IP Telephony Cookbook

> TERENA REPORT //

> / MARCH, 2004

TRANS-EUROPEAN RESEARCH AND EDUCATION NETWORKING ASSOCIATION

The IP Telephony Cookbook was created through the IP Telephony project as a reference document for setting up IP Telephony solutions at university campuses and NRENs. The project started in April 2003 and ran until February 2004. The Cookbook provides an overview of available and future IP Telephony technologies, scenarios for IP Telephony deployment and infrastructures, guidelines on protocols, service set-ups and connection to a global 'dialling plan'. Furthermore, the Cookbook reports on the interoperability of equipment, existing IP Telephony projects and regulatory aspects.

The project was carried out by the University of Pisa, Italy, TZI-University of Bremen and FhG FOKUS, Germany, with contributions from CESNET, GRNET, SURFnet and the University of Graz.

Funding to the project was provided by TERENA, ARNES, CARNet, CYNET, SUNET and UKERNA. Representatives of the funding organisations were members of the Project Review Committee.

ISBN 90-7759-08-6

Production: TERENA Secretariat
Design: Eva de Lange
Printing: GraphicResult

AUTHORS:

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

CONTENTS //

1. INTRODUCTION 7

1.1 Goal
1.2 Reasons for writing this document
1.3 Contents
1.4 How to read this document
1.5 Techno-economic aspects of moving from classic telephony to VoIP

2. TECHNOLOGICAL BACKGROUND 11

2.1 Components
2.1.1 Terminal
2.1.2 Server
2.1.3 Gateway
2.1.4 Conference bridge
2.1.5 Addressing
2.2 Protocols
2.2.1 H.323
2.2.2 SIP
2.2.3 Media gateway control protocols
2.2.4 Proprietary signalling protocols
2.2.5 Real Time Protocol (RTP) and Real Time Control Protocol (RTCP)

3. IP TELEPHONY SCENARIOS 52

3.1 Introduction
3.2 Scenario 1: Long-distance least cost routing
3.2.1 Least cost routing - an example of an implementation
3.3 Scenario 2: Alternatives to legacy PBX systems
3.3.1 Scenario 2a: IP Phones without a PBX system
3.3.2 Scenario 2b: Integration of VoIP with legacy PBX systems
3.3.3 Scenario 2c: Full replacement of legacy PBX systems
3.4 Scenario 3: Integration of VoIP and videoconferencing
3.4.1 Integrating voice and videoconferencing over IP - an example

4. SETTING UP BASIC SERVICES 64

4.1 General concepts
4.1.1 Architecture
4.1.2 Robustness
4.1.3 Management issues
4.2. Dial plans
4.3 Authentication
4.3.1 Authentication in H.323

4.3.2 Authentication in SIP
4.4 Examples
4.4.1 Example 1: simple, use IP Telephony like legacy telephony
4.4.2 Example 2: complex, full-featured
4.5 Setting up H.323 services
4.5.1 Using a Cisco Multimedia Conference Manager (MCM Gatekeeper)
4.5.2 Using a RADVI SION-Enhanced Communication Server (ECS Gatekeeper)
4.5.3 Using an OpenH.323 Gatekeeper - GNU Gatekeeper
4.6 Setting up SIP services
4.6.1 Operation of SIP servers
4.6.2 SIP express router
4.6.3 Asterisk
4.6.4 VOCAL
4.7 Firewalls and NAT
4.7.1 Firewalls and IP Telephony
4.7.2 NAT and IP Telephony
4.7.3 SIP and NAT

5. SETTING UP ADVANCED SERVICES **135**
5.1 Gatewaying
5.1.1 Gateway interfaces
5.1.2 Gatewaying from H.323 to PSTN/ISDN
5.1.3 Gatewaying from SIP to PSTN/ISDN
5.1.4 Gatewaying from SIP to H.323 and vice versa
5.1.5 Accounting gateways
5.2 Supplementary services
5.2.1 Supplementary services using H.323
5.2.2 Supplementary services using SIP
5.3 Multipoint conferencing

6. SETTING UP VALUE-ADDED SERVICE **173**
6.1 Web integration of H.323 services
6.1.1 RADIUS-based methods
6.1.2 SNMP-based methods
6.1.3 Cisco MCM GK API
6.1.4 GNU GK status interface
6.2 Web integration of SIP services
6.2.1 Click-to-dial
6.2.2 Presence
6.2.3 Missed calls
6.2.4 Serweb
6.2.5 SIP express router message store
6.3 Voicemail

7. INTEGRATION OF GLOBAL TELEPHONY **184**
7.1 Technology
7.1.1 H.323 LRQ

7.1.2 H.225.0 Annex G
7.1.3 Telephony routing over IP (TRIP)
7.1.4 SRV-records
7.1.5 ENUM
7.2 Call routing today
7.2.1 SIP
7.2.2 Using H.323
7.3 Utopia: setting up global IP Telephony
7.4 Towards Utopia
7.4.1 Call routing assistant

8. REGULATORY / LEGAL CONSIDERATIONS **200**
8.1 Overall
8.2What does regulation mean for Voice over IP?
8.3 Regulation of Voice over IP in the European Union
8.3.1 Looking back into Europe's recent history in regulation
8.3.2 The new regulatory framework - technological neutrality
8.3.3 The new regulatory framework - an overview
8.3.4 Authorisation system instead of licensing system
8.3.5 Numbering
8.3.6 Access
8.3.7 Interconnection
8.3.8 Quality of Service
8.4Voice over IP in the United States
8.5Conclusion and summary

// **ANNEX A.** European IP Telephony Projects **210**
A.1Evolute
A.26Net
A.3Eurescom P1111 (Next-Gen open Service Solutions over IP (N-GOSSIP)
A.4HITEC
A.5The GRNET/RTS project
A.6SURFWorks
A.7VC Stroom
A.8Voice services in the CESNET2 network

// **ANNEX B.** IP Telephony Hardware/Software **214**
B.1Softphones
B.2Hardphones
B.3Servers
B.4Gateways
B.5Testing
B.6Miscellaneous

// **GLOSSARY** **226**

1 Introduction //>

> 1.1 Goal

The IP Telephony Cookbook is a reference document addressing technical issues for the setup of IP Telephony solutions. Its goal is to provide the user community with guidelines and information about the IP Telephony world and everything related to it. Since the Cookbook is intended to be a technical document, the main target audience are the network engineers and system administrators at universities and national research and education networks (NREN); however, university students and researchers may find it useful, both for enriching their technological background as well as for finding information about advanced research topics and projects in the European community.

> 1.2 Reasons for writing this document

Members of the NREN community asked TERENA to start an investigation into IP Telephony in September 2001. The response was very positive and suggestions were made to co-ordinate the creation of a cookbook with recommendations for setting up IP Telephony solutions at university- and national-level, with information about protocols and the interoperability of equipment as well as about 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 compose this document, The IP Telephony Cookbook.

> 1.3 Contents

The IP Telephony Cookbook is divided into chapters, which guide the reader through increasing levels of knowledge of the IP Telephony world. This first chapter contains introductory information and gives details of the contents of the Cookbook, useful tips on how to read this document and techno-economic considerations. Chapter 2 explains the technological background needed in order to understand the topics addressed in the rest of the Cookbook. This chapter describes the basic IP Telephony components and gives an overview of the IP Telephony protocols. Chapter 2 ends with additional considerations on call routing and perspectives about the future. Chapter 3 gives a high-level overview of scenarios a user may face when building an IP Telephony environment. Details are given to explain what a particular scenario is about, what is needed in order to deploy it and what needs it is serving. 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 to learn how to set up basic services, advanced services (still telephony-centric) and value-added services (with respect to classic telephony service). Chapter 7 is about the integration of global telephony, describing the technological solutions available for the integration of global IP Telephony and the successful replacement of classic telephony. Chapter 7 reports on today's situation, as well as migration and future trends. The last chapter contains the regulatory/legal considerations users have to be aware of when moving from classic telephony to IP Telephony. The topics here relate to the regulation of IP Telephony in Europe and in other countries outside the European Union. A large number of legal issues for classic telephony are detailed, from licensing to unbundling, and their mapping to the IP world. Finally, the IP Telephony Cookbook contains two annexes. Annex A lists and describes current and future IP Telephony Projects in Europe. Annex B gives the reader useful information about IP Telephony hardware and software, reporting 'hands on' experience (i.e., how the devices performed, how good tech-support was, what were the workarounds for some of the problems faced, 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, so that they can 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 reader should identify himself as belonging to one of the following three groups:

- readers who have no knowledge of IP Telephony;
- readers who have basic knowledge of IP Telephony;
- readers who have advanced knowledge of IP Telephony.

Readers 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. Readers who are interested in setting up an IP Telephony service should read Chapter 3 to have a clear picture of the possible scenarios offered by IP Telephony and target the one best-suited to the needs of their environment. The second group of readers may skip the previously-mentioned chapters, but Chapter 3 may be of some interest to them; the main focus of this group of users is more likely to be in Chapter 4 and Chapter 5 which give 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 problems of an IP Telephony architecture that is widely distributed across multiple sites and organisations (Chapter 7). All three groups of users may find useful information in Chapter 8 and European project information in Annex A. Last but not least, the list of products and testing experience reported in Annex B is a must for all users who do not want to risk making the wrong choices in a buying decision.

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

Many institutions are facing investment decisions with respect to replacing or expanding their existing telephony infrastructure, which currently consists mainly of large PBXs with proprietary phones and interfaces. As is there such a clear trend to replace old-style (TDM) PBXs with IP Telephony ones, it is important that there is a guide on how to attach such an IP Telephony solution to the existing network. 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. Even where financial constraints are not the driving force, the potential for enhancing IP Telephony with additional services that support scientific co-operation makes IP Telephony an attractive solution.

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

Enhanced speech quality

The PSTN (and most PBXs) are limited to 3.1 kHz, 8-bit/sample audio. It is likely that future IP phones can provide CD quality and possibly even stereo audio. Even where the additional bandwidth required for this extreme level of quality cannot be provided; modest codecs such as G.722 (7 kHz speech bandwidth) can be used to provide better quality than conventional telephony;

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 than traditional telephony. Integrating telephony with location-based computing and group-awareness systems can make the communication partners much more 'available', or provide the means to transfer 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 consider the cost of phone installations high enough to force their employees to share phones in a common office, again, not necessary when workstation-based IP Telephony is used;

Improved mobility

It is very easy to move an IP phone to another room. There is no need to deal with ports on the PBX and change dial numbers. Simply plugging it into an ethernet socket in a new room makes it available;

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, whereas the standards are not yet there for coupling traditional phones and workstations;

New services

As IP Telephony evolves, it can be used to provide new services (like user-defined call processing) or to integrate existing concepts, e.g., Presence, Location Awareness or Instant Messaging. Because of the open standards available for these services, they need not to be limited to vendor-specific solutions. In other words, it can be much easier to deal with issues such as CTI (Computer Telephony Integration) and so pave the way to a completely new way of understanding telephony;

Research

As mentioned before, the protocols and standards used for IP Telephony are open and publicly available. This allows research institutions to work on their own services and solutions.

It is important to point out that before introducing IP Telephony into the network of an organisation; several issues unknown to the old telephone system have to be taken into account. A rough, non-exhaustive list may include addressing (special subnet/VLAN for phones), Quality of Service (QoS), security, positioning of gateways, interfacing of firewalls and, last but not least, maintenance of the system (backups, spares, etc., - something not very common in the legacy PBX world).

With regard to the economic aspects, the ‘packetisation’ of voice using Voice over IP has given rise to new international 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 are required between traditional circuit-switched PSTN carriers. Moreover, since packet networks are software driven, they can be configured more dynamically than traditional PSTN networks. For example, with a global voice over packet network, new destinations are available to all users on the network, without the need for constant additional investment.

IP Telephony telecommunications companies may expand the availability of services to a wider audience. IP Telephony technologies can be used to build 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 timeframe than would be possible with conventional PSTN networks. At the same time, not having to build extensive infrastructure provides the motivation for many companies to migrate to IP Telephony architectures.

2 Technological Background }

This chapter provides technical background information about the protocols and components used in IP Telephony. It introduces the relevant component types, gives detailed information about H.323, SIP and RTP as well as information about media gateway control and vendor -specific protocols.

} 2.1 Components

An IP Telephony infrastructure usually consists of different types of components. This section gives an overview of typical components without describing them in a protocol-specific context.

} 2.1.1 Terminal

A terminal is a communication endpoint that terminates calls and their media streams. Most commonly, this is either a hardware or a software telephone or videophone, possibly enhanced with data capabilities. There are terminals that are intended for user interaction and others that are automated, e.g., 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 been assigned one or more addresses (see Section 2.1.5), which others will use to dial to it. If IP Telephony servers are used, a terminal registers the addresses with its server.

} 2.1.2 Server

Placing an IP Telephony call requires at least two terminals, and the knowledge of the IP address and port number of the terminal to call. Obviously, forcing the user to remember and use IP addresses for placing calls is not ideal and dynamic IP addressing schemes (DHCP) make this requirement even more intolerable.

As mentioned before, terminals usually register their addresses with a server. The server stores these telephone addresses along with the IP addresses of the respective terminals, and is thus able 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 (Call Processing Language) scripts or skill-based routing (e.g., route calls to 'WWW-Support' to a list of persons who are tagged to be responsible for this subject).

Finally, a telephony server is responsible for authenticating registrations, authorising calling parties and performing the accounting

{ 2.1.3 Gateway

Gateways are telephony endpoints that facilitate calls between endpoints that usually would not interoperate. Usually this means that a gateway translates one signalling protocol into another (e.g. SIP/ISDN signalling gateways), but translating between different network addresses (IPv4/IPv6) or codecs (media gateways) can be considered gatewaying as well. Of course, it is possible that multiple functionalities exist 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 the means to have 3-point or multi-point conferences that can either be 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 himself and the called party. Ideally, such an identifier should be independent of the user's physical location. The network should be then responsible for finding the current location of the called party. A specific user may define to be reached by multiple contact address identifiers.

Regular telephony systems use E.164 numbers (the international public telecommunication numbering plan). An identifier is composed of up to fifteen digits with a leading plus sign, for example, +1234565789123. When dialling, 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.

The first IP Telephony systems used the IP addresses of end-point devices as user identifiers. Sometimes they are still used now. However, IP addresses are not location-independent (even if IPv6 is used) and they are hard to remember (especially if IPv6 is used) so they are not suitable as user identifiers.

Current IP Telephony systems use two kinds of identifiers:
- URIs (RFC2396);
- Numbers (E.164).

Some systems tried to use names (alpha-numeric strings), but this 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 (h.323, RFC3508) or telephone numbers (draft-ietf-iptel-rfc2806bis-02). 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 found 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 suited for an Internet world relying on domain names. Therefore, the ENUM system was invented, using adapted phone numbers as domain names. ENUM is described 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 standardisation on video telephony and videoconferencing 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 analogue Public Switched Telephone Network (PSTN) using modem technology (H.324), and over the stillborn 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 the 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 base on which the industry could converge. The initial focus was clearly on local network environments, because 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 applications. The first major additions to this end were included in H.323 version 2, approved by the ITU-T in January 1998. In September 1999, H.323v3 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 as well as making it meet requirements in enterprise environments. Moreover, many new enhancements were introduced into the H.323 protocol. Version 4 was approved on November 17, 2000 and contains enhancements in a number of important areas, including reliability, scalability, and flexibility.

New features help facilitate more scalable Gateway and MCU solutions to meet the growing market requirements. H.323 has been the undisputed leader in voice, video, and data conferencing on packet networks, and Version 4 endeavours to keep H.323 ahead of the competition.

{ 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 (in particular to ISDN-based video telephony and to PSTN/ISDN/GSM for voice communication).

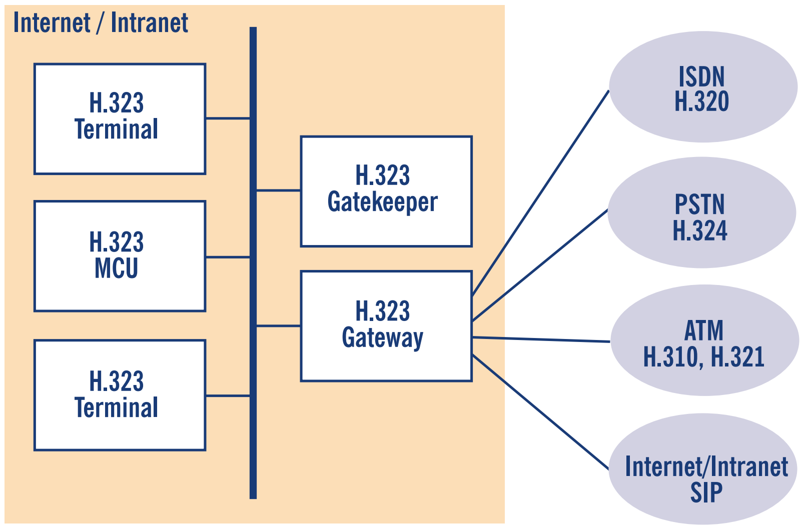


Figure 2.1 Scope and components defined in H.323

H.323 defines a number of functional / logical components as shown in Figure 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 utilisation / 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 added value, e.g., act as a

conferencing bridge or offer supplementary call services. An H.323 Gatekeeper can also be equipped with the proxy feature. Such a feature enables the routing through the gatekeeper of the RTP traffic (audio and video) and the T.120 traffic (data), so no traffic is directly exchanged between endpoints. (It could be considered a kind of IP-to-IP gateway that can be used for security and QoS purposes);

- **Multipoint Controller (MC)**
  A Multipoint Controller is a logical entity that interconnects the call signalling 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 is simply 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 Signalling protocols

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

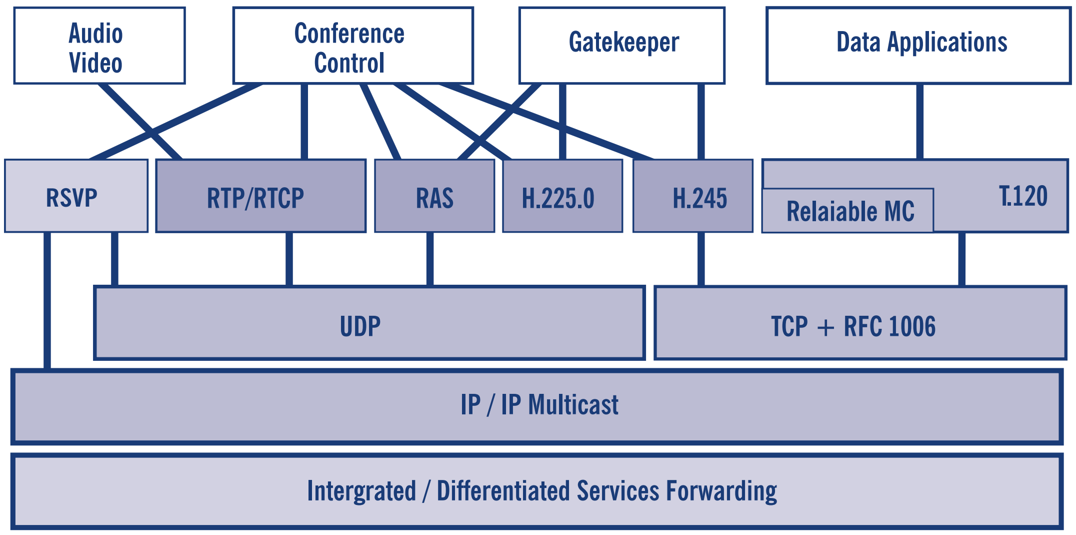


Figure 2.2 H.323 protocol architecture

For basic call signalling 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 utilise 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 utilisation after call termination. RAS provides mechanisms for user/endpoint authentication and call authorisation;
- **H.225.0 Call Signalling**
  The call signalling channel is used to signal call setup intention, success, failures, etc, as well as to carry operations for supplementary services (see below). Call signalling messages are derived from Q.931 (ISDN call signalling); however, simplified procedures and only a subset of the messages are used in H.323. The call signalling channel is used end-to-end between calling party and called party and may optionally run through one or more gatekeepers (the call signalling models are later described in the 'Signalling models' Section).

  Optimisations: 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 will 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.

  The 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 that 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) and enables confidential media exchange and 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 gatekeepers, or directly between calling party and called party (please refer to the 'Signalling 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.

  Optimisations:
  The conference control channel is also subject to optimisations. Per default, it is transported over an exclusive TCP connection but it may also be tunnelled within the signalling connection

(H.245 tunnelling). Other optimisations deal with the call setup time. The last chance to start an 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 signalling, 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.323v2, 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, 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 and H.323.Annex F).

{ 2.2.1.3 Gatekeeper discovery and registration

An H.323 endpoint usually registers with a gatekeeper that provides basic services like address resolution for calling the other endpoints. 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 manual configuration. While there is no need for a gatekeeper request (GRQ) to be sent to the preconfigured gatekeeper, some products need this protocol step. If a gatekeeper receives a GRQ via unicast, it must either confirm (GCF) the request or reject it (GRJ).

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

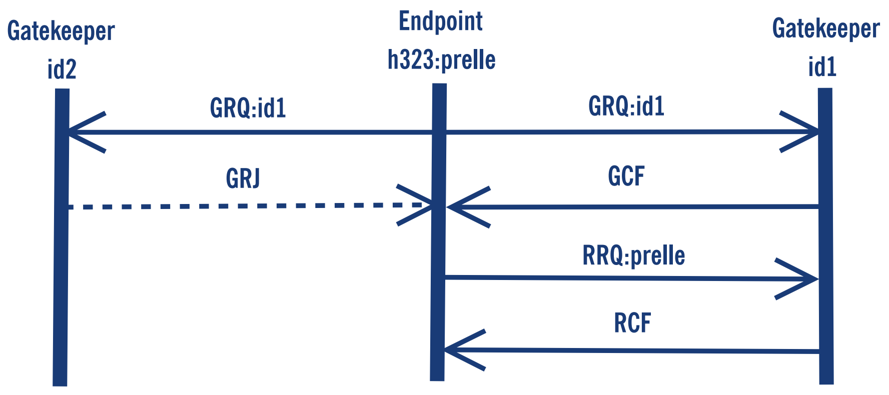


Figure 2.3 Discovery and registration process

After the endpoint discovers 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 the 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 will last. This value can be overwritten by gatekeeper policies.

The gatekeeper checks the requested registration information and confirms the (possibly modified) values (RCF). It may also reject a registration request because of, for example, invalid addresses. In the case of a confirmation, the gatekeeper assigns a unique identifier to the endpoint, which will be used in subsequent requests to indicate that the endpoint is still registered.

2.2.1.3.1 Addresses and registrations

H.323 defines and utilises several address types. The one most commonly used and derived from the PSTN world is the dialled digit address, which is defined as a number dialled by the endpoint. It does not include further information (e.g., about the dial plan) and needs to be interpreted by the server. The server might convert the dialled number into a party number that includes information about the type of number and the dial plan.

To provide alphanumeric or name dialling, H.323 supports H.323-IDs that represent either usernames or e-mail-like addresses, or the more general approach of URL-ID which represent any kind of URL.

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

2.2.1.3.2 Updating registrations

A registration expires after a defined time and must therefore be refreshed i.e., kept alive by subsequent registrations which include the previously-assigned endpoint identifier. To reduce the registration overhead of regular registrations, H.323 supports **KeepAlive** registrations that contain only the previously-assigned endpoint identifier. Of course, these registrations may only be sent if the registration information is unchanged.

Endpoints requesting the registration of large numbers of addresses would exceed the size of a UDP packet, so H.323v4 supports **Additive Registration**, a mechanism that allows an endpoint to send multiple registration requests (RRQ) in which the addresses do not replace existing registrations but are submitted in addition to them.

{ 2.2.1.4 Signalling models

Call signalling messages and H.245 control messages may be exchanged either end-to-end between calling party and called party or through a gatekeeper. Depending on the role the gatekeeper plays in the call signalling and in the H.245 signalling, the H.323 specification foresees three different types of signalling models:

- **Direct signalling**
  With this signalling 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 signalling**
  With this signalling model, H.225.0 RAS and H.225.0 call signalling 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 signalling and H.245 Conference Control messages are routed through the gatekeeper and only the media streams are directly exchanged between the two endpoints.

The following sub-sections detail each signalling model. The figures displayed in this section apply both to the use of a single gatekeeper and to the use of a gatekeeper network. Since the signalling model is decided by the configuration of the endpoint's gatekeeper and applies to all the messages the gatekeeper handles, the extensions to the multiple gatekeeper are straightforward (they simply apply the definition of the signalling model described in the itemised list above to each gatekeeper involved), except for the location of zone-external targets (described later in the 'Locating zone external targets' section). Message exchanges in any of the figures in this section are not reported, as the figures are intended to remain bounded in the ellipse where the H.323 Gatekeeper is depicted. Also, it is described in the 'Locating zone external targets' section. Please note that there is no indication about the call termination in the sub-section of each signalling model. Please refer to the 'Communication phases' Section for details.

The direct signalling model is depicted in Figure 2.4. In this model, the H.225.0 Call Signalling and H.245 Conference Control messages are exchanged directly between the call terminals. As shown in the figure, the communication starts with an ARQ (**Admission ReQuest**) message sent by the calling party (which may be either a terminal or a gateway) to the gatekeeper. The ARQ message is used by the endpoint to request access to the packet-based network from 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 signalling part of the call begins with the transmission of the SET UP message from the calling party to the called party. The transport address of the SET UP message (and of all the H.225.0 call signalling messages) is retrieved by the calling party from the **destCallSignalAddress** field carried inside the ACF received. In the case of the direct signalling model, it is the address of the destination endpoint. Upon receiving the SET UP message, the called party starts its H.225.0 RAS procedure with the gatekeeper. If successful, a CONNECT message is sent back to the calling party to indicate acceptance of the call. Before sending the CONNECT message, two other messages may be sent from the called party to the calling party (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.

