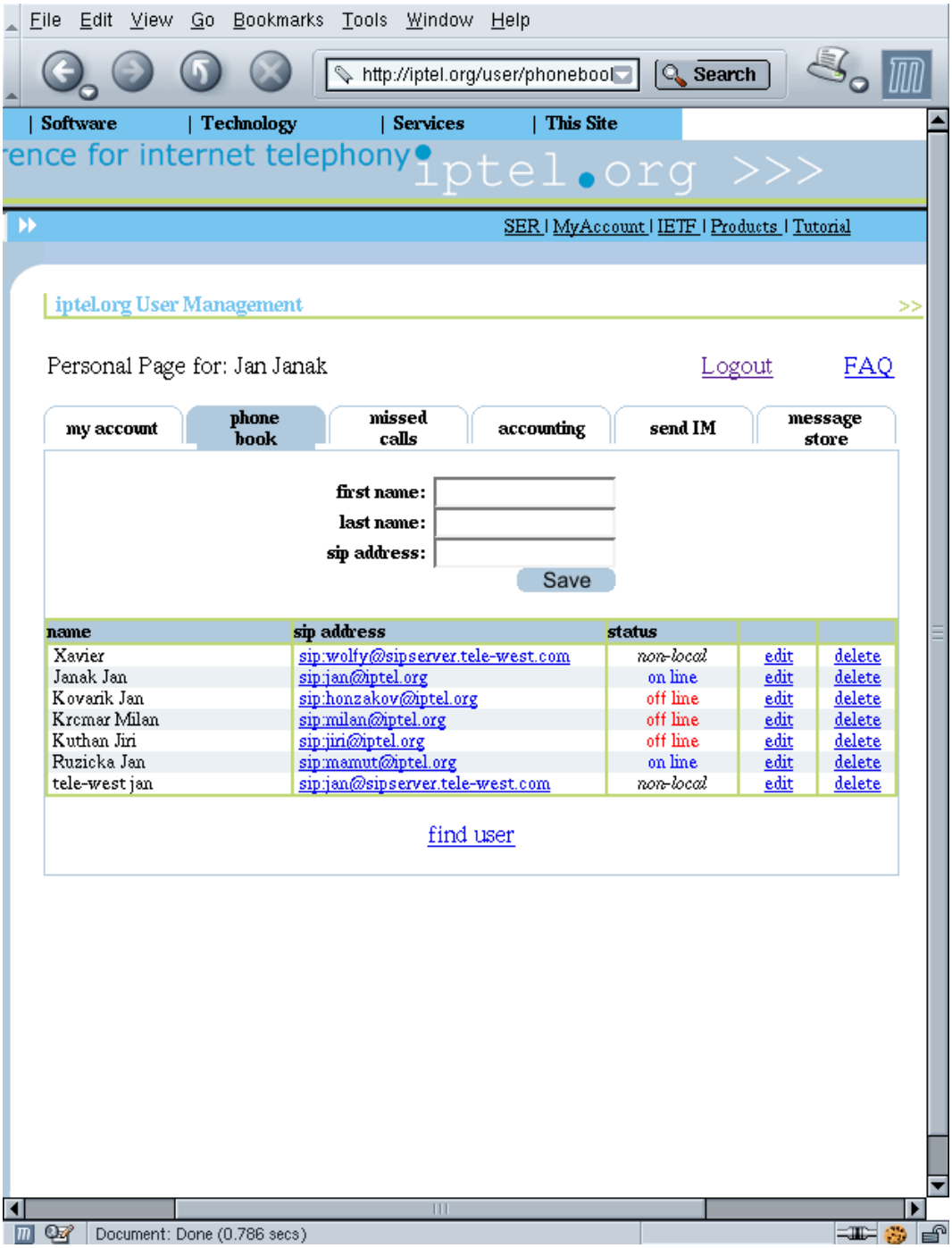


Plane “My Account”, see Figure 6.2 allows users to change their preferences and modified registered contacts.

They can also see aliases they created and permissions for calling to PSTN.

Users can also create their own phone book, see Figure 6.3. In the phonebook you can see presence status of each user. If the user is currently registered then you will see “on line” in the status column. If he is not registered then you will see “offline” and if the user doesn't belong to administrative domain of the server then you will see “non-local”.

Figure 6.3. Serweb - Phonebook



Clicking on the address of a user will establish a phone call between your and his phone, provided that your phone supports REFER, as described in Section 6.2.1.