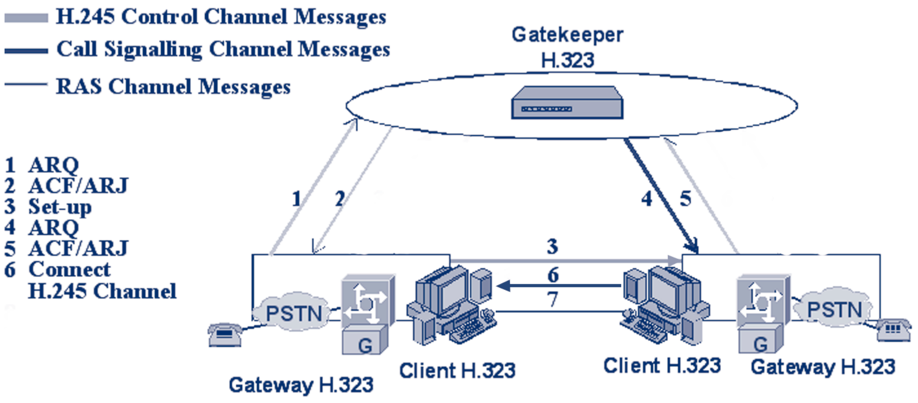


Figure 2.4 Direct signalling model

The CONNECT message closes the H.225.0 call signalling part of the call and makes the terminals starting the H.245 conference control one. In such call mode, 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 SIGNALLING messages;
- manage the logical channels used to transport the audiovisual and data information;
- 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;
- carry various control and indication signals;
- control the bit rate of individual logical channels and the whole multiplex, with the MULTIPLEX TABLE SIGNALLING messages;
- 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 signalling model

The gatekeeper-routed call signalling model is depicted in Figure 2.5. In this model, the H.245 Conference Control messages are exchanged directly between the call termination clients. With each call, the communication starts with an ARQ message (**Admission ReQuest**) sent by the calling party to its gatekeeper. The ARQ message is used by the endpoint to request access to the packet-based network from 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 signalling part of the call begins with the transmission of the SET UP message from the calling party to its gatekeeper. The transport address of the SET UP message (and of all the H.225.0 call signalling messages) is retrieved by the calling party from the **destCallSignalAddress** field, carried inside the ACF received. In the case of the gatekeeper-routed call signalling 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 called party 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, this message is also sent to the called endpoint's gatekeeper which is in charge of forwarding it to the calling party endpoint (either directly or using the gatekeeper network). Before sending the CONNECT message, two other messages may be sent from the called party to its gatekeeper (those two messages are not depicted in the figure since only mandatory messages are reported):

- **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.

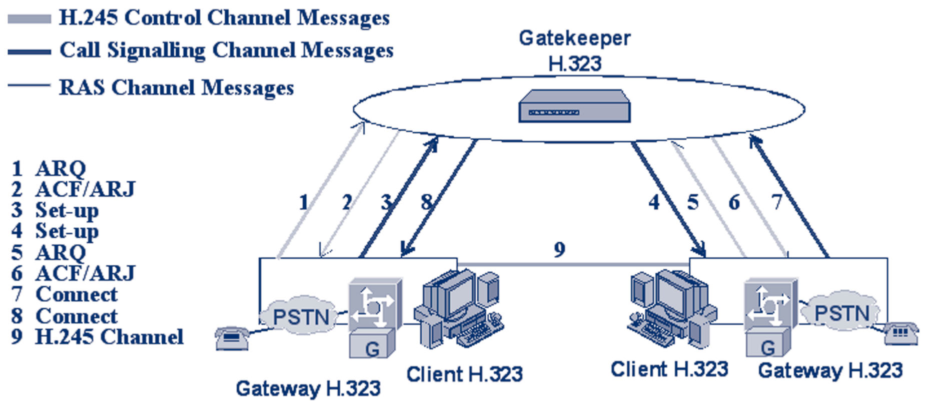


Figure 2.5 Gatekeeper-routed call signalling model

The two optional messages listed above are then forwarded by the gatekeeper (or by the gatekeeper network) to the calling party. After receiving the CONNECT message, the calling party starts the H.245 Conference Control channel procedures directly with the called party (the correct h245Address was retrieved from the CONNECT message itself). The scope of the H.245 Conference Control channel procedure is the same as is detailed above. Please refer to the 'Direct signalling 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 clients. For each call, the communication starts with an ARQ (**Admission ReQues**t) message sent by the calling party 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 signalling part of the call begins with the transmission of the SET UP message from the calling party to its gatekeeper. The transport address of the SET UP message (and of all the H.225.0 call signalling messages) is retrieved by the calling party 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 called party 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, this message is also sent to the called endpoint's gatekeeper, which is in charge of forwarding it to the calling party endpoint (either directly or using the gatekeeper network). Before sending the CONNECT message, two other messages may be sent from the called party to its gatekeeper (those two messages are not depicted in the figure since only mandatory messages are reported):

- **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.

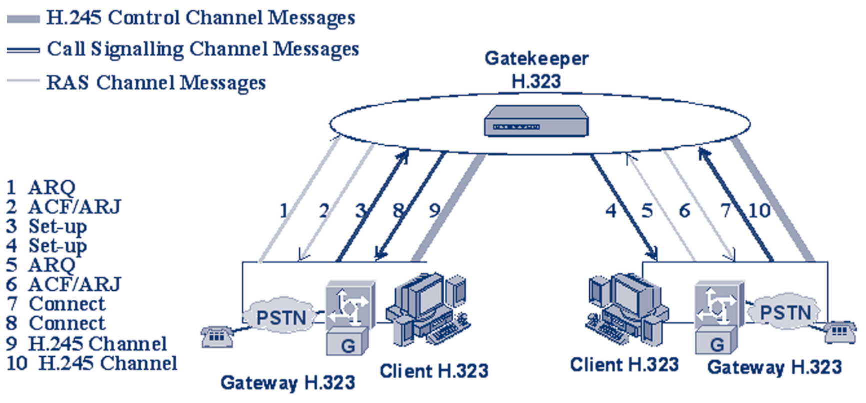


Figure 2.6 Gatekeeper-routed H.245 control model

The two optional messages listed above are then forwarded by the gatekeeper (or by the gatekeeper network) to the calling party. After receiving the CONNECT message, the calling party starts the H.245 Conference Control channel procedures with its gatekeeper (the correct h245Address was retrieved from the CONNECT message itself). All of 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 scope of the H.245 Conference Control channel procedure is the same as is detailed above. Please refer to the 'Direct signalling model' Section for details.

{ 2.2.1.5 Communication phases

In a H.323, communication may be identified in five different phases:
- Call set up;
- Initial communication and capability exchange;
- Establishment of audiovisual communication;
- Call services;
- Call termination.

2.2.1.5.1 Call setup

Recommendation H.225.0 defines the call setup messages and procedures detailed here. The recommendation foresees that requests for bandwidth reservation should take place at the earliest possible phase. Unlike other protocols, there is no explicit synchronisation between two endpoints during the call setup procedure (two endpoints can send a SET UP message to each other at exactly the same time). Actions to be taken when problems of synchronisation during the exchange of SET UP messages arise are resolved by the application itself. Applications not supporting multiple simultaneous calls should issue a busy signal when they have an outstanding SET UP message, while applications supporting multiple simultaneous calls issue a busy signal only to the same endpoint to which they sent an outstanding SET UP message. Moreover, an endpoint should be capable of sending the ALERTING messages. ALERTING means 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 originates the ALERTING message and only when the application has already alerted the user. If a gateway is involved, the gateway sends 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 four 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 setup phase may have different realisations:

- **basic call setup when neither endpoint are registered**
  In this call setup the two endpoints communicate directly;
- **both endpoints registered to the same gatekeeper**
  In this call set up the communication is decided by the signalling model configured on the gatekeeper;
- **only calling endpoint has gatekeeper**
  In this call setup only the calling party sends messages to the gatekeeper depending on the signalling models configured while the called party sends the messages directly to the calling party endpoint;
- **only called endpoint has gatekeeper**
  In this call setup only the called party sends messages to the gatekeeper depending on the signalling models configured while the calling party 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 signalling 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 setup with Fast Connect procedure**
  In this call set up, the media channels are established using 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 advise it is going to use the **Fast Connect** procedure).

This kind of element contains, among the other things, a sequence of all of the parameters necessary to immediately open and begin transferring media on the channels. The **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). The **Fast Connect** procedure may be refused with any H.225.0 call signalling 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 the exchange of capabilities and the opening of media channels. Moreover, the **Fast Connect** procedure allows more information for the scope of 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 setup;
- **call setup with an MCU**
  When an MCU is involved, all endpoints exchange call signalling with the MCU (and with the interested gatekeepers, if any). No changes are foreseen between an endpoint and the MCU call setup since it proceeds the same as the endpoint-to-endpoint;
- **broadcast call setup**
  This kind of call setup 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, 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 are neither started nor closed if CONNECT does not arrive. An H.245 Control Channel can also be opened on reception of ALERTING or CALL PROCEEDING messages) or when an endpoint sends RELEASE COMPLETE. H.323 endpoints support the capabilities exchange procedure of H.245. The H.245 TERMINALCAPABILITYSET message is used for the exchange of endpoint system capabilities. This message is the first H.245 message sent.

The master-slave determination procedure of H.245 must be supported by H.323-compliant endpoints. In cases of multipoint conferencing (MC), capability is present in more than one endpoint and 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 bi-directional channels for data.

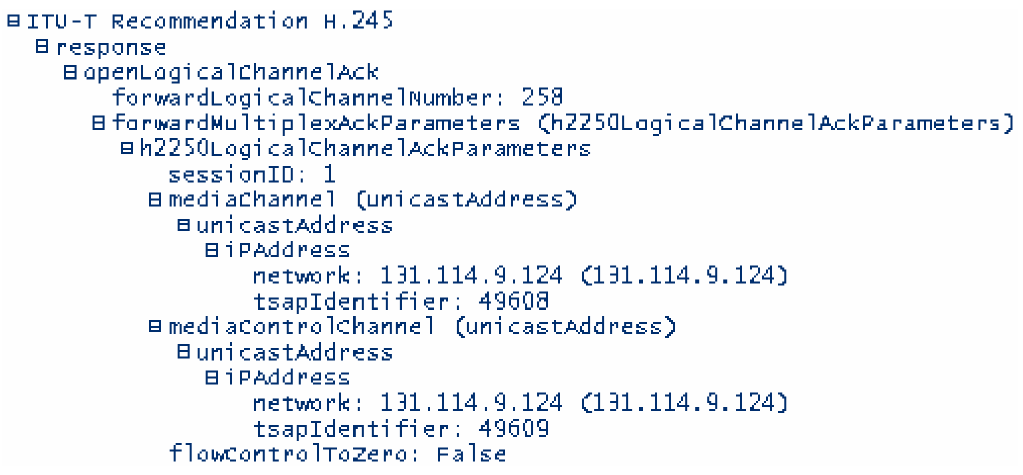
After **Terminal Capability Exchange** has been initiated, a master-slave determination procedure (consisting of either MASTERSLAVEDETERMINATION or MASTERSLAVEDETERMINATIONACK) has to be started as the first H.245 Conference Control procedure. Upon failure of initial capability exchange or master-slave determination procedures, a maximum of two retries are performed before the endpoint passes to the Call Termination phase. Normally, after successful completion of the requirements of this phase, the endpoints proceed directly to establishment of the audiovisual communication phase.

2.2.1.5.2.1 Encapsulation of H.245 messages within H.225.0 call signalling messages

Encapsulation of H.245 messages inside H.225.0 call signalling messages instead of establishing a separate H.245 channel is possible in order to save resources, synchronise call signalling and control and reduce call setup time. This process is called 'encapsulation' or 'tunnelling' 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 Signalling Channel. If tunnelling is used, any H.225.0 call signalling message may contain one or more H.245 messages. If there is no need to send an H.225.0 call signalling message when an H.245 message has to be transmitted, a FACILITY message is sent detailing (with appropriate fields inside) the reason for such a message.

2.2.1.5.3. Establishment of audiovisual communication

The establishment of audiovisual communication follows 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.



```
⊟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
```

Figure 2.7 OPENLOGICALCHANNELACK message content

2.2.1.5.4. Call services

When the call is active, the terminal may request additional call services. Among the services reported here are the Bandwidth Change Services and Supplementary Services. With Bandwidth Change 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 a bandwidth change, the endpoint waits for confirmation prior to actually changing the bit rate (confirmation usually comes from the gatekeeper). Asking for 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 the endpoint. If the request is accepted, a BANDWIDTH CHANGE CONFIRM (BCF) is sent back to the endpoint. With Supplementary Services, support is optional. The H.450 Series of Recommendations describes 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 number in the series.

Recommendation number	Recommendation Title
H.450.1	Supplementary Service 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 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

Figure 2.8 Supplementary services of the H.450-Series

2.2.1.5.5. Call termination

A call may be terminated either by both endpoints 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 Signalling Channel if this is still open.

An endpoint receiving an 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 that the whole 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 followed immediately by the endpoint up to the RELEASE COMPLETE message. 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 as the calling party, the gatekeeper just needs to look up its internal tables to resolve the target address. Complexity enters the picture if the destination address is registered with another gatekeeper. While Chapter 7 will cover this topic in more detail, the most basic mechanism that H.323 provides is explained here.

A gatekeeper may explicitly request the resolution of an address from other gatekeepers. On receipt of a request to call an address for which the gatekeeper has no registration, it can send out a location request (LRQ) to other gatekeepers (see Figure 2.9). The receiving gatekeeper, assuming it knows the address, will reply with the **Transport Service Access Point** (a combination of IP address and port number) of either the requested address or its own call signalling TSAP.

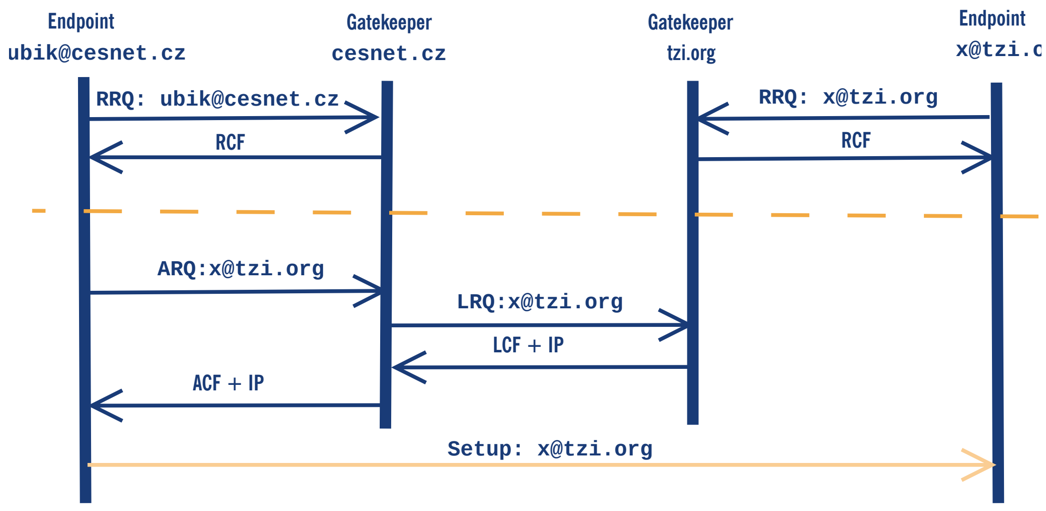


Figure 2.9 External address resolution using LRQs

A location request can be sent via unicast or multicast. If sent via multicast, only the gatekeeper that can resolve the address replies. If a gatekeeper receives a unicast LRQ, it either confirms or rejects the request.

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

{ 2.2.1.7 A sample call scenario

Figure 2.10 depicts an example of an inter-zone call setup using H.323 with one gatekeeper (A) using direct signalling while the other uses routed signalling. The calling party in zone A contacts its gatekeeper to ask for permission to call the called party in zone B (1). The gatekeeper of zone A confirms this request and provides the calling party with the address of zone B's gatekeeper (2).1 The calling party establishes a call signalling channel (and subsequently/in parallel the conference control channel) to the gatekeeper of zone B (3), who determines the location of the called party and forwards the request to the called party (4).

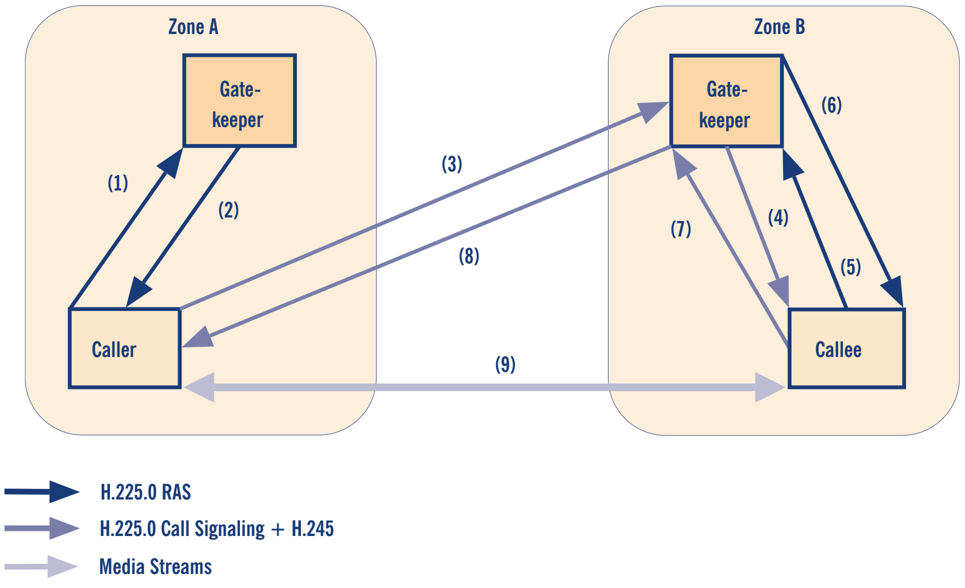


Figure 2.10 A sample H.323 call setup scenario

The called party explicitly confirms with its gatekeeper that it is allowed to accept the call (5, 6) and, if so, 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 calling party (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

It is well known from our daily interaction with PBXs that 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 in order 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 synchronisation of conference state among all participants from the outset, through the concept of a Multipoint Controller and an optional Multipoint Processor. While control is centralised in the MC, in theory, data exchange may be either via IP multicast, multi-unicast (i.e., peer-wise fan-out between endpoints without MP), or through an MP. (There seems to be practically no H.323 equipment supporting media multicast.) 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 a 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 defines a framework (derived from QSIG, the ECMA/ISO/ETSI standard for supplementary service signalling in PBXs) and a number of services (call transfer, call diversion, call hold, call park & pickup, call waiting, message waiting indication and call completion).

Further extensions to 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 standardisation.

2.2.1.9 H.235 Security

The H.235 recommendation defines elements of security for H.323:

- **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 authenticates the sender by comparing the received token with a self-generated token;
- **Message Integrity**
  Integrity is achieved by generating password-based checks on 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 employ DES, Triple DES or RC2. The use of SRTP is not supported yet (in H.235v2).

These mechanisms are grouped into the 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 simply 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 The purpose of SIP

SIP stands for Session Initiation Protocol. It is an application-layer control protocol that has been developed and designed within the IETF. 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, 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. The purpose of SIP is just to make the communication possible. The communication itself must be achieved by other means (and possibly another protocol). Two protocols that are most often used along with SIP are RTP and SDP. The 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 these 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 all of the devices can participate, including, 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 signalling protocol which means that all the logic is stored in end-devices (except routing of SIP messages). State is also stored only in end-devices. There is no single point of failure and networks designed this way scale well. The price 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 a regular PSTN (Public Switched Telephone Network) where all the state and logic is stored in the network and the end-devices (telephones) are very primitive. The 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 hardly be implemented in the traditional PSTNs.

SIP is based on HTTP protocol. The HTTP protocol inherited format of message headers from RFC822. HTTP and is probably the most successful and widely used protocol in the Internet. SIP tries to combine the best of both. In fact, HTTP can be classified as a signalling protocol too, because user-agents use the protocol to tell an HTTP server 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 the encoding of message headers from RFC822. 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 the form of sip:`username@domain`, or `sip:joe@company.com`. SIP URI consists of a username part and a domain name part, delimited by the @ (at) character. SIP URIs are similar to e-mail addresses and 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 element. Basic SIP elements are user agents, proxies, registrars and redirect servers. They are described briefly 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, for instance, to increase the speed of processing, but that depends on the particular implementation and configuration.

2.2.2.2.1. User agents

Internet endpoints that use SIP to find eachother and to negotiate a session's 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 and 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, user agents behave like a UAC or a UAS. For instance, a calling party's user agent behaves like UAC when it sends an INVITE request and receives responses to the request. A called party's user agent behaves like a UAS when it receives the INVITE and sends responses.

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

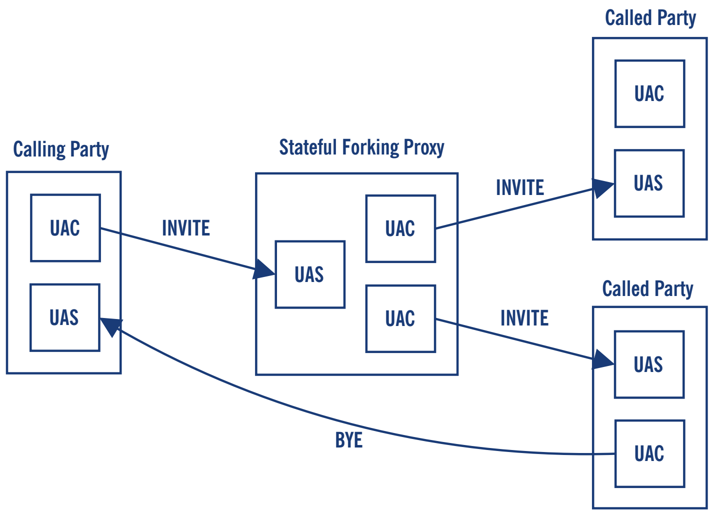


Figure 2.11 UAC and UAS

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 calling party, in fact, acts as a UAS. When forwarding the request statefully, the proxy creates two UACs, each of them responsible for one branch.

In the example, called party B picked up and later, when he wants to tear down the call, he 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

SIP allows the 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 a called party. The session invitation will usually traverse a set of proxies until it finds one which knows the actual location of the called party. Such a proxy will forward the session invitation directly to the called party and the called party 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 eachother. 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 re-transmissions of messages or 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 the called party 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 that 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 does not pick up, redirect the call to his cell phone. Stateless proxies cannot 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 and some sort of NAT traversal aid and all those features require a stateful proxy.

2.2.2.2.2.3 Proxy server usage

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

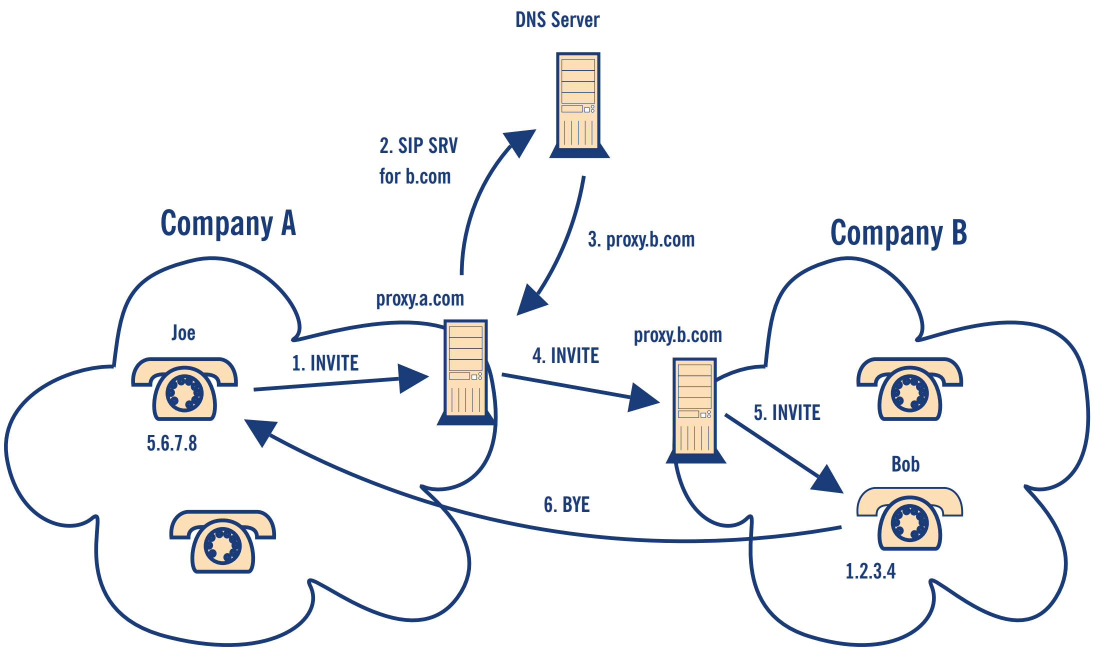


Figure 2.12 Session invitation

User Joe uses address `sip:bob@b.com` to call Bob. Joe's user agent does not 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

Its has been mentioned that the SIP Proxy at `proxy.b.com` knows current Bob's location but have not 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 a location database. The 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.

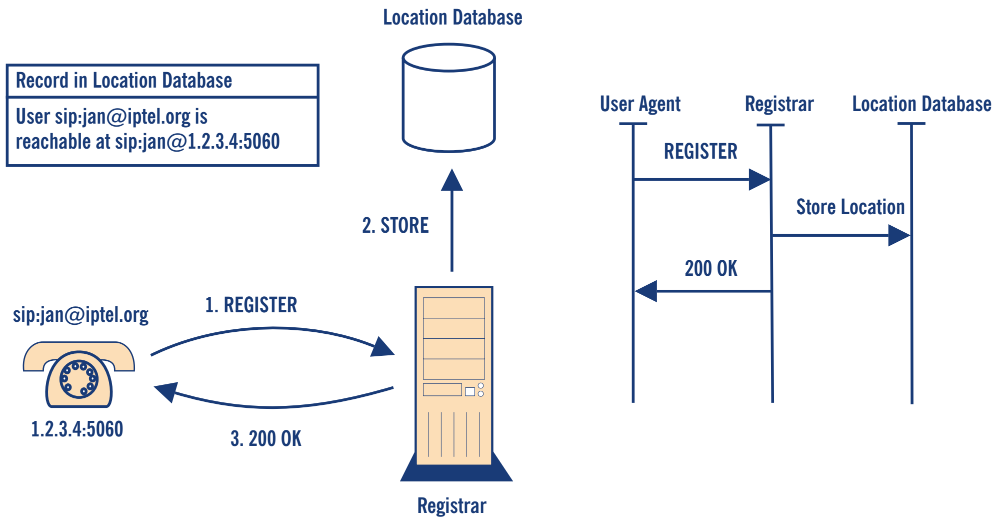


Figure 2.13 Overview of Registrar

Each registration has a limited life span. The **expires** header field or the **expires** parameter of the contact header field determines for how long the registration is 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 SIP 3xx redirection responses 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.

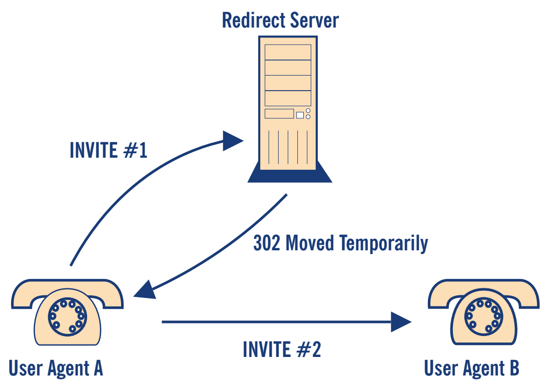


Figure 2.14 SIP Redirection

2.2.2.3 SIP messages

Communication using SIP (often called signalling) is comprised of a series of messages. Messages can be transported independently by the network. Usually they are each transported in a separate UDP datagram. Each message consists of a 'first line', a message header and a 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 the 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-Authorisation: 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 an 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 the 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.