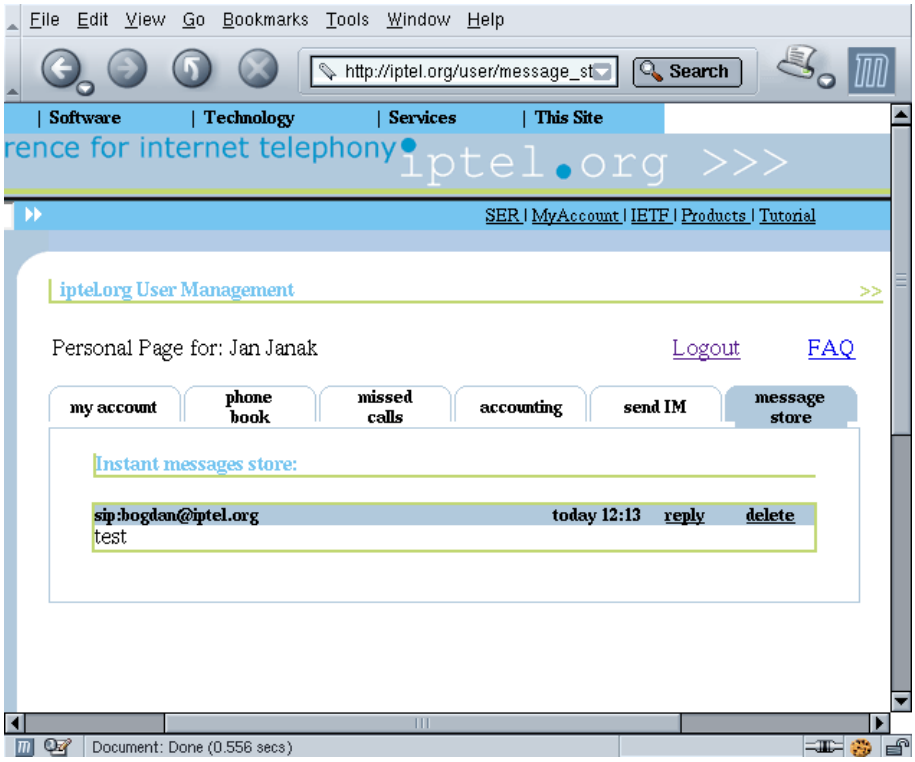
Missed calls plane, see Figure 6.4, allows users' to see their missed calls. Again, clicking on an entry will con-

nect you with that user and status describes presence of the user as described in the previous section.

Figure 6.4. Serweb - Missed Calls

When anyone sends a SIMPLE message to a user that is currently not online, the server will store the message and send it later when the recipient becomes online. In plane “message store”, see Figure 6.5 you can see all messages that are stored for you.

Figure 6.5. Serweb - Message Store



6.2.5. SIP Express Router Message Store

Message store has been implemented as a separate module for SIP Express Router. To use the module, you will need to load the module:

```
loadmodule "/usr/local/lib/ser/modules/msilo.so"
```

Configure address of the server (it will be used when sending stored messages) and URL of the database:

```
modparam("msilo", "registrar", "sip:registrar@<your_domain>")
modparam("msilo", "db_url", "sql://ser:passwd@dbhost/ser")
```

Then, when the server receives a MESSAGE requests and it can't deliver it because the recipient is offline, it will save the message:

```
if (!lookup("location")) {
    if (method == "MESSAGE") {
        if (!t_newtran()) {
            sl_reply_error();
            break;
        };

        if (m_store("0")) {
            t_reply("202", "Accepted for Later Delivery");
            break;
        };

        t_reply("503", "Service Unavailable");
```

```
        break;
    };
};
```

When lookup of recipient's location fails (the recipient is not registered), we create a new transaction (needed for msilo module), save the message using **m_store** command, and reply with “202 accepted”

Each time we call **save("location")** we have to check if the previously available user is registered again and if so then send stored messages. The following example shows how to do that:

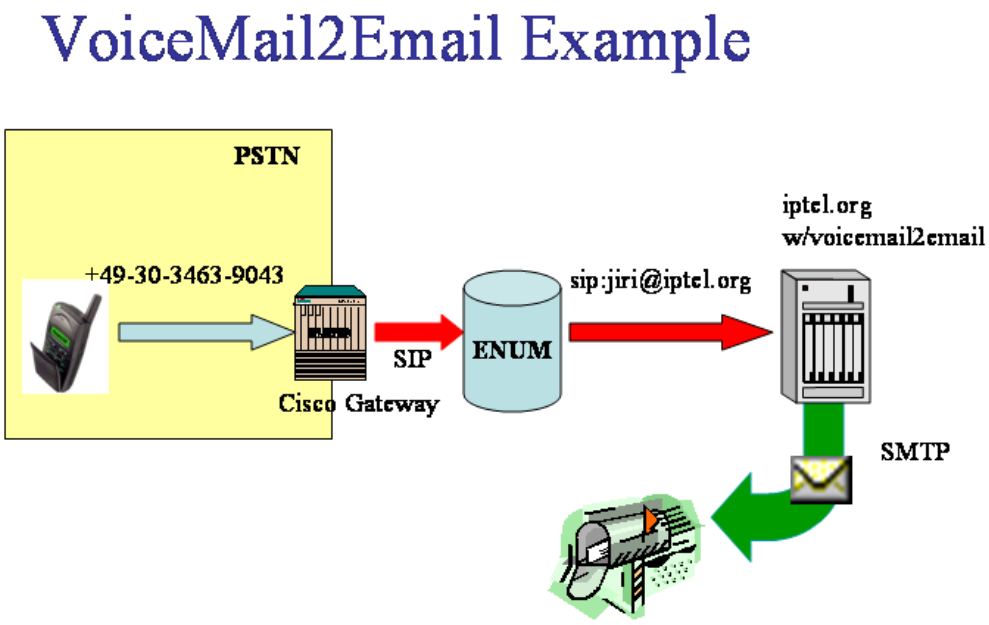
```
if (!save("location")) {
    sl_reply_error();
};

m_dump();
```