The **From** and **To** header fields identify initiator (calling party) and recipient (called party) of the invitation (just like in SMTP where they identify sender and recipient of a message).

The **From** header field contains a tag parameter which serves as a dialogue identifier and will be described in Section 2.2.2.5.

The **Call-ID** header field is a dialogue identifier and its 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, sequence numbers must be present in the messages so that recipient can identify re-transmissions and out-of-order requests.

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

The **Message** header is delimited from message body by an empty line. The **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

An INVITE request has been described. The request is used to invite a called party to a session. Other important requests are:

- ACK
  This message acknowledges receipt of a final response to INVITE. Establishing of a session utilises 3-way hand-shaking due to asymmetric nature of the invitation. It may take a while before the called party accepts or declines the call so the called party's user agent periodically re-transmits a positive final response until it receives an ACK (which indicates that the calling party 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 a not yet fully-established session. It is used when the called party has not replied with a final response yet but the calling party wants to abort the call (typically when a called party does not respond for some time);
- REGISTER
  The purpose of REGISTER is to let the registrar know of current user's location. Information about the 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, many other request-types have been defined but their descriptions are 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 to 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"
```

Responses are very similar to the requests, except for the first line. The first line of response contains a 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 a response that tells to its recipient that the associated request was received but the result of the processing is not known yet. Provisional responses are sent only when the processing does not 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 called party'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 the 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 the 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 the **To** header field. Each response represents a distinct dialogue with an unambiguous dialogue identifier:

- 3xx responses are used to redirect a calling party. A redirection response gives information about the user's new location or an alternative service that the calling party might use to satisfy the call. Redirection responses are usually sent by proxy servers. When a proxy receives a request and does not want or can't process it for any reason, it will send a redirection response to the calling party and put another location into the response which the calling party might want to try. It can be the location of another proxy or the current location of the called party (from the location database created by a registrar). The calling party 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 could not 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 fulfil 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 does not want to participate in the session.

In addition to the response class, the first line also contains the 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 the method of corresponding request. In our example it was a 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 behaviour, 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 do 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 a 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 this, 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 its state gets updated from the message.

In the previous SIP RFC2543, 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 were a common source of problems.

In the new RFC3261, the way of calculating transaction identifiers was completely changed. Instead of the complicated hashing of important header fields, a SIP message now includes the identifier directly. The branch parameter of **Via** header fields directly contains the transaction identifier. This is a significant simplification, but there still exist old implementations that do not support the new way of calculating of the 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.

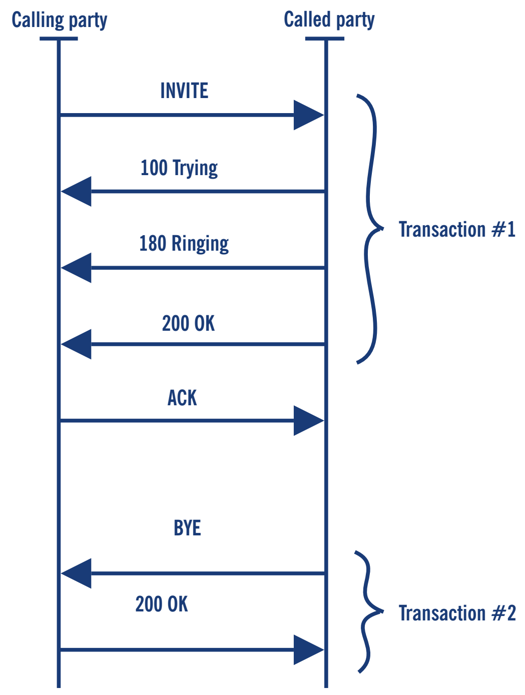


Figure 2.15 SIP transactions

{ 2.2.2.5 SIP Dialogues

It has been shown what transactions are, that one transaction includes INVITE and its responses and another transaction includes BYE and its responses when a session is being torn down. Those two transactions should be somehow **related-both** of them belong to the same dialogue. A dialogue represents a peer-to-peer SIP relationship between two user agents. A dialogue persists for some time and it is very important concept for user agents. Dialogues facilitate the proper sequencing and routing of messages between SIP endpoints.

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

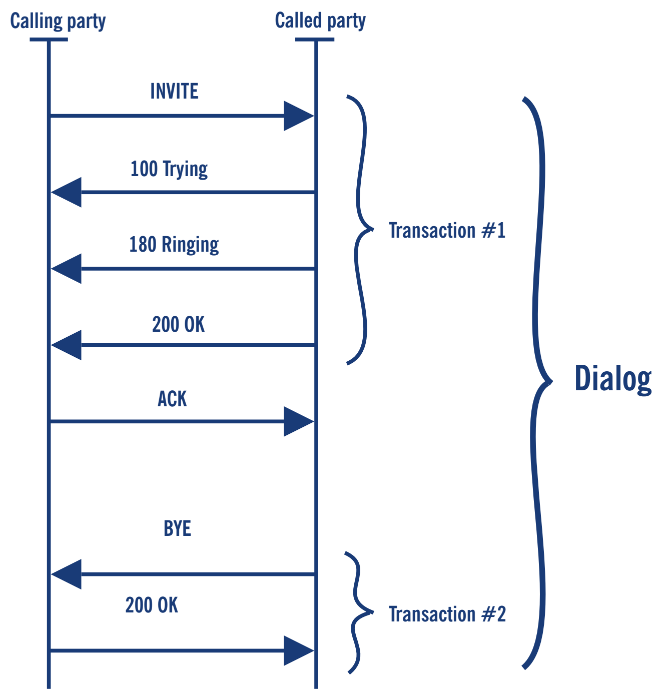


Figure 2.16 SIP dialogue

Some messages establish a dialogue and some do not. This is used to explicitly express the relationship of messages and also to send messages that are not related to other messages outside a dialogue. That is easier to implement because user agents do not have to maintain the dialogue state.

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

But, if a user agent sends a MESSAGE request, such a request does not establish any dialogue. Any subsequent messages (even MESSAGE) will be sent independently of the previous one.

2.2.2.5.1. Dialogues facilitate routing

Dialogues are also used to route the messages between user agents, as described briefly.

Suppose that user `sip:bob@a.com` wants to talk to user `sip:pete@b.com`. He knows the SIP address of the called party (`sip:pete@b.com`) but this address does not say anything about current location of the user, i.e., the calling party does not 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 the current location of the called party. This process is called routing. Once the request reaches the called party, the called party's user agent will create a response that will be sent back to the calling party. The called party's user agent will also put a contact header field into the response which will contain the current location of the user. The original request also contained a contact header field which means that both user agents know the current location of the peer.

Because the user agents know the 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 is exactly how dialogues facilitate routing.

Further messages within a dialogue 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 dialogue. They are used to route just the first request that establishes the dialogue. 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 dialogue (BYE) that bypasses the proxies.

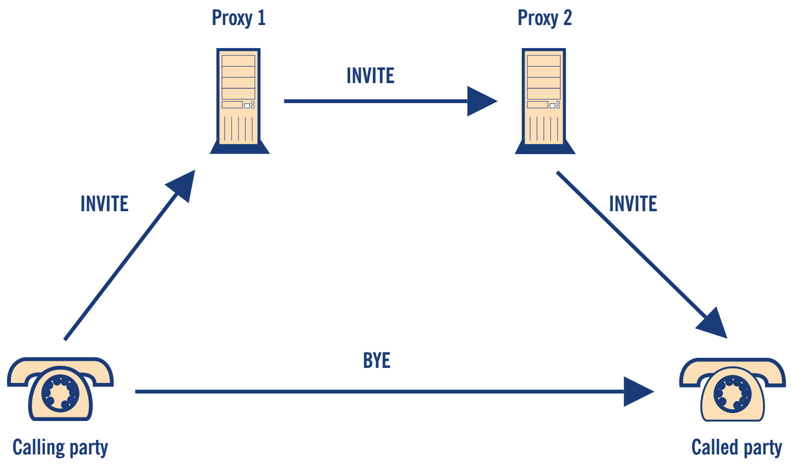


Figure 2.17 SIP trapezoid

2.2.2.5.2 Dialogue identifiers

Dialogue identifiers consist of three parts, **Call-Id**, From tag and **To** tag, but it is not that clear why dialogue identifiers are created exactly this way and who contributes which part.

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

A **From** tag is generated by the calling party and it uniquely identifies the dialogue in the calling party's user agent.

A **To** tag is generated by a called party and uniquely identifies it, just like the **From** tag is the dialogue in the called party's user agent.

This hierarchical dialogue identifier is necessary because a single call-invitation can create several dialogues and the calling party 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 the registrar if the registration was successful. Registrations are usually authorised so a 407 reply which can appear if the user did not provide valid credentials. Figure 2.18 shows an example of a registration.

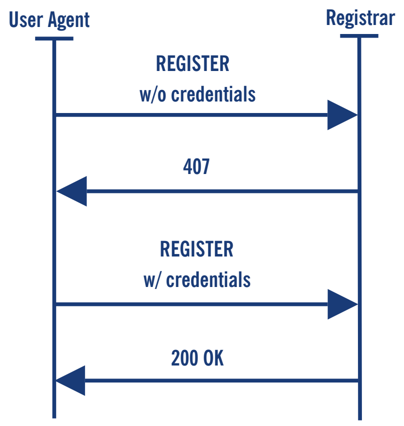


Figure 2.18 REGISTER message flow

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 the called party are sent back to the calling party. See teh **180 Ringing** response in the call flow. The response is generated when the called party's phone starts ringing.

A 200 OK is generated once the called party picks up the phone and it is re-transmitted by the called party's user agent until it receives an ACK from the calling party. The session is established at this point.

2.2.2.6.3 Session termination

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

A 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.

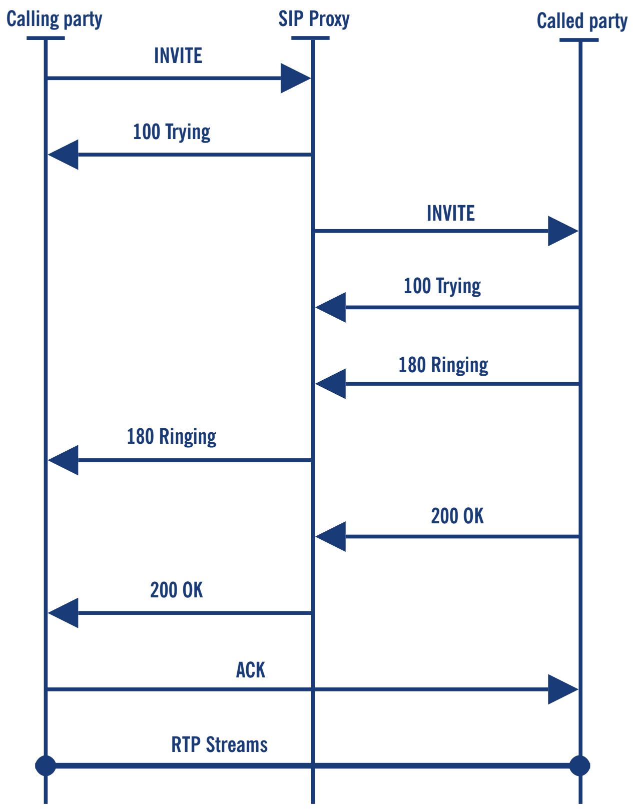


Figure 2.19 INVITE message flow

2.2.2.6.4 Record routing

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

There are certain situations in which a SIP Proxy needs 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.

The 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 a **Record-Route** header field into SIP messages which contain address of the proxy. Messages sent within a dialogue 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.

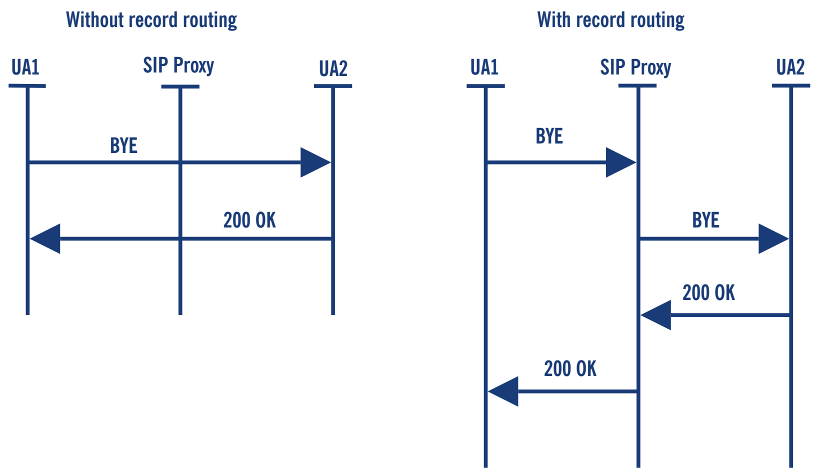


Figure 2.20 BYE message flow (with and without record routing)

The lefthand message flow of Figure 2.20 shows how a BYE (request within dialogue established by INVITE) is sent directly to the other user agent when there is no **Record-Route** header field in the message. The righthand message flow shows 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 events. Such evens can include, SIP Proxy statistics changes to, presence information, session changes and so on.

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

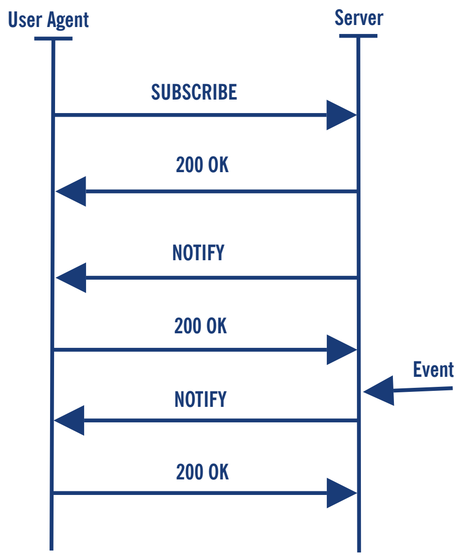


Figure 2.21 Event subscription and notification

A user agent interested in an event notification sends a SUBSCRIBE message to a SIP server. The SUBSCRIBE message establishes a dialogue and is immediately replied to by the server using a 200 OK response. At this point, the dialogue 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 dialogue 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 a limited life span and therefore must be periodically refreshed.

2.2.2.6.6 Instant messages

Instant messages are sent using a MESSAGE request. MESSAGE requests do not establish a dialogue 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.

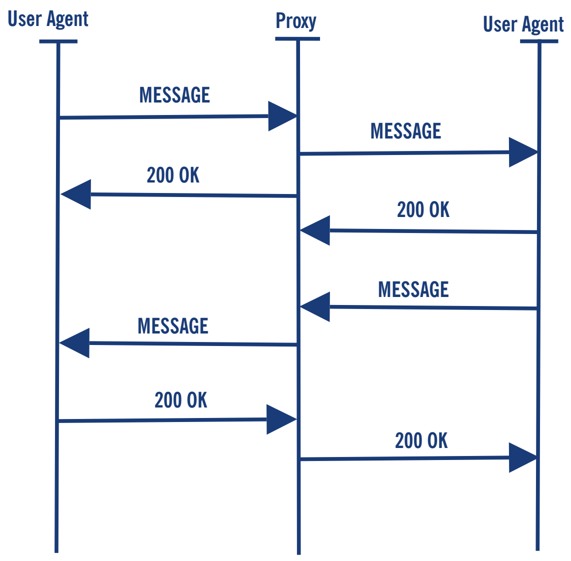


Figure 2.22 Instant Messages

{ 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 equipment (PBXs, telephones). While the internal network today is based upon digital communication, links to customers may be either analogue (PSTN) or digital (ISDN). The links to customers are shared between call signalling (for dialling, invocation of supplementary services, etc.) and carriage of voice/data. In the backbone, dedicated (virtual) links interconnecting switches are reserved for call signalling (de-facto creation of a dedicated network of its own) whereas voice/data traffic is carried on separate links. The Signalling System No. 7 (SS7) or variants of it are used as the call signalling 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 signalling 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.

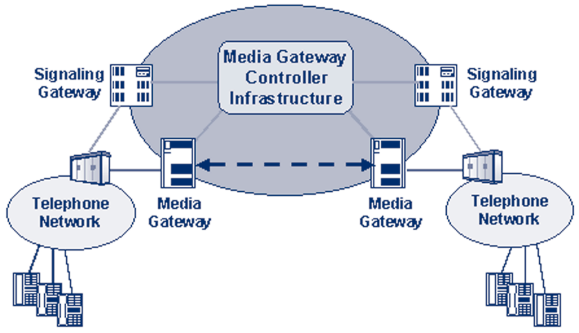


Figure 2.23 Application scenario for Media Gateway Control Protocols

A similar construction is now considered by a number of telecom companies 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. Instead, 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. Two types of gateways are used at the edges of the IP network to connect to the conventional telephone network: signalling gateways to convert SS7 signalling into IP-based call control (which may make use of H.323 or SIP or simply provide a transport to carry SS7 signalling in IP packets [SIGTRAN]) and media gateways that perform voice transcoding. Some central entity (or more probably, a number of co-operating entities) forms the intelligent core of the backbone, the Media Gateway Controller(s). They interpret call signalling and decide how to route calls and they 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 standardised, 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.

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 signalling 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 specific supplementary services in the Voice over IP world, in order to offer customers all the features they already know from their classic PBX. The enterprise solutions usually feature such proprietary protocols at the cure and provide minimalist support for standardised protocols (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 is usually difficult to provide because most protocols are not publicly available.

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

RTP and RTCP are the transport protocols used for IP Telephony media streams. Both of them were defined in RFC1889: 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 synchronisation 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 means 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 video data;
- advertising the number of session participants which is used to adjust the RTP data transmission rate;
- carrying session control information used to identify the session participants.

The next two subsections describe the RTP and RTCP header and the different types of packets that the two protocols use.

2.2.5.1 RTP header

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            contributing source (CSRC) identifiers             |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.24 RTP header

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 identifier number which follows the header;
- marker (M - 1 bit) is the 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, is 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 synchronisation and jitter calculation;
- SSRC ID (32 bits) identifies the synchronisation source, chosen randomly within a RTP session;
- CSRC ID list (from 0 to 15*32 bits) is an 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, and this part is then followed by structural elements of variable length. More than one RTCP packet may be linked together to build a COMPOUND PACKET. Moreover, in order to maximise the statistics resolution, the SR and the RR packet-types are to be sent more often than the other packet-types.

3 IP Telephony Scenarios >

> 3.1 Introduction

Using the background technological terminology defined in the previous chapter, this chapter describes the most interesting scenarios in building an IP Telephony infrastructure. The following scenarios 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 reader with both an overview of each scenario and motivation for deploying them. Each scenario is analysed from the user-needs point of view and is described architecturally, giving an example of an implementation.

> 3.2 Scenario 1: Long-distance least-cost routing

This scenario is likely to be adopted by enterprises with high-cost call volumes. Traditionally, separate links have been used for transferring voice and data between two sites (see Figure 3.1), thus making it simple to achieve a reduction in costs by establishing accounts with a lower-cost long-distance carrier. Voice over IP offers an alternative solution for this kind of problem; existing enterprise data networks (using the 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 a lower-cost, long-distance carrier and Voice over IP voice-data integration is seen as the most cost-effective solution in this area. This requires the routing of calls to the lowest-cost network, depending on the time of day and destination, and it is referred to as Least-Cost Routing (LCR). In order to achieve greater savings, it routes calls to destinations by re-dialling them through the lowest cost alternative carrier/terminator available. A basic scenario's architecture (depicted in Figure 3.3) is able to handle all calls originating from the enterprise network. Elements needed to deploy this scenario are: terminals (both IP and PSTN) and the necessary gateways to route the call from the IP network to the ISDN/PSTN/GSM and vice versa. Other elements like MCUs or servers may be present, but are not required.

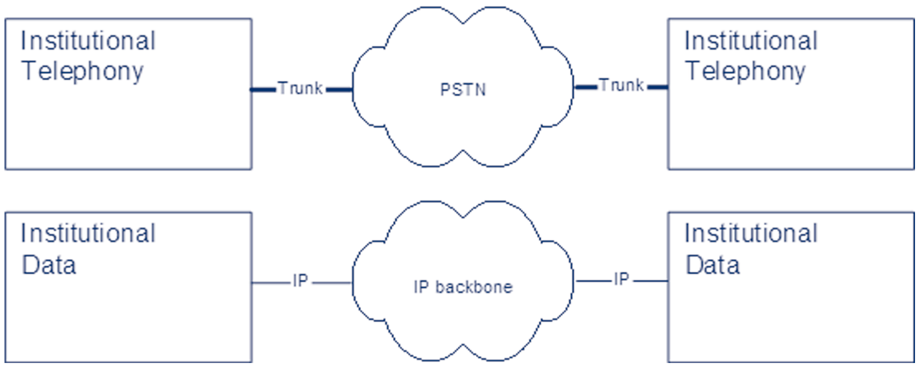


Figure 3.1 Traditional separation of data and telephony between locations

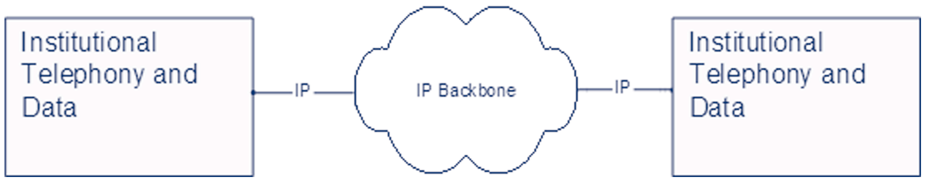

Institutional Telephony and Data
IP
IP Backbone
IP
Institutional Telephony and Data

Figure 3.2 Integration of data and telephony between locations

A hybrid solution, including both traditional processing of calls over PSTN/ISDN and additional IP Telephony parts, results in this detailed architecture.

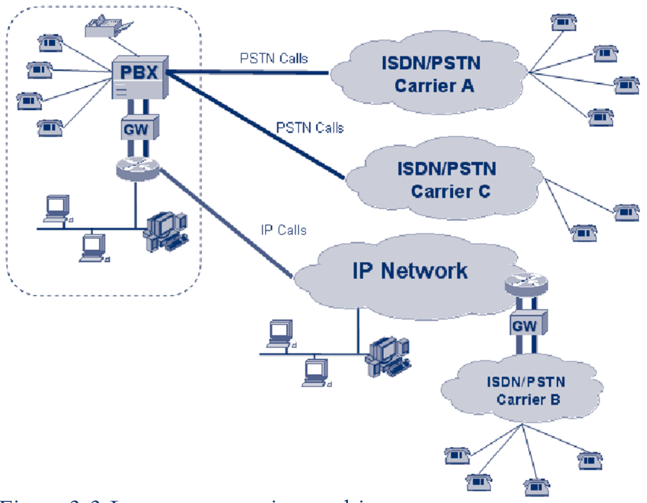


Figure3.3 Least-cost routing architecture

The features that a least-cost routing architecture may provide can be summarised as:
- call routing by time of day and day of the week, allowing selection of the best rates for specific time periods;
- call routing by destination, allowing selection of the best rates depending on the destination of the call;
- number modification, allowing dial-string manipulation of the original number dialled, to facilitate prefix-based routing;
- class of service management, allowing management of 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 an implementation

A company with head-quarters offices and multiple-branch office sites 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 geographic areas, the competitive telephone market may be used to reduce communication costs. A first solution would require the maintenance of an up-to-date table, based on the savings depending on the time of day and destination. The problems arising from this solution in the maintenance and distribution of this table to the employees are evident. Moreover, it is certain that not every employee will remember to dial the extra digits for each appropriate prefix, both because it is time-consuming and, as a result of negligence.

Therefore, an engineering process is needed to keep the costs low. Least-cost routing is the solution to these kinds of problems, because 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 reducing the employee's effort in making calls.

In order to put such solution in place, the company needs to deploy a set of gateways in the locations where branch offices are located, to take advantage of the integration of data and telephony links between locations as depicted in Figure 3.2. This can result in savings from both calls located in the area of the branch offices, as well as office-to-office calls, taking advantage of the data network connecting the company's sites. Note that in this case, a distributed routing table has to be implemented, in order to facilitate control by the system administrator, who may wish to update it anytime changes in long-distance rates occur.

> 3.3. Scenario 2: Alternatives to legacy PBX systems

Traditionally, institutions and companies are equipped with a PBX on each one of its sites. 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).

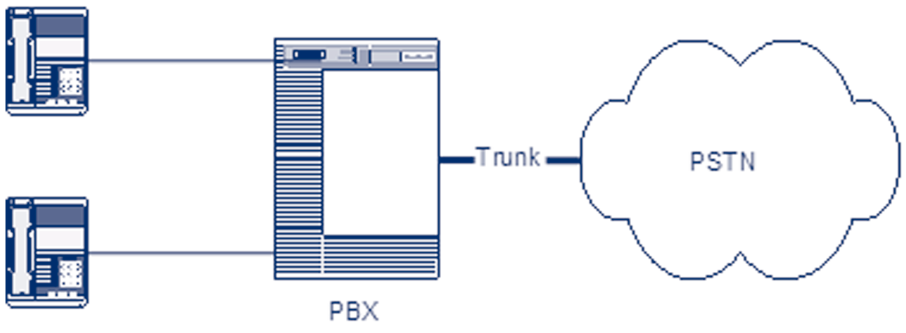


Figure 3.4 Legacy PBX which trunks to the PSTN

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

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

Then, a more common scenario will be described, where IP Telephony is introduced into the existing telephony infrastructure. The legacy PBX is still functional in this scenario and voice calls can be carried not only over regular PSTN trunks, but also over IP backbones.

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

> 3.3.1 Scenario 2a: IP Phones without a PBX system

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

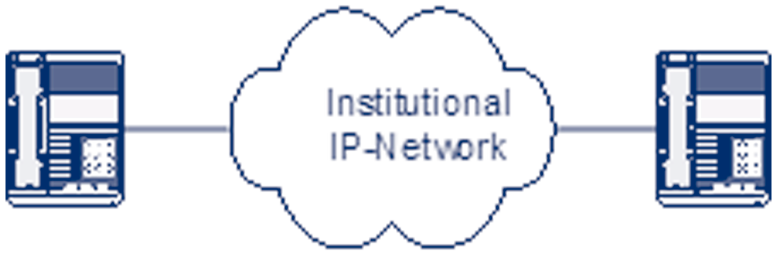


Figure 3.5 IP Phone to IP Phone without PBX

For mission-critical cases, such as a commercial company or an 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, because no VoIP-to-PSTN gateway is available. Also, common features like central address books, call forwarding services, etc. 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 a voicemail service, etc. This setup is therefore only recommended for testing purposes.

Call setup is very simple, when using either H.323, or SIP, or any variations of these protocols. Since the calling party directly enters the IP address of the destination, call initiation signalling is sent directly to that IP address. If the terminal is functional, it will process the signalling 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 signalling that terminates the call is exchanged.

> 3.3.2. Scenario 2b: Integration of VoIP with legacy PBX systems

This scenario allows the co-existence and intercommunication of the institutional conventional telephony network (conventional phones connected to PBX) and the 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.

For example, in order to provide for smooth transition, it might be worthwhile to buy a gateway with two ISDN PRI interfaces (or just with one interface and borrow the second interface for the transition period). One interface is connected to the PSTN and the second one to the PBX. During the transition period, a gateway also performs call routing between the PSTN and the old PBX and vice versa, providing a smooth transition.

This chapter gives 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.

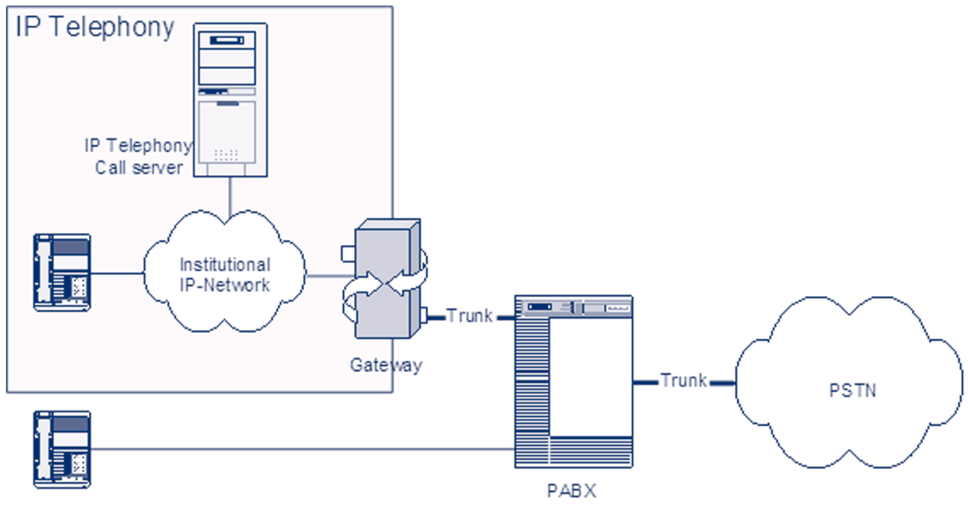


Figure 3.6 Integration of IP Telephony with a legacy PBX system

The choice of a particular interface between a PBX and a VoGW depends on the required functionality, availability of interconnection ports on both sides and also on cost constraints. Interfaces can be divided into analogue and digital. The former include a 2-wire U-interface with a subscriber loop and various types of E&M interfaces. The latter include an E1/CAS trunk with MFC-R2 signalling and ISDN with DSS1 or QSIG signalling. Giving technical details about the trunks and interfaces mentioned above is outside the scope of this chapter Please refer to Chapter 4 for further details. On the other hand, technical people who want to understand this kind of scenario may benefit from a discussion of the advantages and shortcomings of individual interfaces, which are summarised in the following list:

- **Subscriber loop** - suitable when conventional phones should be connected directly to VoGW (Voice GateWay) via an 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 towards the PBX, an extension number can be dialled only as DTMF (Dual-Tone Multi-Frequency) suffix, after a call is established and is already accounted for. This type of interface is usually a low-cost solution;
- **E&M interfaces** - E&M commonly stands for both Ear and Mouth or recEive and transMit. It allows extension dialling before the conversation begins. It requires a special interface card for the PBX, but if the PBX is already equipped with this card, this can also be a low-cost solution;
- **E1/CAS trunk with MFC-R2 signalling** - CAS (Channel Associated Signalling) exists in many variants that operate over analogue and digital interfaces. 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 largely

replaced by ISDN interfaces. It requires special interface cards on both sides of the interconnection, and it is a rather expensive solution;

- **ISDN with DSS1 signalling** – In addition to calling party identification, supplementary services are available such as Call Waiting, Do Not Disturb, etc. It can be used with a BRI interface (Basic Rate Interface, up to 2 simultaneous calls) or a 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 signalling** – QSIG signalling supports more supplementary services, such as completion, Path replacements, etc. However, QSIG uses proprietary features of the 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 of legacy PBX systems

It is only in greenfield situations, when building a telephony service from scratch, or when an existing PBX is fully depreciated, that IP Telephony can be considered as a complete replacement alternative to a traditional PBX.

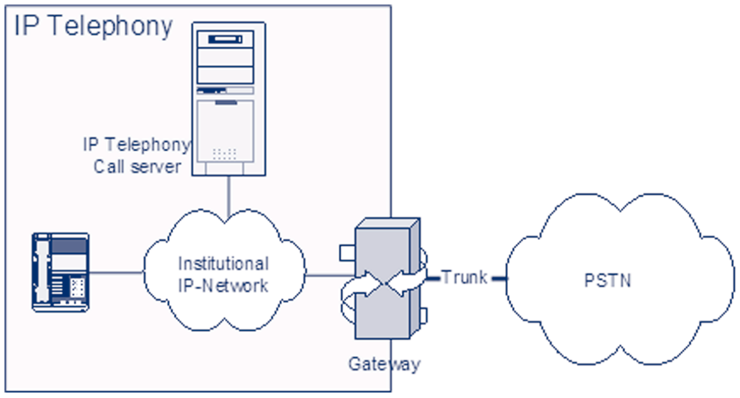


Figure 3.7 IP Telephony fully-replacing PBX

The design of an IPBX system involves a couple of choices.

> 3.3.3.1 Intelligent vs. simple terminals

IPBXs can support terminals in two ways: either through analogue ports that support analogue terminals, or through IP only. The latter implies that the terminals are intelligent devices, including an implementation of signalling functions. Since intelligence in IP phones is built-in anyway, these terminals are often equipped with interactive screens and other sophisticated functions. As a result, the equipment is expensive and requires the provision of power, either externally or by Power-over-Ethernet. A feature that most of these advanced terminals support is pass-through of Ethernet packets, so that a single wall outlet can be used for both IP Telephony and computer data.

> 3.3.3.2 Signalling

Though the choice among H.323, SIP and proprietary protocols seems a purely technical one, it has implications on the interoperability with future expansions, inter-department trunking and the deployment of new advanced features, like messaging, etc. It is wise to require that a supplier complies with at least one of the open standards.

> 3.3.3.3. Inter-department trunking

The choice of a complete, IP-based institutional voice architecture does not automatically lead to a specific solution for connecting geographically-separated locations. The cookbook examines the options for this case in the 'Least-Cost Routing' Section.

The inter-departmental architecture also involves a choice of whether to break out local calls at local PSTN trunks, or to centralise 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 centralised break out, as well as the volume of calls that have a local public destination as compared to long-distance publicly- and privately- destined calls.

> 3.3.3.4 Legacy functionality

Traditional PBXs have the advantage that long-recognised needs have been incorporated in their functionality through decades of development. These important features need to be implemented by the IP Telephony architecture as well, if it is to become a competitive solution. The most elementary of these are:

- emergency call handling to public emergency numbers (911, 112 etc);
- public dialling plan routing (regular numbers, blocking/routing of premium numbers etc);
- integration with public wireless telephony;
- voice/data integration and call distribution for call centre/help desk department;
- support for beeper systems;
- support for private wireless telephony;
- support for elevator phones.

> 3.3.3. Wireless VoIP

With at least 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 with Wi-Fi networks in place, will 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 as intelligent as their fixed IP equivalents, but are different on the Ethernet level. The usual issues in wireless data communications are battery autonomy, portability, coverage, etc. Current developments show that manufacturers of public network mobile phones like GSM are planning to include Wireless VoIP into their terminals. This would enable seamless roaming from public mobile telephony networks to the campus wireless environment, potentially reducing costs when calling locally on campus.

> 3.3.3.6. Issues

Since the field of IPBXs is rapidly emerging, many features that are known in the traditional PBXs are quickly adopted. Additionally, new issues arise as data networks develop. An example is the introduction of network access control by IEEE 802.1X. This standard forces equipment to first authenticate at a RADIUS server before accessing the network. All equipment on 802.1X-enabled network ports should be 802.1X-enabled as well. With the adoption of 802.1X, the vendors are announcing terminals that support this standard as well.

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

> 3.4. Scenario 3: Integration of VoIP and videoconferencing

When referring to VoIP and IP Telephony, the main focus is on voice services, which may be misleading regarding the support of video. IP Telephony standards have the capability to signal and are able to initiate multimedia communication (see Chapter 2). This scenario details how voice over IP technologies/standards and videoconferencing solutions may be seamlessly integrated. The goal is to provide the users with a global architecture derived from IP Telephony standards, giving videoconferencing systems the chance of becoming widely used and adopted. Videoconferencing systems have the purpose of facilitating meetings of remote participants, and to support the illusion that they are all sharing the same space and communicating as if they were in the same room.

Perfect videoconferencing sessions are achieved when the technology is no longer noticeable. Even though the perceived quality of video and audio plays the most important role, there are a number of other factors influencing the perception of successful videoconferencing:

- **accessibility of the system** – the system should be broadly accessible, giving users the easiest way of communicating without worrying about how to join a conference or how to find 'reachability' information about the party to call;
- **value-added services, such as data/application-sharing and voice mail,** are only two examples of value-added services that are not feasible with classic telephony systems, yet 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, it is necessary to examine which are the possible applications related to the videoconferencing scenario. The 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 the basic functionality, with enhanced options.

Here we cite 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 deliver high-quality video images to remote medical specialists. Specialised 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 invited to a classroom and private lessons to groups of students located in different locations are some of the educational processes requiring the use of videoconferencing tools;
- **Customer Services** – Videoconferencing-based customer services enable call centre operators to be more effective when interacting with customers.
- **Justice services** – Many legal systems have introduced the use of videoconferencing to enable police officers to attend legal proceedings. This optimises the time police need to spend in courthouses;
- **Virtual/Remote Laboratories** – Remote laboratories allow researchers to share advanced appliances using existing network infrastructure. In telemedicine, specialised devices not available with the standard videoconferencing systems may be required to enable high-quality video contents to be transmitted. Moreover, special considerations apply when such interaction modes are considered.

While simple desktop conferencing equipment may be enough for basic meetings, a successful integration scenario would require, on the application side, application-specific devices to deliver the content to the end-user. The technical side would require servers to build a global architecture accessible by all group users, gateways to provide interoperability with different access technologies and different IP Telephony protocols, conference bridges and multipoint conferencing units to provide capabilities for multipoint conferencing.

> 3.4.1 Integrating voice and videoconferencing over IP - an example

In order to give the reader an understanding of a complex scenario, such as the integration of voice and videoconferencing over IP, an example, actually implemented at the SURFnet offices in The Netherlands is described. An integrated voice and video environment is a setup based on H.323 components (endpoints, MCU and gateway), a Cisco CallManager (using the Skinny protocol), and a PBX. Therefore this is also an example of scenario 2b (Integration with legacy PBX systems). Figure 3.8. gives an example of integrated voice and video over IP architecture at the SURFnet offices.

The goal of the integration of voice and videoconferencing over IP was to make it possible to refer directly to the user without knowing his location and what terminal he is actually using. When someone contacts the user by any means (PSTN or H.323 of H.320), the call should be completed by reaching any device the user may have operational. 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;

- an H.323 Gateway, in this case a RADVISION L2W (LAN to WAN H.323) gateway, on the one side connected to the PBX (by 2 ISDN lines) and on the other side to the local IP network;
- an H.323 Gatekeeper, in this case the build-in gatekeeper of 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 call manager, in this case a Cisco CallManager, being the gatekeeper to perform 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 on the CallManager;
* H.323 videoconferencing stations, in this case VCON Escort Pro 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.

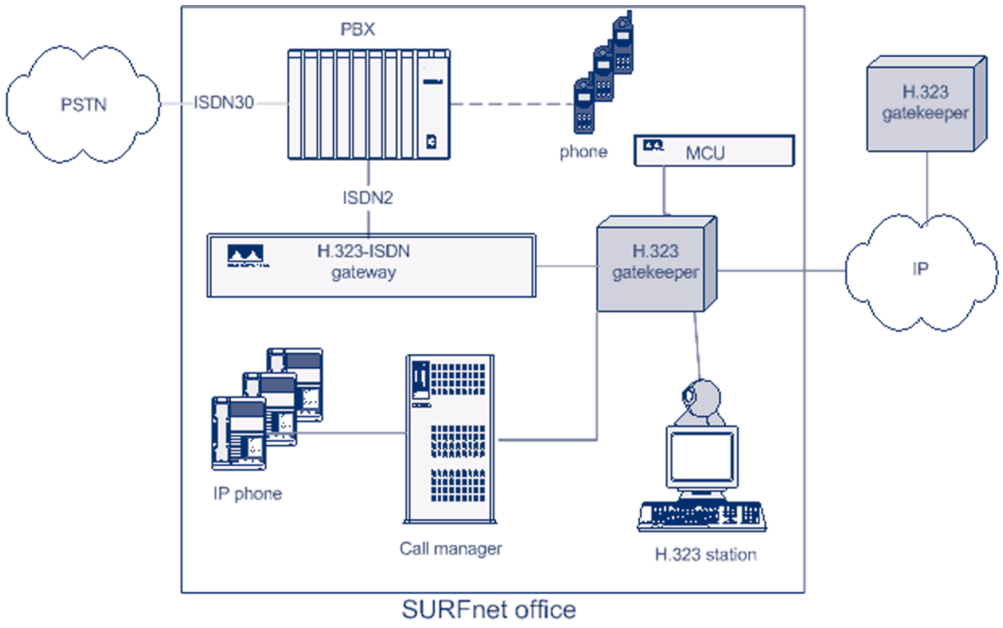


Figure 3.8. Integrated voice and video over IP architecture at the SURFnet offices.

To allow multipoint calls, an MCU (RADVISION MCU323) has to be added and the conference feature on the PBX has to be enabled. These are not necessary functionalities, but can enhance the communication experience.

The means by which integration was established was the Dial Plan that guaranteed unique number-addressing for all devices. The Global Dialling System (GDS) was adopted, and the full E.164 addressing, number of videoconferencing and IP Telephony endpoint numbering allow all terminals to be used like regular phones (see the Section on 'Dial Plans' and also `http://www.wvn.ac.uk/support/h323address.htm`). Therefore, it is guaranteed that for terminals called/dialled from the PSTN, the call 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 dialled without the need for rewriting the dial string. 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, and his international phone number is 0031302305367. His IP phone number is 030-2305167. For demonstration purposes, he has also registered his H.323 station as 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 control of and 67 is the local office number assigned to ‘A’ (Note that the assignment is 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 a service at the office gatekeeper (with the prefix 5) and connected to the PBX. In the PBX, it 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 even as a Web-based service, so users can maintain their own preferences. At the PBX, the group number 501 for group that A belongs to is also defined (the group number for making all telephones in a group ring). At the gatekeeper, the number 500 is configured as a backup number that will be called if the H.323 call is not answered. The IP phones are registered with their 1xx number (in this case 167) at the CallManager, which itself is registered as a gateway serving all these numbers (167, 109, 1xx etc) at the gatekeeper.

The following situations are not a complete list of possibilities, but a several representative ones:

- A user on the PSTN calls A at SURFnet, who 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 recognises 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 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 set up. A answers the incoming call on his videoconferencing station, only receiving audio, since there is no video attached to the signal from the PSTN user. If A does not answer on his H.323 station, then the gatekeeper sees this and dials the backup number 501. The gatekeeper recognises 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 makes all phones in the group ring. A or one of his colleagues can then answer the call by picking up any of the phones in the group.

- 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 CallManager recognises this as an international call and forwards it to the local gatekeeper. The gatekeeper sees that it is an international call and forwards it to the ViDeNet root gatekeeper. Here the prefix 31 is recognised and the call is forwarded to the Dutch national gatekeeper. There the prefix 3023053 is recognised and the call is forwarded to the SURFnet office gatekeeper. Here the number 67 is recognised. 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.

- A videoconferencing station dials A's IP phone. Someone using an 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 an H.323-Skinny gateway and forwards the call to the IP phone.

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

Local dialling between the systems is also supported: A can call 97 from his phone to reach his H.323 station, or 167 to reach his IP phone. The other way around (from IP phone 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.

If MCU was involved, people using either a PSTN device, or an H.323 IP phone, or a videoconferencing station would dial the routable E.164 number of the multipoint conference that is registered at the office gatekeeper, as if it was an 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 the devices used even further.

Note that the example above relies on a numeric dialling plan. Alphanumeric dialling and routing is implemented differently (see Section 2.1.5 on 'Addressing').

4 Setting Up Basic Services ~

In Chapter 2, the working of H.323 and SIP protocols was explained. This chapter explains how to set up an infrastructure including H.323 and/or SIP components and gives real-world examples of configuration for popular equipment. The focus will be on setting up basic services, meaning the equipment that is 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, among others, is to resolve a dialled address to an IP target. However, when we talk about real-world set-ups, this server infrastructure tends to be more complicated. The 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 signalling protocol.

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

A server that supports more than one signalling 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.

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 an H.323 Gatekeeper, then call routing inside the domain becomes an issue. A signalling gateway is required to enable an H.323 endpoint to call a SIP endpoint and vice versa (see Figure 4.2).

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

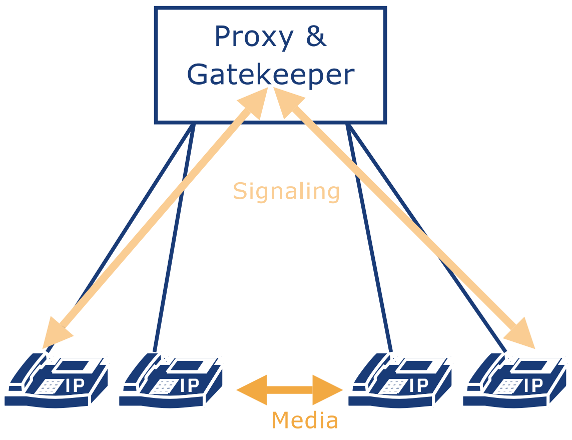


Figure 4.1 SIP/H.323 zone using a multi-protocol server

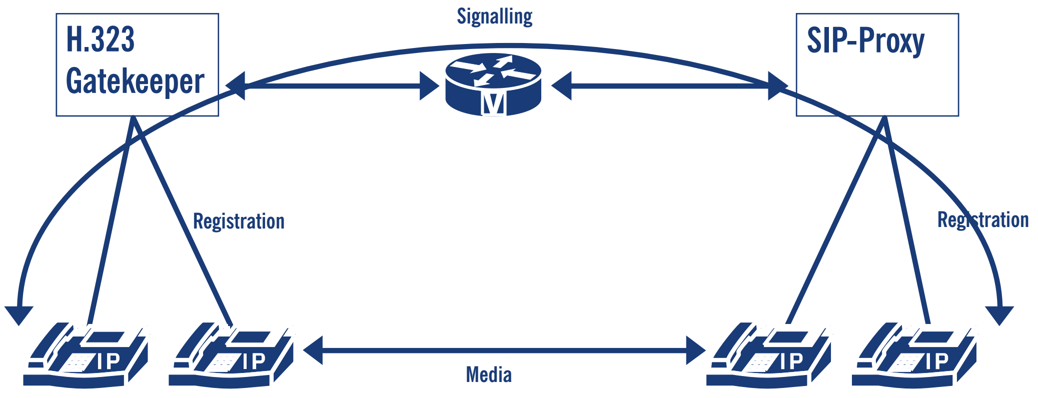


Figure 4.2 SIP/H.323 zone using a signalling gateway

The gateway is used to translate signalling 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 (see Section 2.2.1.5.1).

An architectural problem similar to the last one is the use of servers that feature a proprietary IP Telephony protocol and provide a SIP or an H.323 interface that is limited to basic call functionality without any supplementary services or security features. An example of 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.

Besides technical aspects, organisational aspects of PBX-VoIP integration call for careful planning and analysis. The question is how to integrate legacy and IP Telephony equipment into the dial plan of an organisation. Is it necessary that the phone number reflects whether the participant is a VoIP user or a PBX user.

Generally, there are three possibilities to explore, as explained below. There are more details on setting up an IP Telephony gateway in Section 5.1.

4.1.1.1.1. Routing based on a number prefix

This option requires that, in the dial plan, there is a numberblock available for IP Telephony use. This is easy to implement on legacy PBXs and IP Telephony servers but the result is new phone numbers for every user that switches from legacy to IP Telephony.

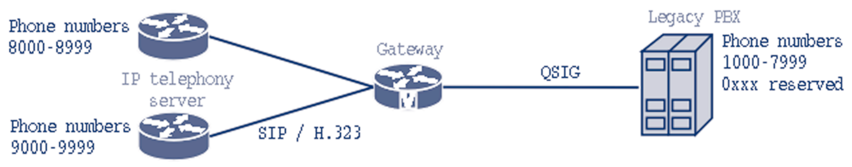


Figure 4.3 Routing based on number prefix

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 only 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 a legacy PBX towards IP Telephony, provision of seamless migration is often required 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.

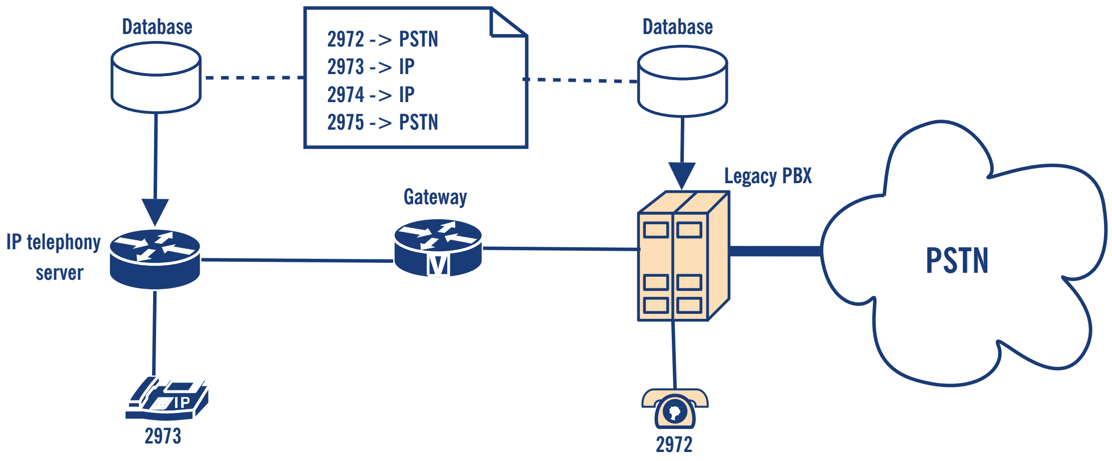


Figure 4.4 Per-number routing

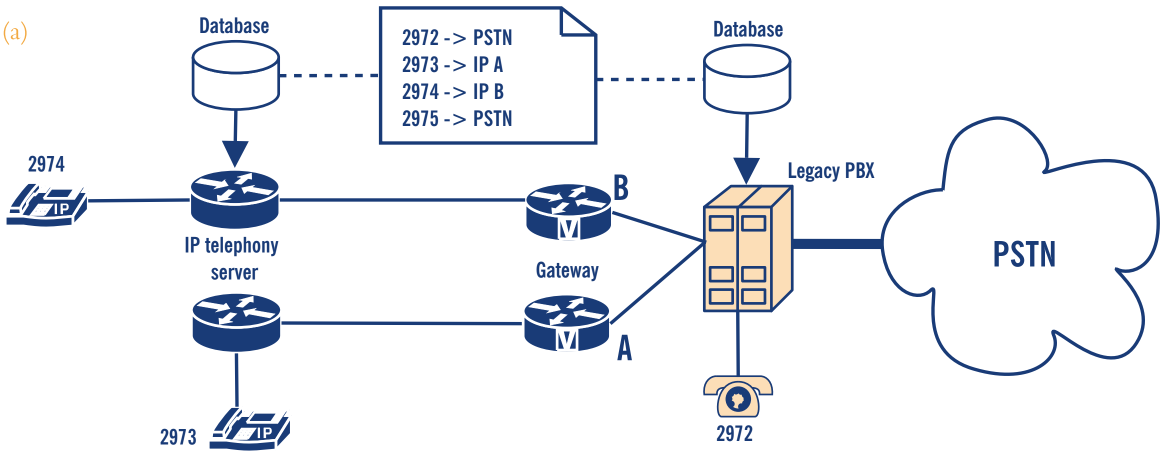
This is quite easy to achieve by setting up a database that stores the information for telephone numbers that belong to the PBX or the IP world (see Figure 4.4). The IP Telephony server and the PBX access the same data to decide where to route a call. Calls to external targets are routed to the PBX and 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 signalling protocol used. In this case, there needs to be a database that contains not only the information about which number shall be reached on the PBX and which in the IP world, but also the information about which IP server (and signalling protocol) to use.

(a)



(b)

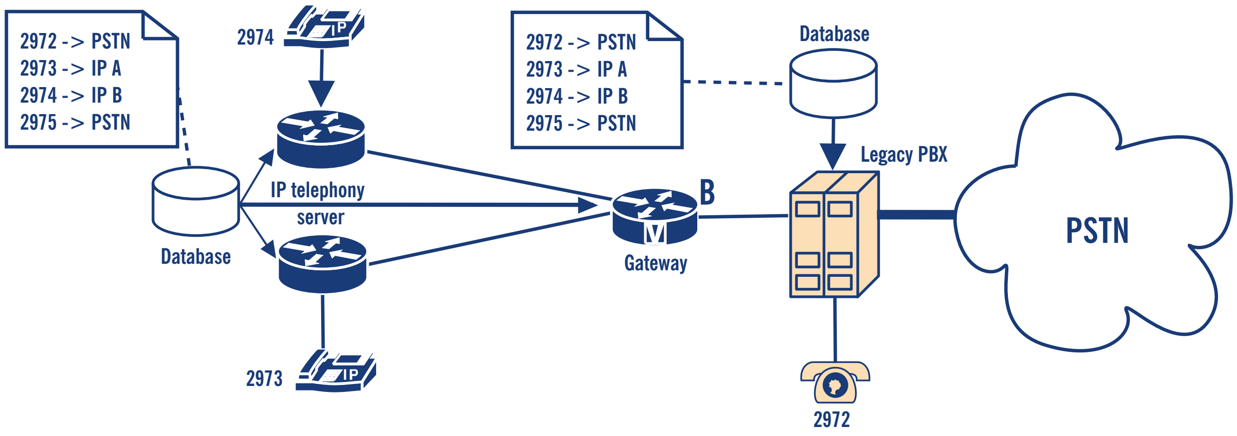


Figure 4.5 Per-number routing with a) two gateways or b) one gateway

The first scenario uses two gateways and thus allows the PBX to decide where to route a call to. 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 signalling 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, but merely needs to know the fact that the call has an IP target. 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., an 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 synchronisation with a high risk of inconsistency.

~ 4.1.1.2 Trunking

The bigger an institution is, the more complex its organisational infrastructure. This may be due to the need to support multiple locations (sites) or because certain organisational units need to administer 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 dialling space.

4.1.1.2.1 Prefix-based trunking

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 dial plan already provides these prefixes, e.g., in the form of area codes plus subscriber prefixes.

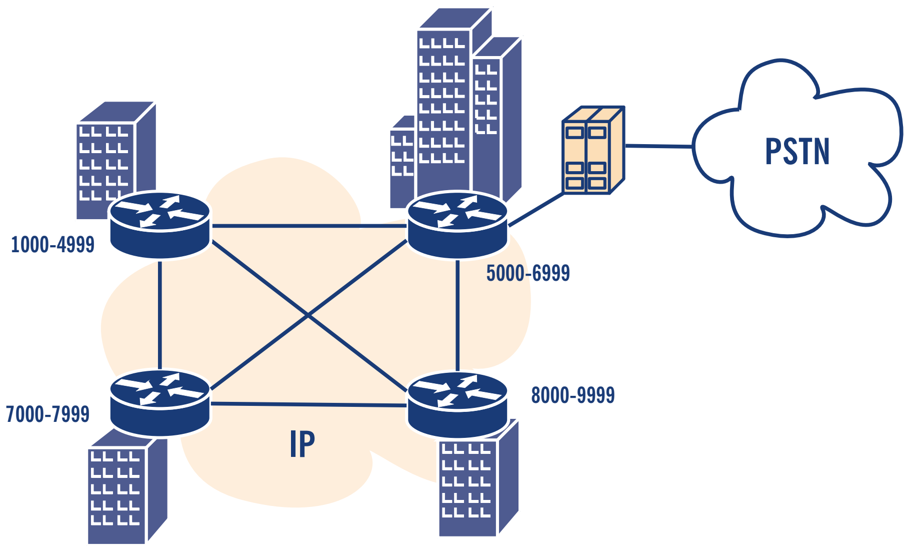


Figure 4.6 Prefix-based trunking

This is the classic branch office scenario that is based upon the assumption that both networks have different locations, and that dialling 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 dialling 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 dial plan 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 dialled number?

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.

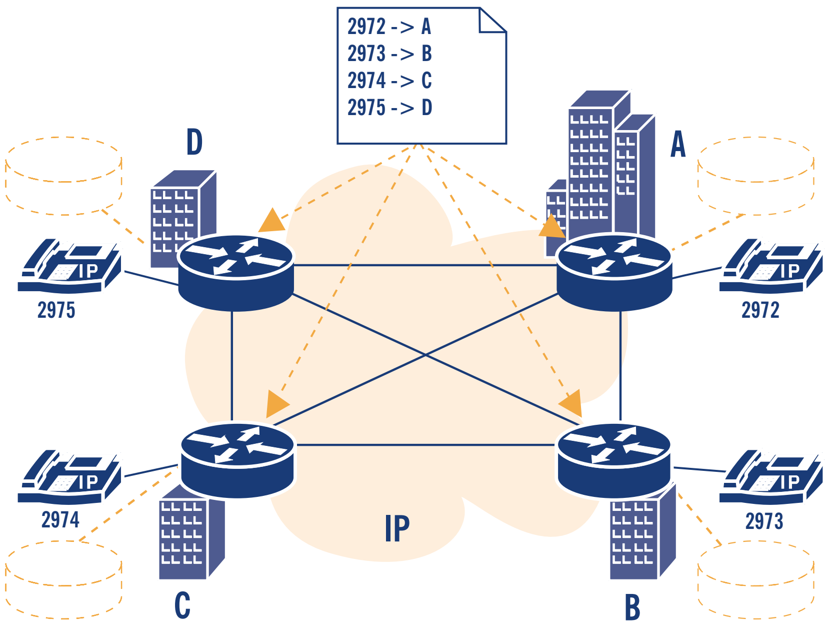


Figure 4.7 Static individual trunking

A more likely solution is a model 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 using at least one routing database, but often more. The requirement for such a burden presumably has its roots in the classic PBX mentality of static configuration. In the IP world, the network offers the possibility of carrying 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 un-registers from the server, thus allowing its peer servers to immediately make routing decisions if necessary. On the other hand, care must be taken that servers that are integrated later, be synchronised.

'Pull' mechanisms are used when a server asks its peer servers for a 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 the same time. Both signalling protocols may use TRIP (see Section 7.1.3) to distribute numeric addresses, but not names.

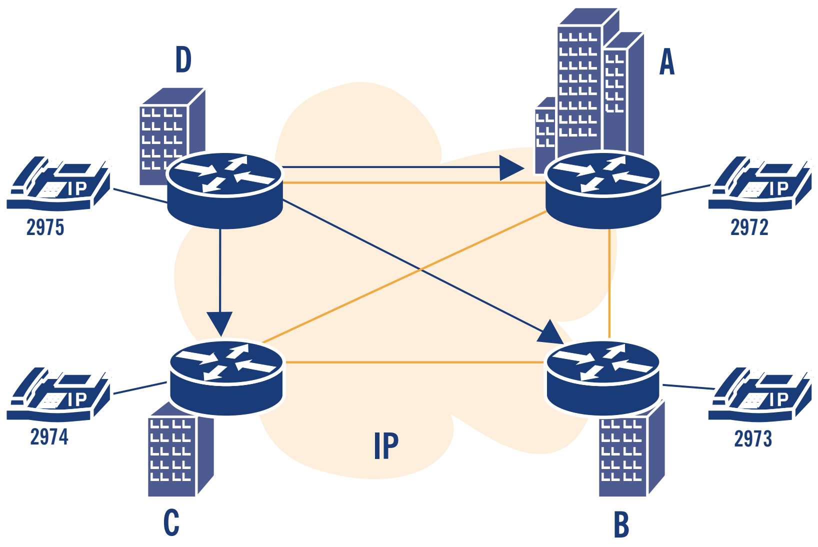


Figure 4.8 Dynamic individual trunking

~ 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 each of them as a separate router (see Section 4.1.1.2.3) that shares 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 will not even notice, at first, because the UDP media stream is usually transmitted directly between the two endpoints. But, the TCP signalling 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 prioritised 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 synchronisation between the replicated servers the failure of one server normally results in the loss of all calls. The server loss is discovered 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 for only a few seconds. Of course, active calls will still be 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. If 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 rewriting-mechanisms ARP might not work in switched networks, which would force both servers to be in the same shared network segment.

It is hard to give general advice on which kind of robustness-mechanism to use. 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 of putting 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 obtain.

Every manufacturer that offers IP Telephony solutions implements some robustness-mechanism. One should 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

The need 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 in 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 Decentralisation

Another issue occurs regarding the question of who is allowed to administer what in the telephone system. In classic environments, there is a small group of PBX administrators that sits in a special location, but in 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 these changes could be decentralised.

Decentralised 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 decentralised administration. Whether different administration permissions can be granted heavily depends heavely on the products used.

~ 4.2 Dial plans

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

1. **IP Telephony numberblocks**
   Usually, there is an already existing PBX dial plan and IP Telephony is to be integrated into this dial plan. If the existing dial plan has a free numberblock, 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 dialled 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 to change from a legacy phone to an IP Telephony phone but 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 if a call that originates on the IP-side has an IP Telephony target or a phone on the PBX. Again, this can be realised 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 the situation 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 being ‘visible’, the PBX might set up a mapping table that maps outdated old addresses to the new addresses, so the PBX maps the dialled 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). Whether 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. **Protocol- and number-based routing**
   The call routing problem gets worse as soon as multiple call-signalling protocols are deployed in the IP world and there is 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. 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 server if there is only one multi-protocol gateway).

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

~ 4.3 Authentication

To charge individuals for used services, it is necessary to have means of authentication for registration and call signalling. This section gives an overview of 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 signalling messages and Security for media streams.

Security for signalling 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 signalling 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 proves 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 some 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 9979. 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[2].

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 signalling 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.

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

~ 4.3.1.4 Confidentiality

Confidentiality ranges from secured H.225.0 signalling channels to secured media streams. The H.323/H.235 suite of protocols does not specify a way to secure the signalling 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 is 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 profile 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 has existed for some years, 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 communication 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, the 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

This section describes authentication mechanisms used in SIP-based networks. First the basics of digest authentication are described, while in following sections a brief overview is given 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 the 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 the 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 the 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 the case of SIP, it is usually set to the domain that 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 a 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**: this is an opaque data string passed to the user in a challenge. The user will pass the data string back to the server in a digest response. That allows servers to be stateless. If there is any state they need to maintain between challenge and response, they can pass it to the client using this parameter and read it again later when a digest response comes.
- **algorithm**: the algorithm used to calculate hashes. Currently only MD5 is supported;
- **qop**: the quality of protection. The parameter specifies what protection schemes the server supports. 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 a more detailed description, see RFC2617.

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:

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

The digest response is similar to the digest challenge. Those parameters that are the same have the same meaning as in the digest challenge. Below is a brief description of only the 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

The appearance of digest challenge and response has been described, but not 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.

When a SIP server receives a SIP request and wants to verify the authenticity of the user before processing the requests, it looks to see 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 if the verification was successful.

Proxy servers use the **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.
```