Command **m_dump** checks if the registering user has any stored messages and sends them if so.

6.3. Voicemail

Another Internet application which lends itself for integration with telephony is email. A traditional PSTN application which can be replaced with VoIP and email is voicemail. Traditional PSTN voicemail systems feature fairly inconvenient user interface for message retrieval: IVR (Interactive Voice Responder). Callers have to navigate through an automated voice menu, listen to lengthy announcements, type digits as prompted and be very patient to achieve very simple tasks. It is undoubtedly more convenient to deliver recorded messages to called party by email. The called party can then listen, store and process the received messages at his convenience. The following picture shows the data flow in a voicemail-2-email scenario. An open-source voicemail2email application, SEMS, is available from iptel.org[8].

Figure 6.6. Voicemail



[8] http://iptel.org

SEMS stands for SIP Express Media Server. It is an application framework that allows easily to build applications that deal with media streams. The framework itself provides only minimal functionality for accessing and manipulation of media streams and signalling. High level logic is stored in additional modules that can be dynamically loaded. Examples of such modules are voicemail, announcement server, and ISDN gateway.

Some other voicemail systems do exist, they include OpenAM which is available from the OpenH323 website[9] and a voicemail system built-in inside the VOCAL system. Unfortunately they are not easy to set-up and they are not yet ready to be used in a production environment if you need a completly integrated product, anyway they are working without bugs and are simple to be customized for small environment scenarios.

[9] http://www.openh323.org

Chapter 7. Global telephony integration

Abstract

World Gatekeeper hierarchy, dynamic routing, dialplans, ...

Leader: TZI Bremen

Contributors: Egon Verharen

7.1. Technology

Mechanisms and protocols for inter-domain routing

7.1.1. H.323 LRQ

7.1.2. H.225.0 Annex G

7.1.3. TRIP

7.1.4. SRV-Records

7.1.5. ENUM

NAPTRs

7.2. Today

Current state of the art in commercial products.

7.2.1. H.323

This would be some kind of gatekeeper hierarchy

7.2.2. SIP

This would be SRV-records

7.3. Migration

Towards decentralized architecture (DNS, ...), coexistance of different models

7.4. The future

SRV-Records, ENUM

Chapter 8. Regulatory / Legal considerations

Abstract

What regulations apply to IP telephony in european countries.

Leader: Karl-Franzens-Uni Graz

Contributors: All

8.1. Regulation of Voice over IP

8.1.1. In Europe

8.1.2. In other countries

(Overview)

8.2. Basic legal framework and problems

8.3. Problem of being a provider

8.4. Voice over IP and the definition of Voice Telephony

8.5. Licensing

8.6. Numbering

8.7. Interconnection

8.8. Unbundling

Appendix A. European IP Telephony Projects

Abstract

List and evtl. brief description of IP telephony projects (past and present) in Europe.

Leader: FhG Fokus

Contributors: Valentino Cavalli

Appendix B. IP Telephony Hardware/Software

Abstract

We classify the product list by product types, something like we maintain at (http://www.iptel.org/info/products[1]: soft phone, hard phone, servers, gateways, accounting, testing, firewalls, miscellaneous.

Leader: FhG Fokus

Contributors: Valentino Cavalli

Primarily interesting is to collect operational experience with the devices.

Write about which devices you have had in your trials/deployments, how they worked, how was tech-support responsiveness of the vendors, what was broken, etc., etc.

I suggest you included at least the following pieces of data in your reports:

- Product Name:
- Product Type: {softphone| hardphone | servers | gateways | accounting | testing | firewalls | miscellaneous }
- Product URL:
- Vendor:
- Supported signaling protocols:
- Platforms (OS, hardware): (n/a for stand-alone boxes)
- Overall Evaluation (tech-support, affordability, etc.)
- Operational Experience:

[1] http://www.iptel.org/info/products/