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

```
SIP/2.0 401 Unauthorised.
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 an **Authorisation** or a **Proxy-Authorisation** 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>.
Authorisation: 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

Above is a description of what digest authentication looks like and how digest challenges and responses are carried in SIP messages. This chapter looks at which SIP messages can be challenged and which cannot. It also describes the 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. These are described in separate sections.

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 hijacking 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 the call flow of a typical SIP registration including digest authentication.

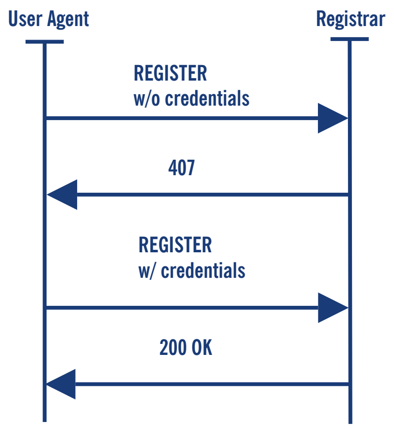


Figure 4.9 **REGISTER** Message Flow

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 that 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.**

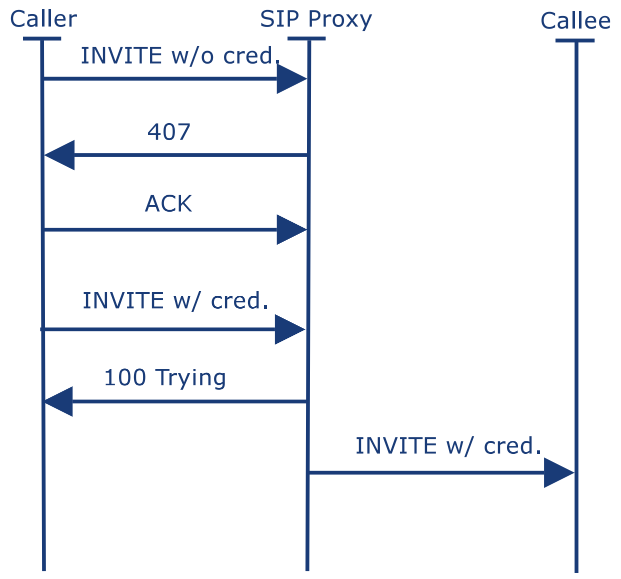


Figure 4.10 **INVITE** with authentication

~ 4.4. Examples

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

~ 4.4.1. Example 1: Simple use of IP Telephony like legacy telephony

Assumption: an institution currently using a PBX with internal numbers of four digits length. There are telephone numbers from 6000 to 6999 available for IP Telephony. There are no requirements regarding authentication. Because only calls into the PSTN will be billed, the PBX is the only place where billing will 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 signalling 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 have 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 passing the PBX.

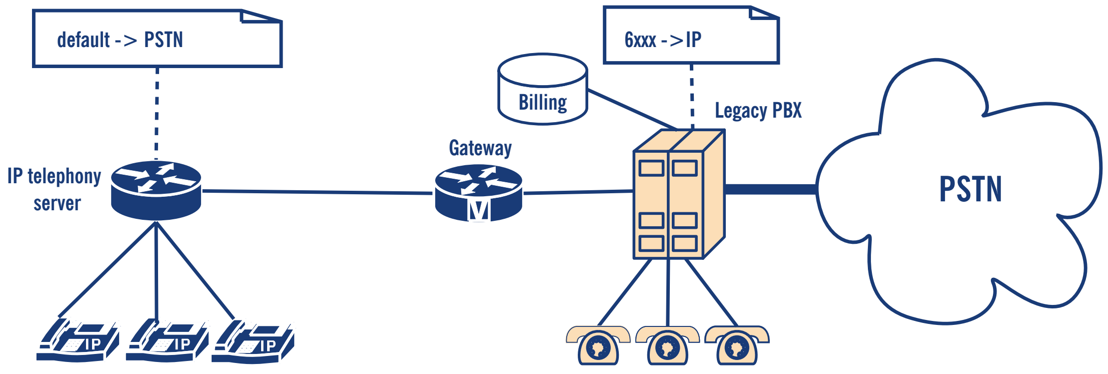


Figure 4.11 Example of simple IP Telephony

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

The disadvantage of this solution is that it relies heavily on the PBX, which remains the core element of the infrastructure. If there is demand for more IP Telephony accounts, more numberblocks must be available. To free such a block requires giving legacy phone users new numbers. The solution also does not make use of the Internet for long-distance calls or select an IP Telephony service provider.

~ 4.4.2 Example 2: An example of complex, full-featured IP Telephony

Assumption: a university with multiple locations, a shared, unstructured dialling space has a need for both SIP and H.323. It should be possible to test new IP Telephony server firmware before installing it 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 standardised protocols, the research network runs either an 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 an 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 organised. There is an 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 back-end 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 production 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 of a PSTN Telco.

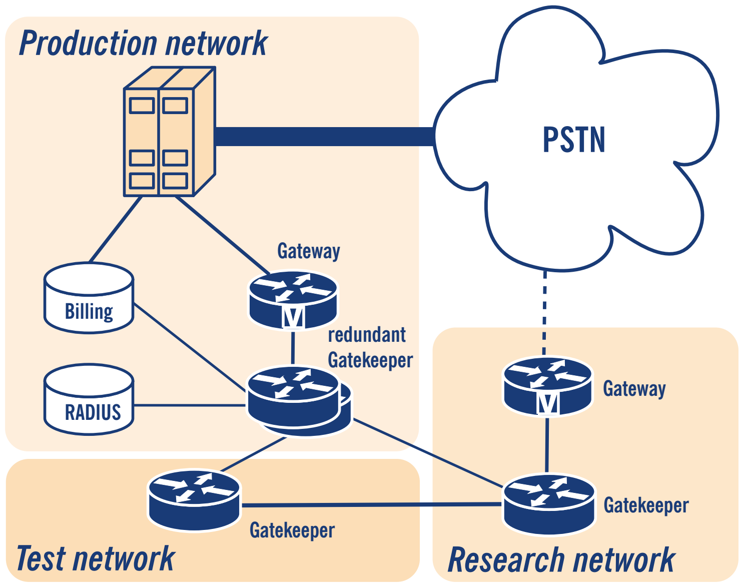


Figure 4.12 Example of a multi-server IP Telephony zone

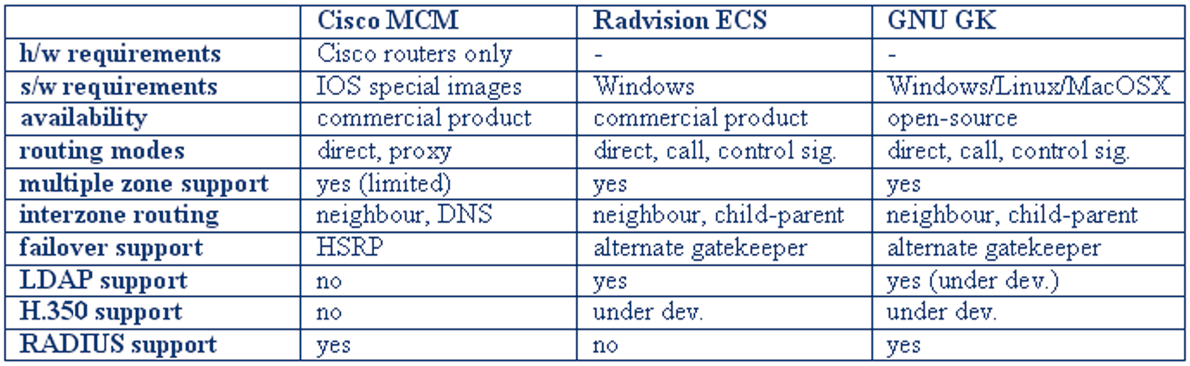
This scenario is quite complex, but it is the most flexible. It allows individual users to move from the legacy telephony world to IP Telephony, eventually reducing the PBX to a minimal state. It is made robust by using redundant servers where necessary. Because routing decisions are made on the IP-side, 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 space for research initiatives and prevents dependency 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, guides will be presented 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:



	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

Figure 4.13 Examples of gatekeeper features

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 organisations that use Cisco hardware, without regard for underlying operating system support. On the other hand, it does not allow the flexibility of installation 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 organisational borders. The MCM supports either direct mode dialling, 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 inter-zone routing features with DNS gatekeeper discovery as extra and performs well 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 online. [3]

~ 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 Website:
- Cisco IOS Upgrade Planner
- Cisco Software Advisor.

Look for 'High-Performance Gatekeeper' under features and for 'IP/H.323' under feature sets. IOS versioning is a subject difficult to follow. Add to this the fact, that the MCM has been

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

undergoing changes during IOS development and gatekeeper features available on different IOS versions vary significantly. 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 requires some experience with basic commands, modes of operation, loading software images and configuration files, none of which 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 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 initialises 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 customise/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 the 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 authorisation 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,
 ! as endpoints that request to be registered by e-mail address must match
 ! the specified "mydomain.org" part.
 ! The zone prefix is important for recognising
 ! 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 /
 !
```

Make sure no H.323 proxy services are unintentionally used, unless proxy functionality is needed for security or QoS reasons.

```
 !
 no use-proxy [myzone] default inbound-to terminal
 no use-proxy [myzone] default outbound-from terminal
```

~ 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 information 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 information and a password, separated by an administrator-defined special character, for example, 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 that stem mostly from the fact that it is a proprietary solution, which, in some cases, exposes clear text passwords to neighbouring 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 information. 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 utilise 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 RASVISION 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 specifications. It is servicing large organisations with a great number of endpoints and most notably, some of the VideNet global root gatekeepers. The ECS supports all three modes of routing: direct, Q.931 routing and both Q.931 and H.245 routing. The ECS has good inter-zone routing features with DNS gatekeeper discovery and neighbour gatekeeper LDAP support as extra. Authentication is very flexible, with the ability for ‘predefined’ endpoint settings enforced at registration time and LDAP H.350 support, but no RADIUS support.

~ 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 already be 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 (see Figure4.14) 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 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.

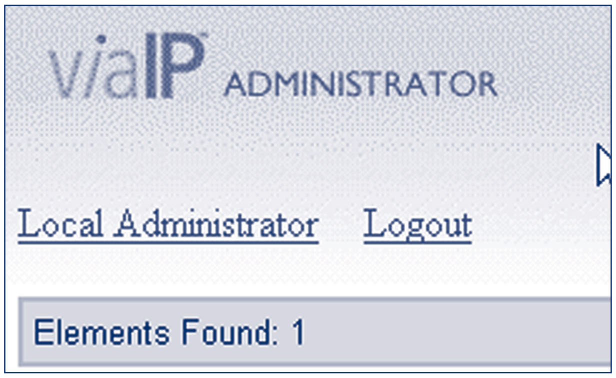
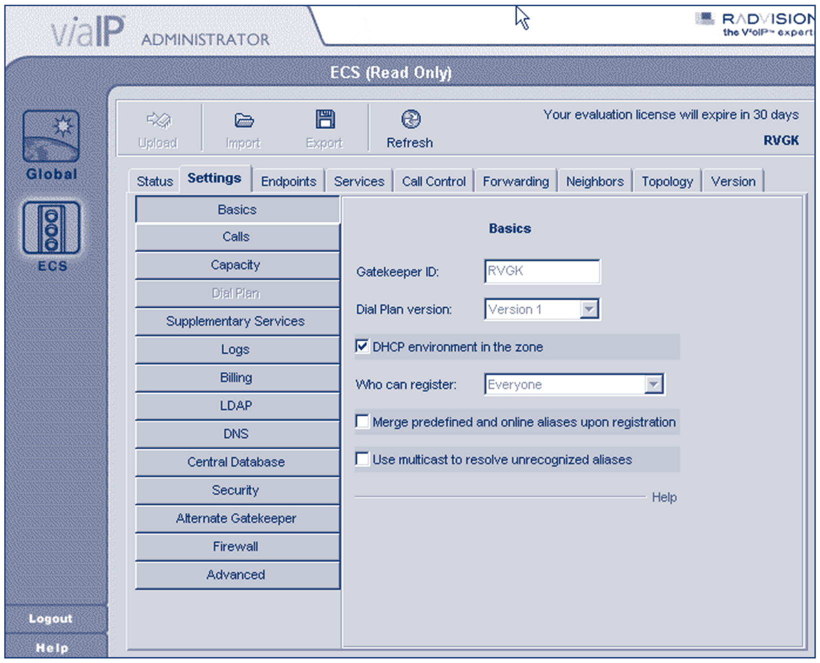


Figure 4.14 ECS local administration entry

Immediately afterwards, the menus for the administration of the locally-installed gatekeeper are shown, as below.

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.



Your evaluation license will expire in 30 days

RVGK

Basics

Gatekeeper ID: RVGK

Dial Plan version: Version 1

☑ DHCP environment in the zone

Who can register: Everyone

☐ Merge predefined and online aliases upon registration

☐ Use multicast to resolve unrecognized aliases

Figure 4.15 ECS administration menus

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 the **Signalling Models** Section 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 **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 alter 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 **Dialplan**, assuming you have chosen dial plan version 2, you will be able to specify the stripping of zone prefixes from destination information 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 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 neighbouring 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 one of the registered aliases (name or number) to 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 un-register an endpoint from the respective tab sections. Logs of 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 on 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 dialled destinations by their 12-digit number, while an organisation-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 standardised protocol for storing and retrieving user settings and preferences regarding H.323 and SIP services. RADVISION is an active partner in the committee that developed the H.350 standard (previously known as H.LDAP or CommObject) and 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 redialling should be successful, without requiring the endpoints to register to a new gatekeeper.

~ 4.5.3 Using an Open H.323 Gatekeeper - GNU Gatekeeper

The GNU GK is the most popular and active in the 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 with 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.

~ 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.

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 that 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 authentication and authorisation 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 this should work if the configuration file (**gnugk.ini**) is correct.

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

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

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

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

Regarding 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 customise/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 authorisation rules are detailed in the [GkStatus::Auth] section)
[Gatekeeper::Main]
Fourtytwo=42
#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 h245Tunnelling is set to TRUE
#This may cause interoperability problems, avoid setting this option to 1
RemoveH245AddressOnTunnelling=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

#Authorisation 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. 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 visualise the operations:
- Java GUI: This 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;
- GkGUI: A new standalone Java program. It requires Java 1.4. The GkGUI is released under GNU General Public License.

~ 4.5.3.4 Endpoint authentication

The GNU Gatekeeper supports all three RASIUS, MySQL and LDAP back-end interfaces (LDAP is still under development) for registration (RRQ) and admission (ARQ) authentication and authorisation mechanisms. This is obviously a very complex as well as flexible environment in which to implement authentication and authorisation 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 the following configuration sections on the manual Web page for a more detailed configuration descriptions 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

~ 4.6.1 Operation of SIP Servers

~ 4.6.1.1 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.

4.6.1.1.1 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 for the problem. We 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**.

4.6.1.1.2 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 Unauthorised.
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 ".
```

4.6.1.1.3.Tracing Errors in Server Chains.

A request may pass any number of proxy servers on its path to the destination. If an error occurs in the chain, it is difficult for upstream trouble-shooters 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, a trace-route-like tool for SIP developed at `iptel.org`. It allows you to send an OPTIONS request with low, increasing **Max-Forwards** header-fields and follow how it propagates in the SIP network. See its Webpage at `http://sipsak.berlios.de/` .

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.

4.6.1.1.4 Server status monitoring.

It is essential for solid operation to monitor server status continuously. Two tools have been used 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.

4.6.1.1.5 Dealing with DNS.

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

- Start a sufficient number of child 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 statefully if memory allows. That helps to absorb retransmissions without having to make DNS queries for each of them.

~ 4.6.2 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.

~ 4.6.2.1 Getting SIP Express Router

SIP Express Router is available for download from BerliOS
The newest release can be found in the folder **/latest**.

~ 4.6.2.2 Installation (From binary packages)

4.6.2.2.1 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.**

4.6.2.2.2 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**, and **make sunpkg** to work.
* **mysql** if you need MySQL support.
* **apache (httpd)** if you want serweb support
* **PHP**, **MySQL-PHP**for serweb support
* **libmysqlclient** and **libz** (**zlib**) if you want MySQL support (the MySQL module)
* **libexpat** if you want the Jabber gateway support (the Jabber module)

4.6.2.2.3 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
```

Red Hat-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 the starting 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.

~ 4.6.2.3 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 packages 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.

Another account needs to be the administrator for your realm. That will be done later.

~ 4.6.2.4 Configuration

4.6.2.4.1 Overview

This section demonstrates simple examples of 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 loading 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 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 are triggered by received replies. Restrictions on use of actions within the **failure_route** blocks apply (see SER Administrators' Guide for more details).

4.6.2.4.2 Default configuration script

The configuration script, **ser.cfg**, is a part of every SER distribution and defines the 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 the user's identity first to avoid unauthorised 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 statefully 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 the 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 the 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 un-commenting-related configuration options, the server will assume that the back-end authentication database contains the 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 Example of a 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 (!_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();
  };
}
```

4.6.2.4.3 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 longer 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");
}
```

4.6.2.4.4. 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 returning to request processing when the 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 the **t_on_failure** action is called before a request is statefully 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 the final reply may take a long time until the called party (finds a mobile phone in his pocket and) answers the call.

In Example 4.5, **failure_route** [1] is set to be entered in 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();
}
```

4.6.2.4.5 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 the 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 the achievement of 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 and requests need to be processed statefully and labelled 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 dialogue (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 on record routing.

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;
  };

}
```

4.6.2.4.6 Reporting missed calls

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

Reporting of missed calls is enabled by the acc module. There are two cases, in which you want to report. The first case is when a called party is offline. The other case is when a user is online, but call establishment fails. There may be many reasons for failure (call cancellation, inactive phone, busy phone, server timer, etc.), all of them leading to a negative (**>=300**) reply sent to the calling party. 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 called party 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 statefully (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;
};
```

4.6.2.4.7 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 calling parties 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 otherwise be 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")
```

~ 4.6.2.5 Operation

4.6.2.5.1 User management

There are two tasks related to the management of SIP users: maintaining user accounts and maintaining user contacts. Both of 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 e-mail 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
```

Typically, user contacts are 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 a 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 s**erctl ul add <username> <contact>**. To delete all users' 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
```

4.6.2.5.2 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, unauthorised users would be able to use it to make calls to the PSTN, inflicting high costs.

There are a 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, calling parties may easily bypass your access control server and communicate with the gateway directly. You then need to enforce, at transport layer, that signalling 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, calling party's devices would send subsequent SIP requests directly to your gateway, which would fail because of transport filtering.

Authorisation (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 calling party is authorised 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 a 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 calling party 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;
        };

    # authorise 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 authorised PSTN

    }; # authorised 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 |
+------+-----+---------------------+
```

~ 4.6.3 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.

~ 4.6.3.1 Getting Asterisk

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

~ 4.6.3.2 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
```

~ 4.6.3.3 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 the dialling plan.

In this simple example, Asterisk is configured 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. Configure Asterisk to listen on port 5060 and to listen on all available interfaces. 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, the parameter **fromdomain** is specified, 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, one SIP phone is configured with username **jan** and with an empty password and with a phone that 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 already configured in **sip.conf** . The following lines describe the dialling 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 an exact match will be accepted.

In the 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 dialling 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 calling party ID. If a call fails then Asterisk will reply with an error message.

~ 4.6.4 VOCAL

~ 4.6.4.1 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). VOCAL also supports analogue 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

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

4.6.4.2 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`
  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.**

4.6.4.3 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 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:

```
/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 the installation is OK, you should see the following text:

```
VOCAL basic call test passed.
```

~ 4.6.4.4 Operation

VOCAL 1.5.0 comes with two provisioning systems. Both work on the same configuration data. The first, 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.

~ 4.6.4.5 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.

~ 4.6.4.6 Advanced features

There are a lot of advanced features that can be deployed on VOCAL architecture. In this section, an overview of these features is presented. 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 specialised servers. Last, but not least, a voice mail server is used to deliver mail with attached voice files containing messages registered to the users.

~ 4.7 Firewalls and NAT

Firewalls and Network Address Translation (NAT) affect IP Telephony signalling protocols, making it impossible to call targets outside the private or protected network. While firewalls and NATs often go hand in hand, they impose two different problems which are 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 signalling 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 the minimum requirement.

After signalling 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 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 them.

~ 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 signalling, but the messages carried in the application layer contain IP addresses that are not recognised 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 signalling: Some call signalling messages contain IP addresses that are necessary for subsequent communication, e.g., the IP address (and port) to which media data will 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[4].

4.7.3 SIP and NAT

4.7.3.1 Overview

Since the start of deployment 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 its beginning, SIP has been designed without considering NAT scenarios. This allows the design of 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 or are 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 work 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. This section describes a solution that is based on ICE, but with 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.

4. *<http://sourceforge.net/projects/openh323proxy/>*

The price for the simplifications will be the use of an RTP relay in cases where it is not absolutely necessary, but such cases seem to be quite rare. The 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 signalling

One of requirements laid on SIP devices that should work behind NAT is support for symmetric signalling. 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 the proper port and not to the port from which the request was sent, and the port number will be also registered in the 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 and usually it is some high port number assigned by the operating system.

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 that if a host with a 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 is no traffic for some interval. However, 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 signalling. In other words, the SIP user agents, using symmetric signalling, send SIP messages from the same source port on which they receive incoming SIP messages (usually 5060). This is the only that way they will be able to create a binding in the NAT that will allow incoming SIP traffic through.

Fortunately, the vast majority of developers of SIP devices seem to understand this, so almost all SIP devices that are available today support symmetric signalling. That includes Cisco phones, Windows Messenger, Mitel phones, Grandstream, Snom, X-lite and kphone (from version 3.13).

Support for symmetric signalling is a must and SIP user agents not supporting it will not work behind NATs.

4.7.3.1.2 Symmetric media

There is a similar problem 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.

Similarly, a user agent behind NAT must send a media packet with a source port that is same as the port on which the user agent expects media from remote party and which was advertised in the SDP. This is the only that way media will be able to pass the NAT in both directions.

In addition, the remote party (which is in the public Internet) must ignore the 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 is coming. That will be the public IP of the NAT box.

This approach is called Connection-Oriented Media and 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 the IP and port to which it should send.

When both parties are behind two different NATs, 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 the 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 ports 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 signalling and symmetric media. In addition to this, they should also be able to use an outbound proxy, because all SIP traffic has to go through a SIP Proxy in the Internet.

The vast majority of SIP user agents available today can work properly behind NATs.

~ 4.7.3.3 Support in the 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;
- Periodically send 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`. Below is a brief description 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 a NAT and the information is saved into a 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 a NAT and the information is saved into a user-location database.

For user agents behind NAT, registrar rewrites IP addresses and ports in the **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 of the need to minimise the use of RTP proxy.

When the server receives an **INVITE** message, it tries to find out if the calling party is behind a NAT. It checks again 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 calling party is behind a 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 the called party and the calling party and it is not necessary 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. This mark will be used later.

After this, 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 called party. You might remember that flags were saved about presence of NAT when processing the registration. If the called party is behind NAT (the flag in the 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 calling party or called party 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 called party 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 the use of the RTP proxy by rewriting the contents of SDP The same applies to **200 OK**.

It has been mentioned that the use of an RTP proxy is not forced when the called party is in the public Internet and does support symmetric media, but how do we know that? Indeed, there is currently no way of finding out whether a SIP user agent supports symmetric media or not. That means that an RTP proxy is not forced, except for destinations that are known to be symmetric, like voicemail or a PSTN gateway.

~ 4.7.3.4 RTP Proxy

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 use 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 the first media packets from both sides, it records the IPs and ports from which the packets came and starts relaying the media packets.

4.7.3.4.2 Drawbacks

Use of the RTP proxy has some drawbacks that are worth mentioning. First of all, it introduces another hop on the path from one user agent to another and this results in increased delay. How much the delay is increased depends on the underlying network and the location of the user agents.

Secondly, the use of an RTP proxy imposes more burden on the server, because all of the media traffic has to go through the server. For example, for G.711, it is 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

This section describes how to set up a SIP Express Router and an 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 describes only the settings necessary for NAT traversal. For the complete configuration guide, please refer to Section 4.6.2.

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. The second parameter is the interval (in seconds) for keeping messages alive (to keep the NAT bindings open), and the last parameter specifies that only contacts that are behind the NAT should be pinged.

To check if the sender of a message is behind a NAT, put the following test at the beginning of the main routing section of the configuration file (right after the test for messages that are too large):

```
# 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 signalling". (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, perform a similar test for the called party and force the use of an RTP proxy, if necessary. Because, potentially, there can be many places in an average configuration script from which to send out messages (all occurrences of the **t_relay()** or **forward()** actions). Put the whole test into a separate route section and call the section **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 the use of the RTP proxy. Also setup an **onreply_route** section which will process **200 OK** messages (it is necessary rewrite the **Contact** header field and SDP body in the reply).

4.7.3.5.2 RTP proxy

The installation and running of the RTP proxy is very simple and straightforward. First of all, download the proxy from the PortaOne site.

Untar the archive and compile the proxy using:

```
make -f Makefile.gnu
```

This will generate a binary called **rtpproxy**. Start the proxy using `./rtpproxy and restart SER.`

SER and the RTP proxy use the socket `/var/run/rtpproxy.sock` to communicate.

5 Setting Up Advanced Services -/

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 deals with an analysis of the characteristics of gateways and the configuration principles for the gatewaying functions to be set up in an advanced environment. The topics detailed in this section range 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 a 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 the 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 a PSTN, the PBX can forward the call directly to the PSTN (e.g., over ISDN) or it can forward the call to a voice gateway, which forwards it to a selected PSTN operator over the Internet.

There are different kinds of PBX to voice gateway interfaces with different features and costs. Your choice of the interface-type will probably depend on which features you require: acceptable cost, availability and whether there is already some interface present in the PBX and the voice gateway. In this section, some technical details are provided on different kinds of PBX-to-voice gateway interfaces. Signalling systems are also described briefly, such as Channel Associated Signalling (CAS), E&M signalling methods, Q-signalling and the Q.931 call control protocol and examples are given of exchanged messages during correct communication.

-/ 5.1.1.1 Subscriber Loop

A subscriber loop, also called a U-interface, is a 2-wire interface used primarily when connecting a telephone set to a Subscriber Line Module Analogue (SLMA). SLMA is the name of the analogue module in a PBX. A corresponding module in a voice gateway is called a Foreign Exchange Station (FXS).

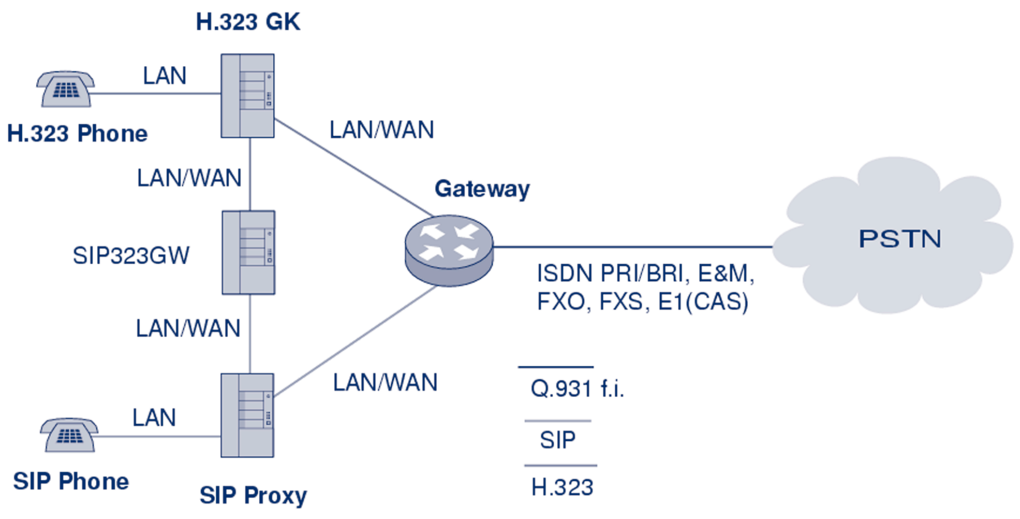


Figure 5.1 Voice gateway interfaces - PBX role

FXS and SLMA modules can also be interconnected to trunk modules. Trunk Module Analogue (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 common terminal equipment). This is a very simple mode. It only detects a ringing signal and provides digit dialling 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 waits for a closed loop that will generate a current flow and a signalling tone of 425 Hz (with 10% tolerance). The Subscriber Line 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 signalling are loop-start and ground-start signalling. 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 sixteen possible combinations, twelve are used on the keypad. DDI (Direct Dialling In) is possible only through a DTMF suffix, that is, during the connection time when the calling party normally is already paying for the connection.

-/ 5.1.1.2 E&M interfaces

E&M is commonly explained as both 'Ear and Mouth' and '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 signalling;
- one pair of wires is used for the outgoing voice path;
- one pair of wires is used for the incoming voice path.

This 6-wire connection can be reduced into 4-wires:
- one pair of wires (wires E and M) is used for signalling;
- one pair of wires is used for the voice path in both directions (which can cause a problem with echo cancellation and inhibits a possibility to use an amplifier).

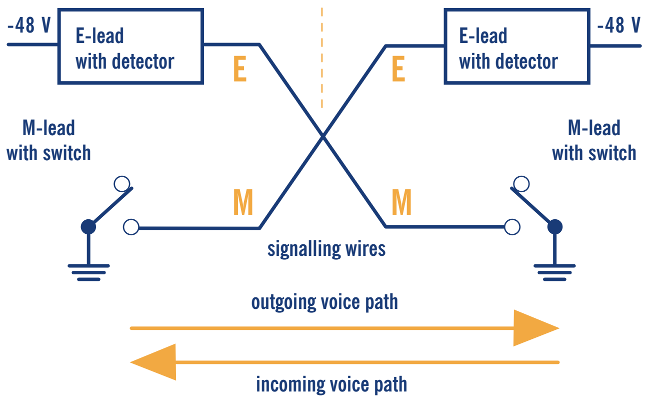


Figure 5.2 E&M signalling, type V

Signalling is carried out with direct current via the E & M control wires for call setup and tear down, pulse dialling and remote blocking. DTMF signals can alternatively be used for dialling. The E&M signalling can operate in several modes:
- continuous signal;
- wink start signal;
- delay dial;
- immediate dial.

-/ 5.1.1.3 E1/CAS trunk

CAS (Channel Associated Signalling) exists in many varieties that operate over analogue or digital interfaces. A common digital interface with CAS signalling is called E1 (European version). The physical layer works 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) signalling 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 authorisation;
- free calls;
- called party hold.

-/ 5.1.1.4 ISDN Access Interfaces

ISDN (Integrated Service Digital Network) is a currently preferred method of PBX to VoGW interconnection. It is described in more detail and some illustrative examples of exchange of its Q.931 signalling messages are given. ISDN is a system of digital connections allowing the establishment a call with end-to-end digital connectivity of nx64kbps. Original recommendations for 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 the 1TR6 protocol in Germany, NI (National ISDN) in the USA and the 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. Twenty-six communication organisations signed the 'Memorandum of Understanding on the Implementation of a European ISDN' in 1992. The signing countries were obliged to offer a common technological substructure for ISDN network development, connecting all of Europe. As a result, Q.931 signalling has been internationally standardised.

Two types of access methods exist for ISDN:
- BRI (Basic Rate Interface);
- PRI (Primary Rate Interface).

BRI delivers two 64 kb/s B channels and one 16 kb/s D channel. The reference configuration of ISDN defined in the ITU specification I.411 is illustrated in Figure 5.3

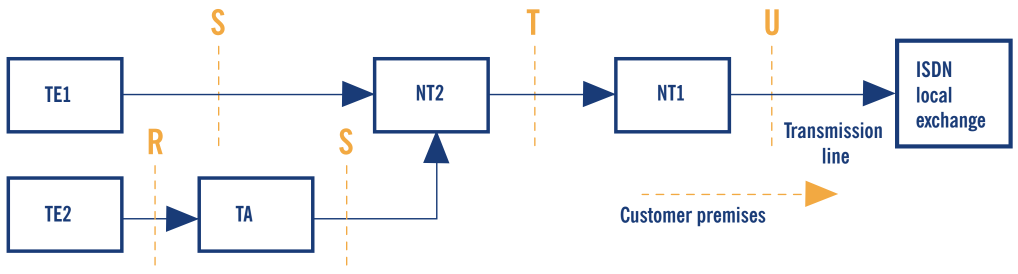


Figure 5.3 ISDN configuration

U interface is a two-wire interface from the telephone exchange; it supports full-duplex data transfer over a single pair of wires: only a single device can be connected to a U interface.

T interface is between network terminations NT1 and NT2: NT1 converts two-wire U interface into four-wire T interface.

S interface and T interface are electrically equivalent. ISDN devices include an NT-2 in their design. The NT2 communicates with terminal equipment TE1 (ISDN terminal) or TE2 (non-ISDN, connected to NT2 via terminal adapter), and handles the layer 2 and layer 3 ISDN protocols.

Network Termination NT is divided into NT1 and NT2; NT1 works 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 an 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.).

Because a PBX can provide NT2 functions, the T interface is commonly used for interconnection of a PBX and a Voice Gateway. The PBX works 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 Signalling 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 and debugging was enabled with 'debug isdn q931' command;

- The call was initiated from the Technical University in Ostrava to the Czech Technical University in Prague,. The PBX was connected through ISDN/PRI and the called number was sent as the en-bloc in the SETUP message;
- The Calling Party Number was 596991699
- The Called Party Number was 224352979

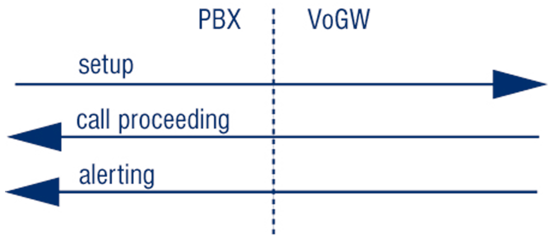


Figure 5.4 Q.931 call control messages in call-setup with the en-bloc signal

```
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.

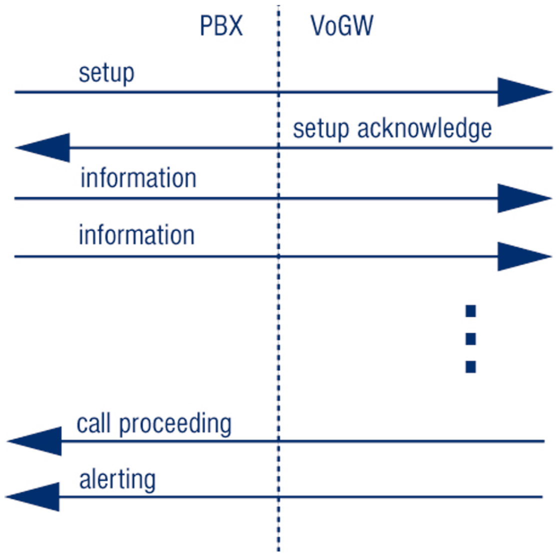


Figure 5.5 Q.931 call control messages in call-setup with overlap

This example corresponds to the following setup:

- Cisco AS5300 was used as a Voice Gateway; debugging was enabled with a **debug isdn q931** command;
- The call was initiated from the Czech Technical University in Prague to PSTN (Public Switched Telephone Network); the PBX in Prague was connected through ISDN/PRI and the called number was sent as the digit by digit (overlap) in the **SETUP** and **INFOMATION** messages;
- The Calling Party Number is 224355406;
- The Called Party Number is 224324997.

```
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 calling parties (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 the PSTN and call signalling to H.323;
- termination of incoming calls from H.323 and forwarding to the PSTN;
- accounting for calls utilising 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 interconnection with PSTN. Moreover, details on the configuration of gateways are given in order to provide guidelines on how to configure gateways to be part of an H.323 network.

-/ 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 specialised 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

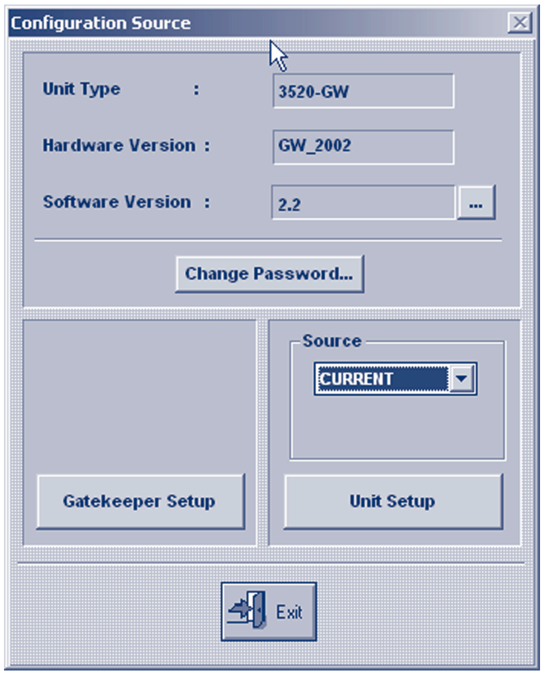
Its installation is straight-forward, requiring merely power, a network interface, BRI ISDN interfaces and a PC on the same LAN for setting-up initial configuration parameters through a windows-based application. To manage the device, install the RADVISION OnLAN Tools from the installation disks, by 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**).

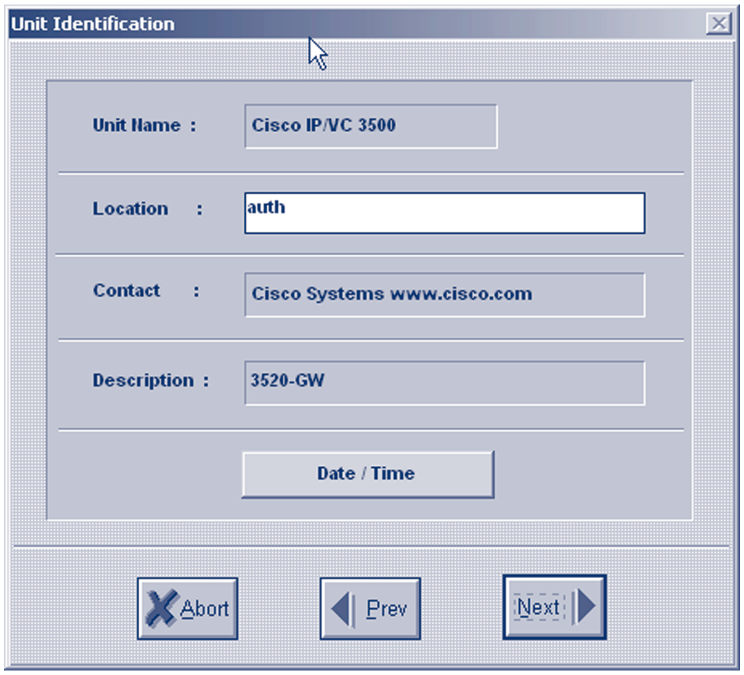
Only the second option is of interest, since the gateway can register with any of the gatekeepers detailed above, and the built-in gatekeeper can be turned off since it provides only basic functionality. Select **Unit Setup** and proceed with **Unit Identification** and **Date/Time** options, which are informational only, by pressing the **Next** button.

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.



Unit Type : 3520-GW
Hardware Version : GW_2002
Software Version : 2.2
Source: CURRENT

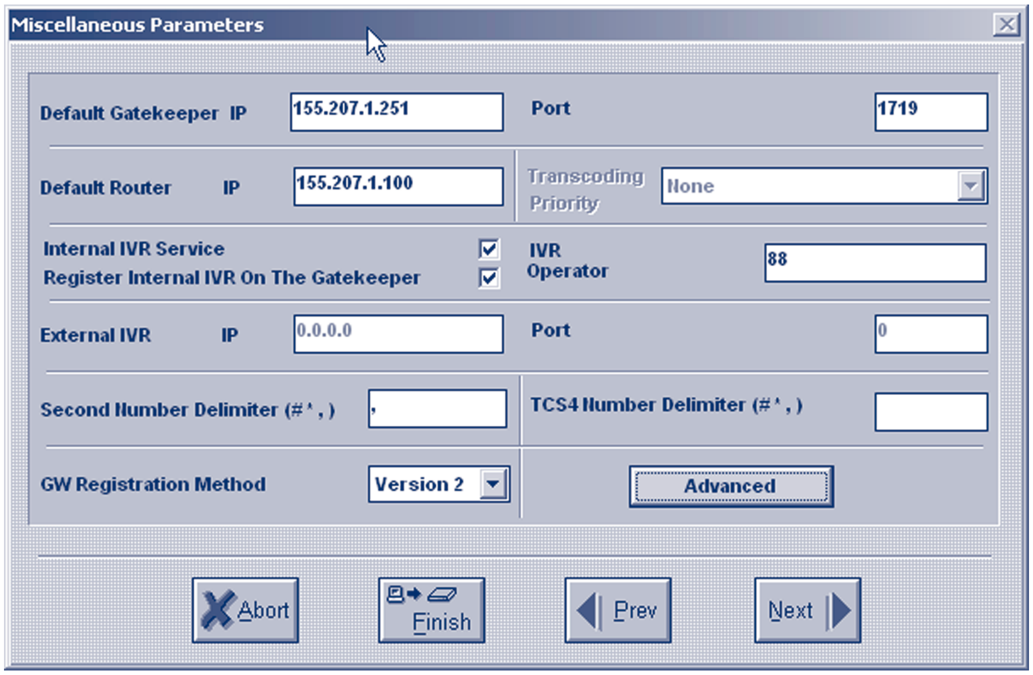
Figure 5.6 OnLAN configuration entry



Unit Name : Cisco IP/VC 3500
Location : auth
Contact : Cisco Systems www.cisco.com
Description : 3520-GW

Figure 5.7 OnLAN unit identification

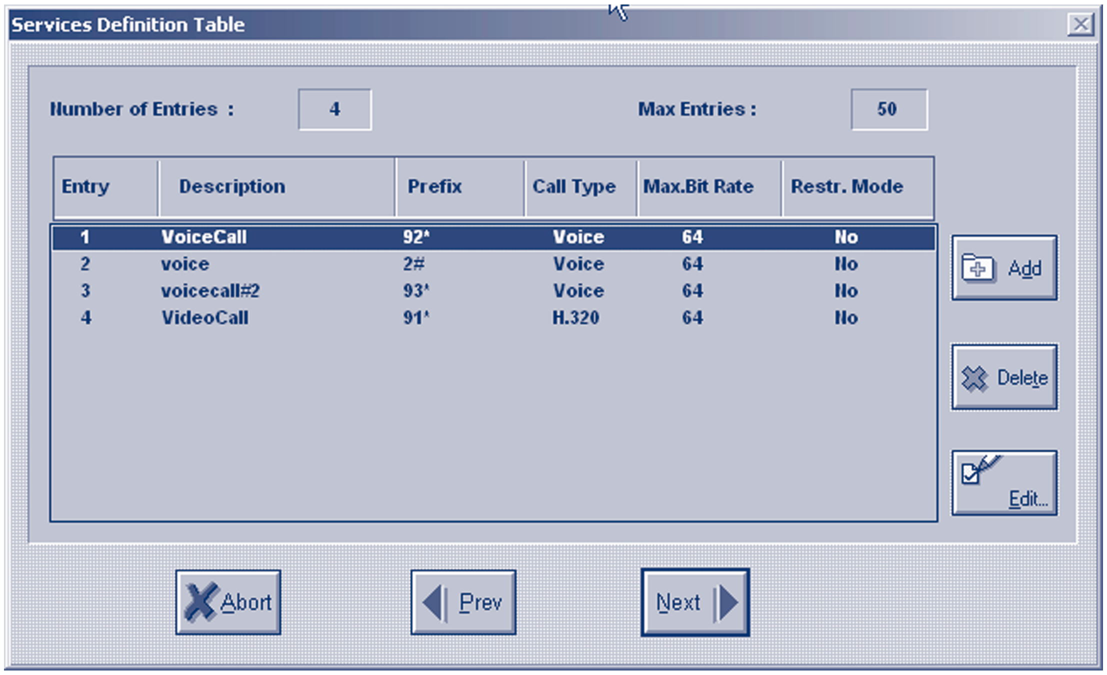
The next screen, **LAN Port Settings**, is responsible for configuring the Ethernet interfaces. There are four screens, one for each of the four Ethernet ports. Configuring just one interface with an **IP Address** and **IP Mask** is sufficient. A summary screen with settings for all four ports follows.


Miscellaneous Parameters

Default Gatekeeper IP	155.207.1.251	Port	1719
Default Router IP	155.207.1.100	Transcoding Priority	None
Internal IVR Service	✓	IVR Operator	88
Register Internal IVR On The Gatekeeper	✓		
External IVR IP	0.0.0.0	Port	0
Second Number Delimiter (# * ,)	,	TCS4 Number Delimiter (# * ,)	
GW Registration Method	Version 2		

Figure 5.8 OnLAN miscellaneous parameters and gateway registration on gatekeeper

The next screen, **Services Definition Table**, is the most important one, since it defines the services that the gateway will make available to calling parties, by registering them with its controlling gatekeeper.

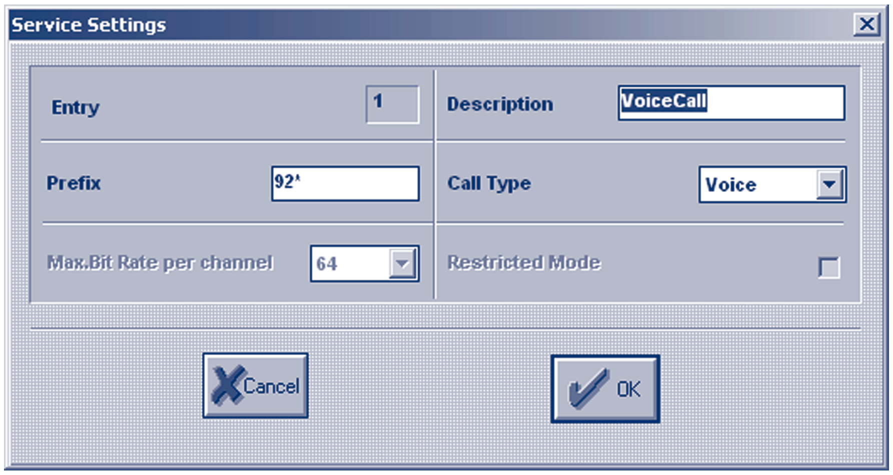

Services Definition Table

Number of Entries : 4 Max Entries : 50

Entry	Description	Prefix	Call Type	Max.Bit Rate	Restr. Mode
1	VoiceCall	92*	Voice	64	No
2	voice	2#	Voice	64	No
3	voicecall#2	93*	Voice	64	No
4	VideoCall	91*	H.320	64	No

Figure 5.9 OnLAN-defined gateway services

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 and will be required. By defining different services, the calling parties 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 customise and that is the service prefixes, in order to make them match your local dialling scheme.



Entry: 1
Description: VoiceCall
Prefix: 92*
Call Type: Voice
Max.Bit Rate per channel: 64
Restricted Mode: ☐

Figure 5.10 OnLAN editing of a service definition

The next screens refer to **WAN Port Settings** which, for an ISDN interface, present relative settings. The 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 port is 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 and 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 the 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 unregister 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 piggy-back 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 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, an example of how a Cisco Voice Gateway should be configured for interconnection with a PBX is presented. See Figure 5.11 for an illustration of the interconnection. The same procedure could be used for interconnecting a Cisco Voice Gateway with PSTN/ISDN

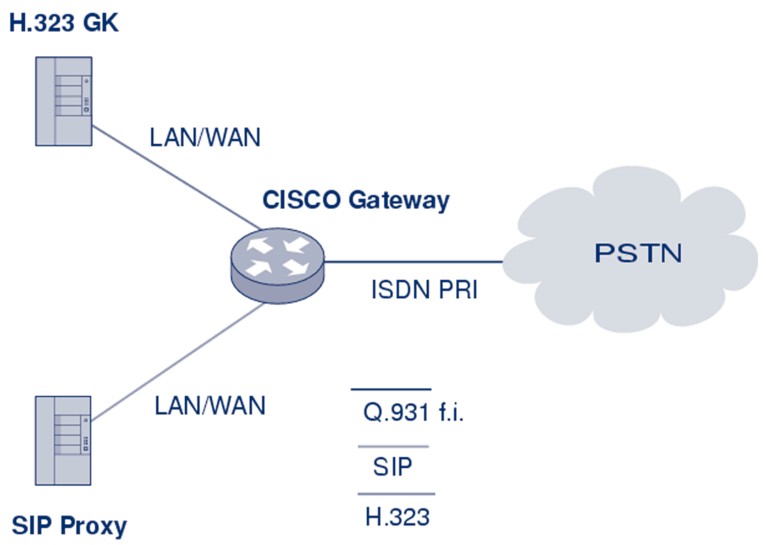


Figure 5.11 Cisco voice gateway interconnection

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 2651XM with appropriate IOS is connected through PRI/ISDN to the PBX and is registered to a gatekeeper. Examples of the configuration were cut off from the Cisco CLI command and show **running-config output** and **comments.**

5.1.2.2.1 Protocol configuration

Gatekeeper peering and gateway id are set up in this section. The Cisco gateway could serve as a voice gateway for both H.323 and SIP protocols at the same time, but the SIP protocol-specific part is configured in a completely different configuration section of the gateway (not the sip-gateway, see Section 5.2.2). Use of the **h323-gateway** command is recommended, set in a loopback interface configuration section. Use of a loopback IP address in registration and accounting messages helps to manage the system easily if the gateway has more than one interface used for VoIP.

```
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 a PBX or a PSTN also has to be properly set up at interface level **(router(config-if)#)**. Telephony interface configuration is independent on the used VoIP signalling protocol. In this section, there are configured ISDN parameters for signalling the channel (the 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 the DSS1 (EuroISDN) signalling protocol on the primary interface PRI (basic-net3 is used on the BRI).

timer T302 is the 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 and 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

The next configuration step is setting up the call rules. This is done by a **dial-peer** command set from the basic config mode (**router(config)#**) of the Cisco gateway. If the gateway should provide calls from both sides (to and from the PSTN/PBX), a set of at least 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 the PSTN(PBX) from the VoIP-side.
The **called number** prefix is 42037763 and must be followed with four digits of extension (four dots substitution pattern). The **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 to interconnect and allows the resolving of a possible problem with the ring-back tone.

The rules are written into the port interface **0/0:15** with **DDI** (Direct Dialling In).

```
dial-peer voice 112 voip
  destination-pattern 420.........
  session target ras
  no vad
```

These commands specify rules for calls leading to the VoIP-side from the PSTN(PBX):
- The called number prefix is 420 and it must be followed with nine digits;
- The target of this session is the gatekeeper, accessible through RAS signalling;
- The last command specifies that the Voice Activity Detection is not possible (higher voice quality);
- The default configuration is VAD with CNG function (Comfort Noise Generation).

The user is reminded here that the listed configuration may not be sufficient to run a voice gateway on Cisco routers. Commands may depend on the type of the router and the IOS version used.

-/ 5.1.2.3 Gatewaying H.323 using a GNU Gatekeeper

How to set up services using GNU Gatekeeper was already described in Section 4.5.3. Here, enhanced configuration and operation for GnuGK to be used with a gateway in order to reach people who are using ordinary telephones is described.

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 tells the gatekeeper the prefix of numbers that will 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 the 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 these alias', the call will fail. Note that you must use the gateway alias. You cannot 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 such a solution is not advised, because it leads to errors when a network reconfiguration is done.

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 this.

[RasSrv::RewriteE164]
12345=08765

In this example, the gatekeeper is configured to replace the numbers 12345 at the beginning of the E.164 number dialled to 08765 (for example, 12345-99 is rewritten to 08765-99). Please refer to **rewrite** 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.Â Gatewaying from SIP to PSTN/ISDN

-/ 5.1.3.1 Gatewaying SIP using Cisco

Configuration of a SIP gateway is almost same as the configuration of an H.323 Gateway and because configuration of an H.323 Gateway is already described in Section 5.1.2.2, the same settings are not described again here (for an interface configuration which is exactly the same). Only configuration specific to SIP is described. 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 is replaced by **sip-server.** For 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 the **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 configure the gateway to be passive. That is good for the **Connection Oriented Media**. Value. Passive means that if there is a **direction=active** parameter in SDP, then the gateway will wait with the sending of media until it receives the first media packet from the remote party. This feature can be very useful for NAT traversal. It can be enabled 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 its media there if it is in a symmetric mode. This option is related to the previous one.
Retry. Parameters specify how many times various SIP messages should be retried.
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.

-/ 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 loose the native protocol, (H.323 / SIP) supplementary services. These types 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. For the 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 a SIP network to an H.323 network and vice-versa. The entities making the calls using such a gateway can be:
- a SIP user agent;
- an H.323 terminal;
- a SIP Proxy Server;
- an H.323 Gatekeeper;
- another gateway (e.g., SIP-PSTN).

Various types of operations can be performed using a SIP/H.323 Gateway and can be basically divided into five categories (detailed descriptions, follow this section):
- a user registration;
- a call from SIP to H.323;

- a call from H.323 to SIP;
- media-switching and capabilities-negotiation;
- a call termination.

There are different modes in which a SIP/H.323 Gateway can operate. In this section, basic operation modes are described. For more detailed operations, please refer to 'Interworking Between SIP/SDP and H.323' for examples of basic functionalities 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 on whether the internal gatekeeper/internal SIP register is activated or not.

If a SIP/H.323 Gateway is configured appropriately, it should try to register both with an H.323 Gatekeeper using RAS procedures and with a SIP registrar, in order to perform the gateway's address resolution from either side. A SIP entity can query the registrar, whereas an H.323 entity can query the H.323 Gatekeeper to locate the SIP/H.323 Gateway. If the internal gatekeeper is activated, then no registration to an external gatekeeper should be performed. If the internal SIP registrar is activated, then no registration to an external SIP registrar should be performed. At least one H.323 Gatekeeper and at least one SIP registrar should be always specified/activated for operations to be successful.

If an 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, an 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 (a **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, details are given on different architectures for user registration/address resolution. User registration servers are SIP registrars and H.323 Gatekeepers. If a SIP/H.323 Gateway has direct access (either built-in or configured) to user registration servers, this simplifies locating users independently of the signalling 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:

1. The SIP/H.323 Gateway contains a SIP Proxy and registrar (Figure 5.12). 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 the H.323 Alias Address;

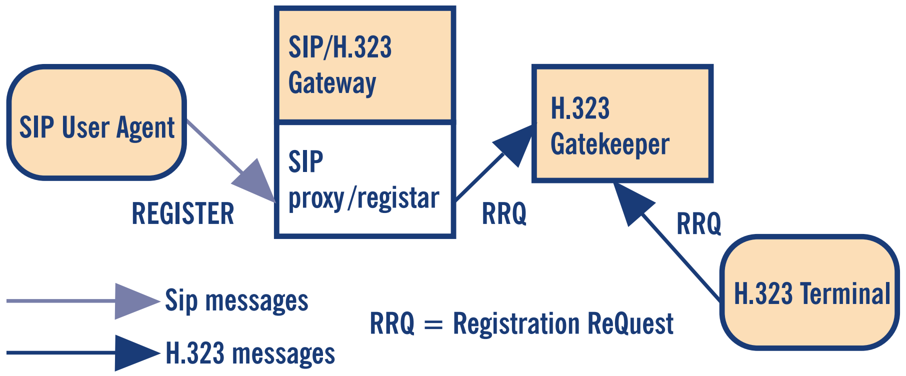


Figure 5.12 A SIP/H.323 Gateway containing a SIP Proxy and registrar

2. The SIP/H.323 Gateway contains an H.323 Gatekeeper (Figure 5.13). 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;

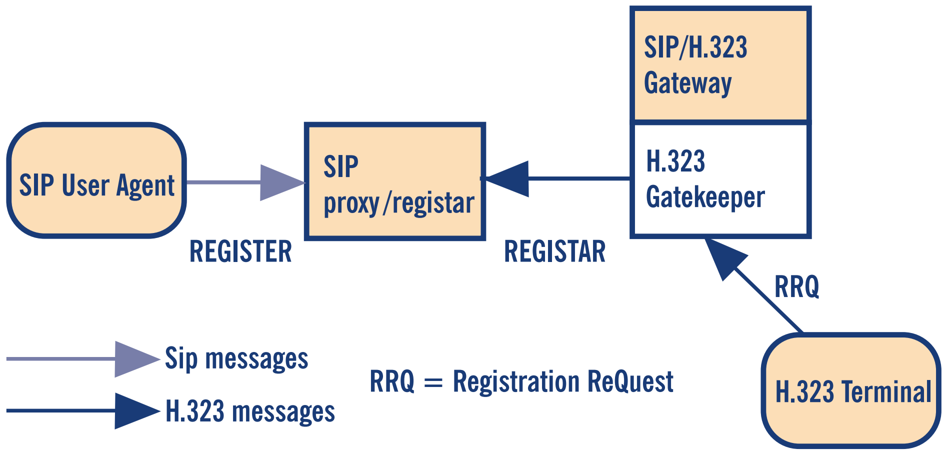


Figure 5.13 A SIP/H.323 Gateway containing an H.323 Gatekeeper

3. The SIP/H.323 Gateway is independent of a proxy or a gatekeeper (Figure 5.14). User registration is done independently in the SIP and H.323 networks.

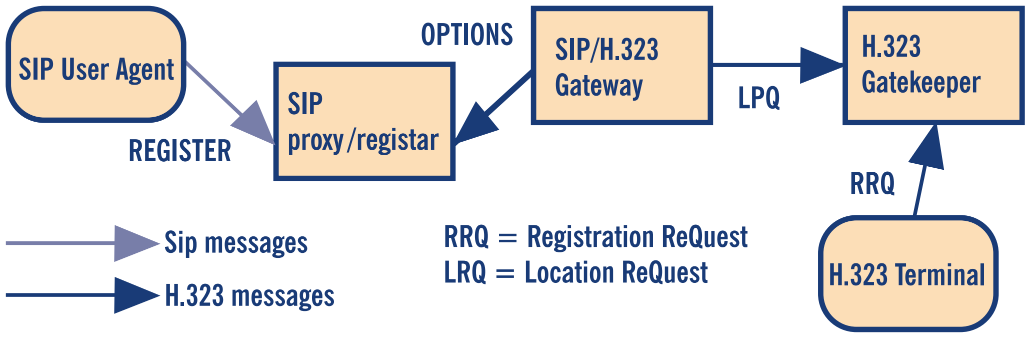


Figure 5.14 SIP/H.323 Gateway Independent

-/ 5.1.4.2 Call from SIP to H.323

The SIP/H.323 Gateway operations are described only when it acts as 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 and 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 originating 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 turn, 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

Similar to the previous case, calls from the H.323 domain to the SIP one only require an active intervention of the SIP/H.323 Gateway when the originating domain contains an H.323 Gatekeeper. Such a consideration applies since H.323 terminals appear to the SIP user agents as SIP URLs within the same domain (for further information on how H.323 addresses are translated to SIP URLs, please refer to 'Inter-working between SIP/SDP and H.323').

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, as 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, making an active intervention in the call.

-/ 5.1.4.4 Media switching and capability negotiation

A 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 inter-working 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 has to start a muted SIP call, and re-negotiate the capabilities later, unless the H.323 client supports the use of the FastStart procedure. If both of these conditions can not be ensured, the gateway must use the default codecs for both sides and, eventually, perform media transcoding.

-/ 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 an H.323 **Release Complete** message and vice-versa.

-/ 5.1.4.6 Configuration guidelines

In this section, basic configuration principles are given, abstracting them from specific software in order to give general guidelines and give information rapidly becoming 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 a need to configure:

- enabling/not enabling of built-in SIP Proxy/registrar and related configuration. Please refer to the 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 have to be sent, or if the SIP/H.323 Gateway is independent of proxy or gatekeeper);
- enabling/not enabling of a built-in H.323 Gatekeeper and related configuration. Please refer to section on setting up H.323 services for information on how to configure an 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 centralise the accounting on gatekeepers, which are able to maintain all the call information (if configured in call signalling 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 ones, 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 describes 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 the H.450 series, which establishes the signalling between endpoints necessary to implement the telephone-like services. Although some of these services have the same functionalities as the equivalent ones developed for the circuit-switched networks, it is relevant to note that the paradigm of 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 signalling. 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 services such as call transfer, call forwarding, call waiting, and so on. Since a peer-to-peer model is used by the supplementary services of H.323, the payload and the signalling 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, it can be deduced 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 characterised by an initial cost for the software implementing the service for unlimited use. This last aspect implies that new services can be distributed to VoIP users in the same way as any other software is sold in the market today.

This scenario raises problems related to the subscription-control of these supplementary services. To this aim, the signalling necessary for these services should be routed through the gatekeeper (or other proxy elements) containing a service database description that permits charging for the use of supplementary services or their blocking when they are not subscribed to by the customer.

Another issue related to the peer-to-peer approach used in the H.323 supplementary services is service incompatibility. In H.323, the clients exchange their capabilities and hence, they are only able to 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 signalling translation necessary to guarantee the services. Moreover, the gateway can be used by the provider to charge for 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 Information 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, attention will be focused mainly on the signalling 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 signalling necessary to use the service is 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 5.15 reports the signalling 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.

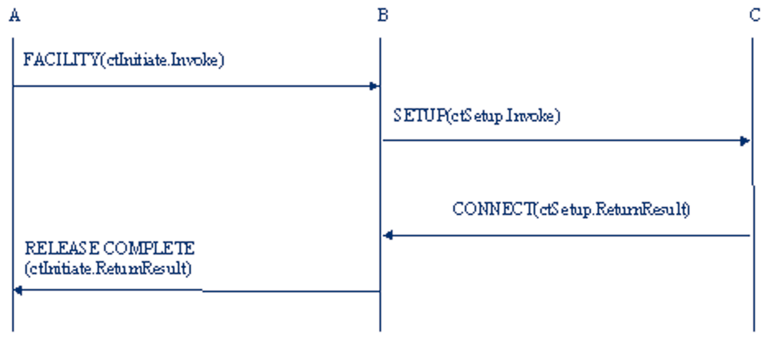
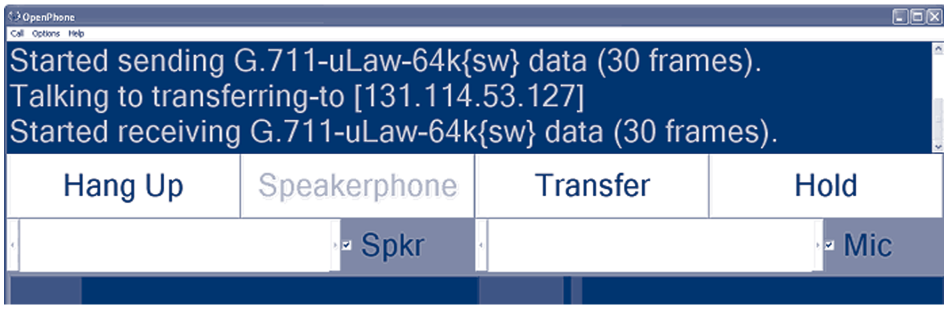

A
B
C
FACILITY(ctInitiate.Invoke)
SETUP(ctSetup.Invoke)
CONNECT(ctSetup.ReturnResult)
RELEASE COMPLETE
(ctInitiate.ReturnResult)

Figure 5.15 Messages exchanged to implement the CT-SS without gatekeeper

The example refers to the case where the gatekeeper is absent or the direct signalling model is adopted. In this case, only the software able to process and to generate the APDUs is necessary to implement the service. An example is given in Figure 5.16 where we report an Ohphone (an openh323 application) interface modified with, the addition, of a Call Transfer Supplementary services button. In this case, simply pressing the **Transfer** button makes the application of an open dialogue screen, allowing the insertion of the H.323 ID of the called party and generating the necessary APDUs.



```
Started sending G.711-uLaw-64k{sw} data (30 frames).
Talking to transferring-to [131.114.53.127]
Started receiving G.711-uLaw-64k{sw} data (30 frames).
```

Figure 5.16 An 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 either transparently transports or performs the operations necessary to offer the service. In particular, the gatekeeper can provide 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** signalling 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 signalling on behalf of transferred-to user) that it has been transferred. Furthermore, the gatekeeper should instruct the endpoint (or endpoints), that it manages the signalling 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 resulting APDU to release the call signalling 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 signalling procedure used, Figure 5.17 illustrates the messages exchanged for the CT-SS when the gatekeeper manages the signalling on behalf of the transferred user, i.e., B user.

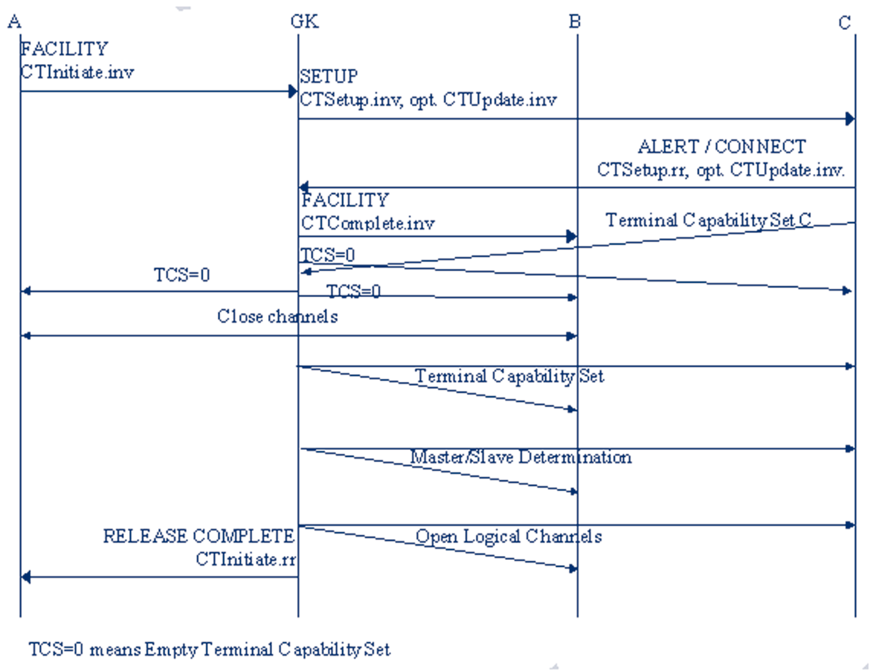


Figure 5.17 Messages exchange for gatekeeper-managed CT-SS

After the reception and the processing of the **FACILITY** message, indicating that user A wants to 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 signalling and proceeds with normal open logical channel signalling 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 signalling channel of the primary calls, i.e., between A and B.

-/ 5.2.1.2 Call Diversion Supplementary Service

The signalling 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 the served user and receiving a call from an originating user to redirect it to another user, denoted as the diverted-to user. This operation leads to the connection of the originating user with the diverted-to user, and it is possible only when the served and originating users are not in a call. There are four kinds of call diversion service. Supplementary Service Call Forwarding Unconditional (SS-CFU) permits a served user, independent of 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 of 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 the provision of the information related to the activated service, and it is performed on the activation procedure.

Indeed, to activate these services, the served user supplies 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, Call Deflection (SS-CD)is also defined, 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 the 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, acting as a 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 signalling address, rather than the call signalling 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 a call arrives, destined for user B (being the served user), the gatekeeper starts a local **no-response** timer and continues the 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. 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 Service

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 utilising 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 utilising 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 the responsibility of implementations to introduce new services on top of well-defined protocol specification. There are only a few well-defined protocol elements such as REFER method (RFC 3515). Such elements can be used for a variety of services without having to undergo the burdensome standardisation process again and again. With the particular REFER example, the REFER method may be used for several variations of call transfer, the 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 or SIP-CGI (RFC 3050).

To show the proper use of protocol elements for building frequently-used services, the IETF issued an informational document, 'Session Initiation Protocol Service Examples'. The document presents call flows for many services, including on-hold, transfers and conferencing. Another essential document is 'A Call Control and Multi-party Usage Framework for the Session Initiation Protocol (SIP)' 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 document related to implementation of telephony services is 'Third-party Call Control', 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 what 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 the user that he is on hold. (It may be silent, display the status using user interface, play local music, etc.) There are subtle differences in how **on-hold** is signalled in earlier and current SIP versions. RFC 2543 used a dummy IP address 0.0.0.0 to indicate an **on-hold** condition whereas RFC 3261 uses SDP send only attribute. Call-flow on Figure 5.18 presents **on-hold** signalling implemented according to RFC 2543. The key message is **re-INVITE**, number 3, which puts the other party on hold. The SDP payload of numbered SIP messages is shown in detail (SIP headers have been removed because they are not important in this case).

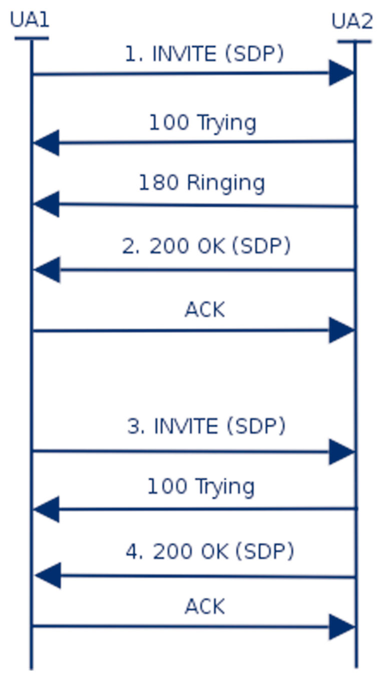


Figure 5.18 On-hold call flow

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 utilise **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 transferring agent first initiates a short conversation with the transfer target before connecting it to the transferred 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** by which the called party suggests that the calling party establish a conversation with the transfer target. The calling party does so by sending the **INVITE**. The called party exits the original conversation by sending **BYE** without awaiting the completion of the call transfer.

The following **REFER** message (red in the call flow) asks the calling party 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 called party).

```
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<
```

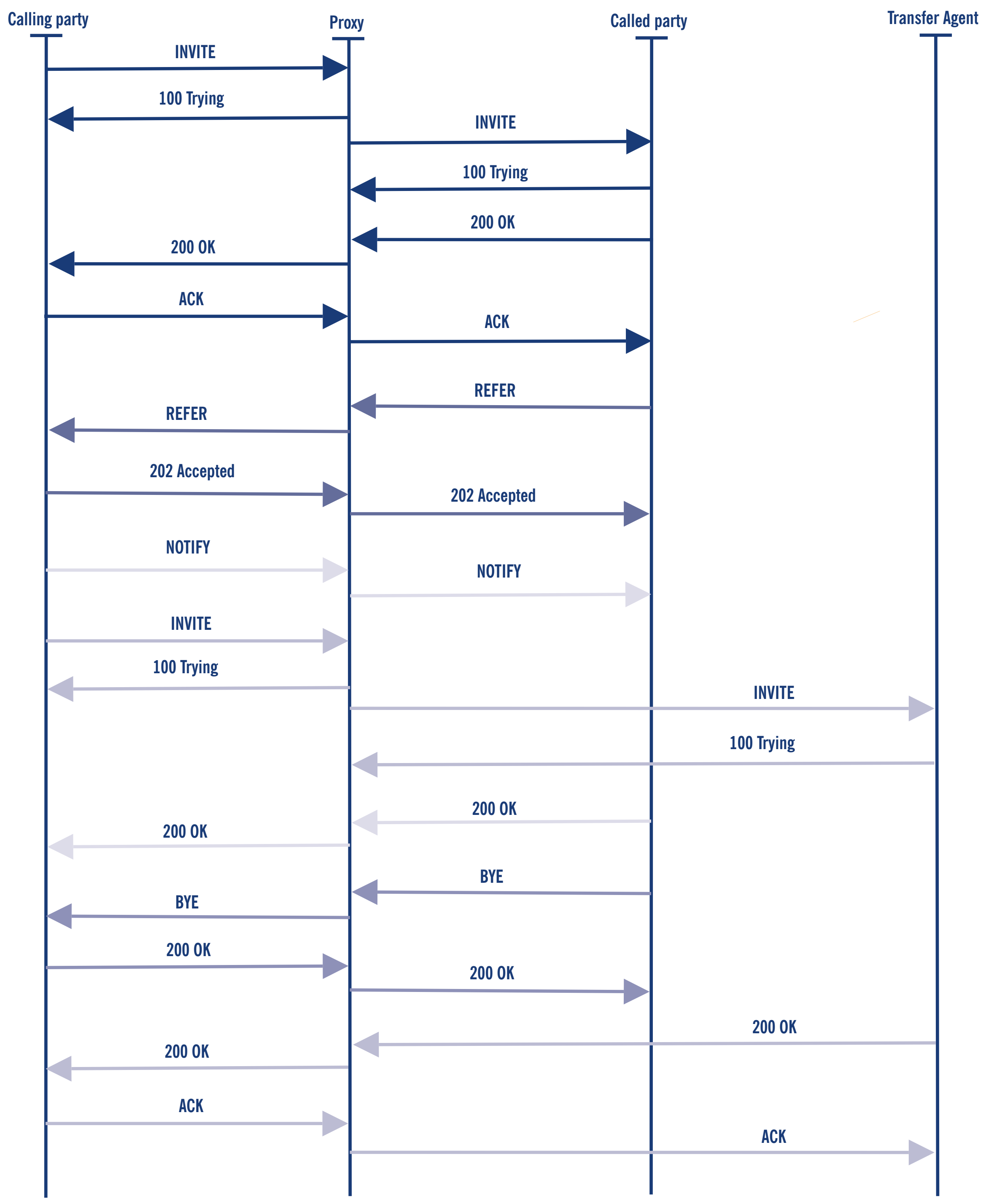


Figure 5.19 Call transfer call flow

The following **INVITE** message establishes a call from the calling party 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 calling party and called party.

```
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

A 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 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 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 ring 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, **lserctl**, to manipulate contacts of a user. Typically, this job is done through a 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.

First, the current status of registered contacts is inspected. 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 the previously-registered SIP phone and the 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

Called parties frequently wish to redirect incoming calls to an alternative destination if the primary destination fails to answer. The reasons for failure are multi-fold. They may include a busy called party, disconnected called party'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.

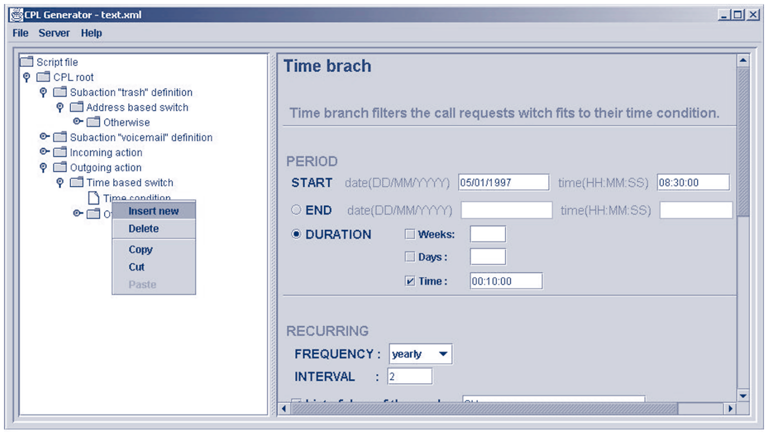
The following configuration fragment demonstrates how set up such a feature using SER:

```
# 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 standardised way of call handling allows easier service portability. The IETF standard for describing called party's preferences is called Call Processing Language, CPL. In short, 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 supports the 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.



Time brach

Time branch filters the call requests witch fits to their time condition.

PERIOD
START date(DD/MM/YYYY) 05/01/1997 time(HH:MM:SS) 08:30:00
END date(DD/MM/YYYY) time(HH:MM:SS)
[✓ DURATION] Weeks: Days: [✓ Time:] 00:10:00

RECURRING
FREQUENCY : yearly
INTERVAL : 2

Figure 5.20 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 dialled 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 from joining the conference. Some MCUs can initiate the set up of the conference itself by dialling all the parties. It needs to know everyone's 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 conversing 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 his 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 cannot approach the MCU directly over IP. No matter whether H.323 or SIP are used for 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.

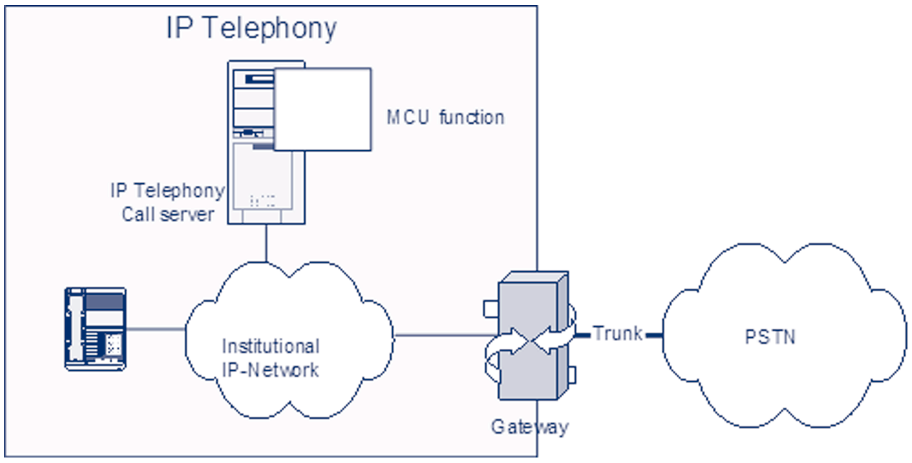


Figure 5.21. MCU function in gatekeepers

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, enabling users with different codecs to join the same conference. If video is also distributed by an MCU, the video streams cannot be mixed. 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. One participant is chosen and both audio and video are locked on the presenter; the rest of the audience can only listen. Some MCUs offer a Continuous Presence mode, in which the 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.

6 Setting up Value-added Services |-

This chapter introduces 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, E-mail 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 to 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 set up 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 of 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 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 provides a library in C and template code for an external application, but it is not fully working code and requires significant resources to be applied before a demo application can be built. Also commercial solutions from Cisco partners exist which are based on this same API (see GK API Guide Version 4.2).

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 information.

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:** 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:** this interface allows the definition and management of groups of endpoints (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: 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

This chapter provides an overview of some added-value services implemented using a Web interface. Serweb, the Web interface for the 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. A Web interface is often used for the provisioning of SIP products and for the implementation of advanced services. Web browsers are available in the vast majority of existing operating systems.

6.2.1 Click-to-dial

Click-to-dial is a method of establishing a call between participants using a Web interface. It greatly simplifies dialling, in that calling parties do not have to dial lengthy addresses and they keep their phonebooks separately from SIP phones. In its simplest form, a user has a Webpage where he can enter the SIP addresses of two users, and the SIP user agents of those two users get connected. We will focus on **REFER**-based click-to-dial.

A **REFER**-based click-to-dial scenario is based on the paradigm of intelligent end-devices and dumb network. One of the involved SIP user agents is asked to connect to the other and report to the server when it is done.

The drawback of this approach is that one of the involved SIP user agents must support the **REFER** SIP method which has been standardised recently (see RFC3515). The 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 in Figure 6.1.

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 the IP address in SDP, because there is no remote phone (the message is sent by user agent within the SIP server which does not deal with media).

After that, the server sends a **REFER** method which will ask the phone to send **INVITE** somewhere else. The URI of the called party is passed to the phone in a **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 the 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 are connected. You can implement a list of missed call in exactly the same way and clicking on an entry in the list will connect your phone with that person. There are many others possible scenarios.

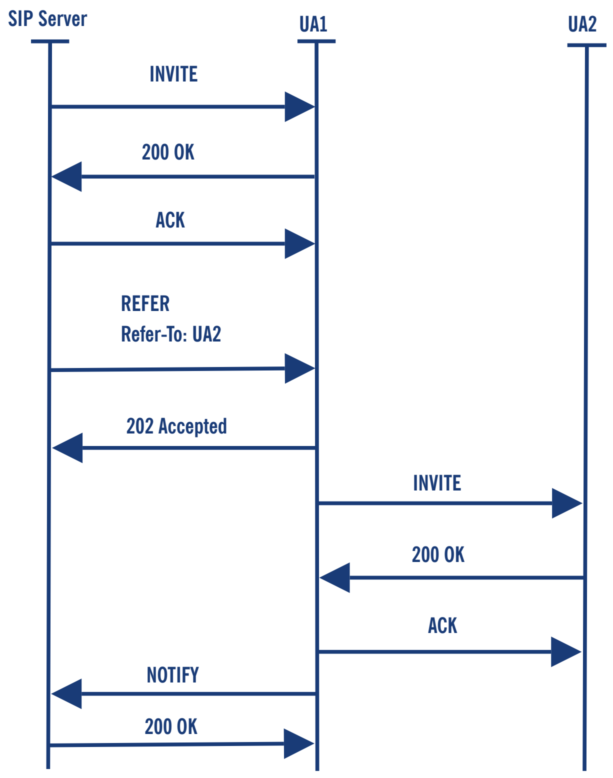


Figure 6.1 **REFER**-based click-to-dial

6.2.2 Presence

It is also possible to display the presence of SIP users in a Webpage (for example, in a phone-book). Displaying the on-line status of subscribers allows calling parties to determine availability and willingness to have a conversation conveniently. The status may be shown, for example, in the calling party's phonebook or on the called party's homepage. Linking on-line users to click-to-dial applications greatly integrates telephony with the Web and introduces 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`, create a new account, and insert some entries into your phone-book to see how it works.

6.2.3 Missed calls

'Missed calls' is a feature that allows a user 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 in common with accounting. Servers doing accounting log some information when a call is successfully established and torn down. When a calling party gets a negative final response to his call, or does not receive any reply at all (timeout), then the server also records this event with a flag, indicating 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 front-end for a SIP Express Router (see Section 4.6.2 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 a SIP Express Router and serweb talk together using a 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

The entire 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/";
```

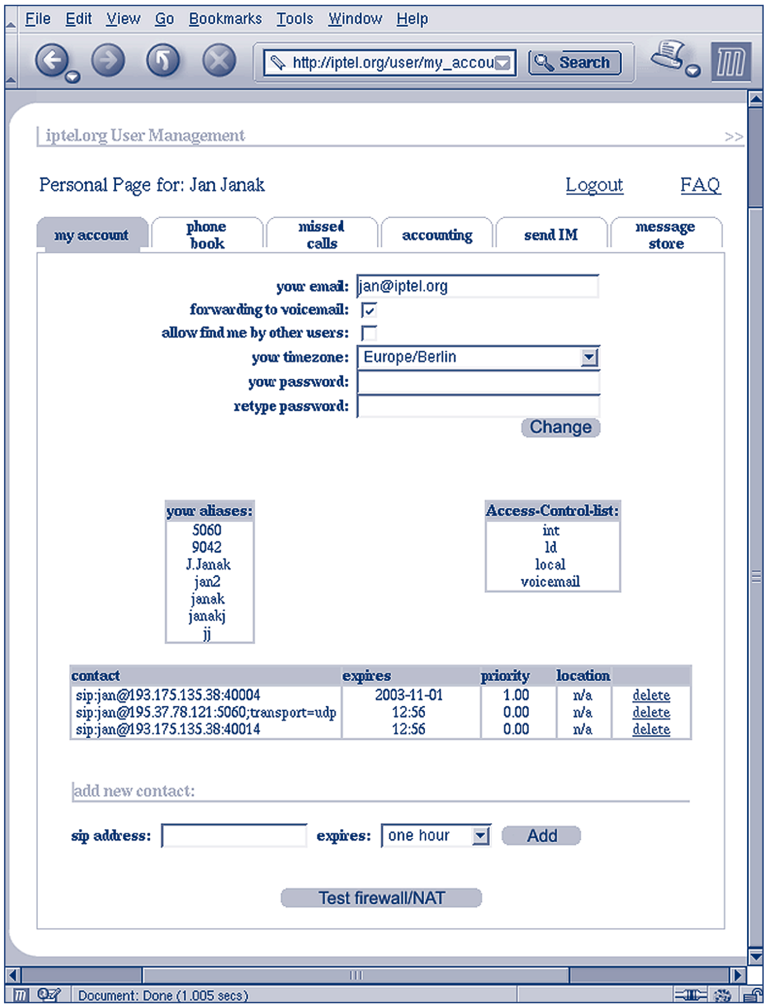
It is necessary to create some aliases in the configuration file of Apache Web server:

```
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.


File Edit View Go Bookmarks Tools Window Help
http://iptel.org/user/my_accou
Search
iptel.org User Management
Logout
FAQ
my account
phone book
missed calls
accounting
send IM
message store
Change
Add
Test firewall/NAT
Document: Done (1.005 secs)

Personal Page for: Jan Janak

your email:	jan@iptel.org
forwarding to voicemail:	✓
allow find me by other users:	☐
your timezone:	Europe/Berlin
your password:	
retype password:	

your aliases:
5060
9042
J.Janak
jan2
janak
janakj
jj

Access-Control-list:
int
ld
local
voicemail

contact	expires	priority	location	
sip:jan@193.175.135.38:40004	2003-11-01	1.00	n/a	delete
sip:jan@195.37.78.121:5060;transport=udp	12:56	0.00	n/a	delete
sip:jan@193.175.135.38:40014	12:56	0.00	n/a	delete

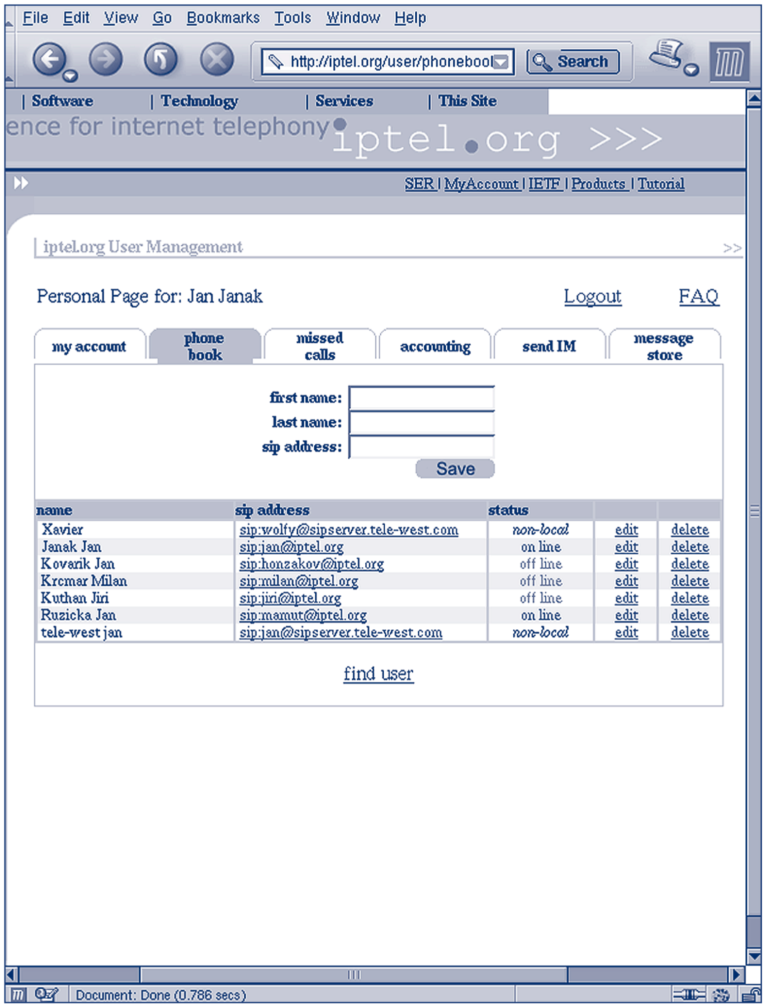
add new contact:

sip address: expires: one hour

Figure 6.2 Serweb – My Account

The **My Account** tab (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 the PSTN.

Users can also create their own phone book (see Figure 6.3). In the phonebook, you can see the presence status of each user. If the user is currently registered, then you will see **online** in the status column. If he is not registered then you will see **offline** and if the user does not belong to administrative domain of the server, then you will see **non-local**.



Personal Page for: Jan Janak

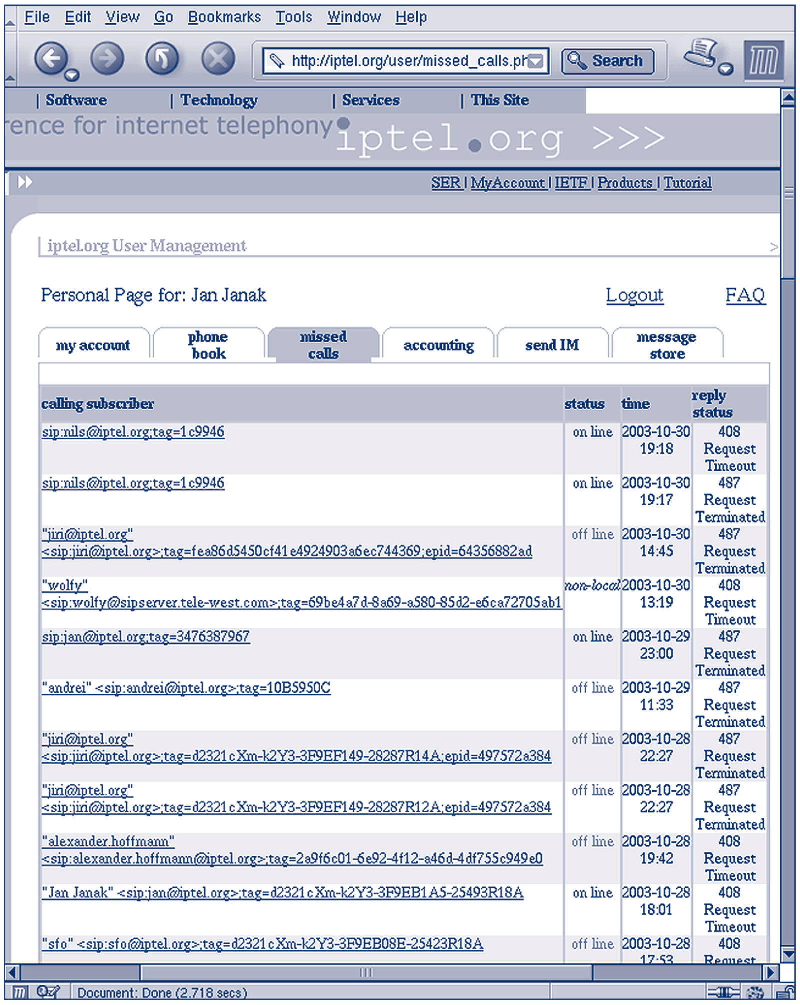
first name:
last name:
sip address:

name	sip address	status		
Xavier	sip:wolfy@sipserver.tele-west.com	*non-local*	edit	delete
Janak Jan	sip:jan@iptel.org	on line	edit	delete
Kovarik Jan	sip:honzakov@iptel.org	off line	edit	delete
Krcmar Milan	sip:milan@iptel.org	off line	edit	delete
Kuthan Jiri	sip:jiri@iptel.org	off line	edit	delete
Ruzicka Jan	sip:mamut@iptel.org	on line	edit	delete
tele-west jan	sip:jan@sipserver.tele-west.com	*non-local*	edit	delete

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.

The **missed calls plane** (see Figure 6.4), allows users' to see their missed calls. Again, clicking on an entry will connect you with that **user** and **status** describes presence of the user as described in the previous section.


File Edit View Go Bookmarks Tools Window Help
http://iptel.org/user/missed_calls.ph
Search
| Software | Technology | Services | This Site
ence for internet telephony
iptel.org >>>
SER | MyAccount | IETF | Products | Tutorial
iptel.org User Management
my account
phone book
missed calls
accounting
send IM
message store
Document: Done (2.718 secs)

Personal Page for: Jan Janak   Logout   FAQ

calling subscriber	status	time	reply status
sip:nils@iptel.org;tag=1c9946	on line	2003-10-30 19:18	408 Request Timeout
sip:nils@iptel.org;tag=1c9946	on line	2003-10-30 19:17	487 Request Terminated
"jiri@iptel.org" <sip:jiri@iptel.org>;tag=fea86d5450cf41e4924903a6ec744369;epid=64356882ad	off line	2003-10-30 14:45	487 Request Terminated
"wolfy" <sip:wolfy@sipserver.tele-west.com>;tag=69be4a7d-8a69-a580-85d2-e6ca72705ab1	*non-local*	2003-10-30 13:19	408 Request Timeout
sip:jan@iptel.org;tag=3476387967	on line	2003-10-29 23:00	487 Request Terminated
"andrei" <sip:andrei@iptel.org>;tag=10B5950C	off line	2003-10-29 11:33	487 Request Terminated
"jiri@iptel.org" <sip:jiri@iptel.org>;tag=d2321cXm-k2Y3-3F9EF149-28287R14A;epid=497572a384	off line	2003-10-28 22:27	487 Request Terminated
"jiri@iptel.org" <sip:jiri@iptel.org>;tag=d2321cXm-k2Y3-3F9EF149-28287R12A;epid=497572a384	off line	2003-10-28 22:27	487 Request Terminated
"alexander.hoffmann" <sip:alexander.hoffmann@iptel.org>;tag=2a9f6c01-6e92-4f12-a46d-4df755c949e0	off line	2003-10-28 19:42	408 Request Timeout
"Jan Janak" <sip:jan@iptel.org>;tag=d2321cXm-k2Y3-3F9EB1A5-25493R18A	on line	2003-10-28 18:01	408 Request Timeout
"sfo" <sip:sfo@iptel.org>;tag=d2321cXm-k2Y3-3F9EB08E-25423R18A	off line	2003-10-28 17:53	408 Request

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 comes online. In the plane **message store** (see Figure 6.5), you can see all messages that are stored for you.

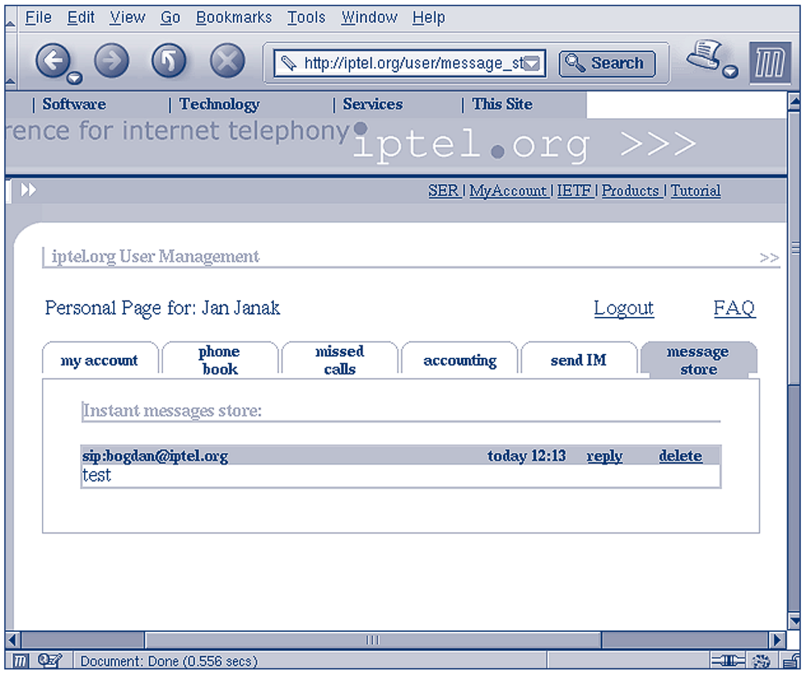
6.2.5 SIP Express Router message store

Message store has been implemented as a separate module for a SIP Express Router. To use the module, you will need to load the module:

```
loadmodule "/usr/local/lib/ser/modules/msilo.so"
```

Configure the address of the server (it will be used when sending stored messages) and the URL of the database:

```
modparam("msilo", "registrar", "sip:registrar@<your_domain>")
modparam("msilo", "db_url", "sql://ser:passwd@dbhost/ser")
```


File Edit View Go Bookmarks Tools Window Help
http://iptel.org/user/message_st
Search
| Software | Technology | Services | This Site
rence for internet telephony
iptel.org >>>
SER | MyAccount | IETF | Products | Tutorial
iptel.org User Management
Logout
FAQ
my account
phone book
missed calls
accounting
send IM
message store
Document: Done (0.556 secs)

Personal Page for: Jan Janak

Instant messages store:

sip:bogdan@iptel.org	today 12:13	reply	delete
test			

Figure 6.5 Serweb - Message store

Then, when the server receives a **MESSAGE** request and it cannot 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 the **lookup** of recipient's location fails (the recipient is not registered), a new transaction is created (needed for **msilo** module). Save the message using the **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 the 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 if so, sends them.

6.3 Voicemail

Another Internet application which lends itself for integration with telephony is e-mail. A traditional PSTN application, which can be replaced with VoIP and e-mail, is voicemail.

Traditional PSTN voicemail systems feature fairly inconvenient user interface for message retrieval: IVR (Interactive Voice Responder). Calling parties 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 the called to party by e-mail. 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-to-e-mail scenario. An open-source voicemail-to-e-mail application, SEMS, is available from `iptel.org`.

SEMS stands for SIP Express Media Server. It is an application framework that offers easily-built applications dealing 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 exist including OpenAM, which is available from the OpenH323 Website 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 completely integrated product. They do not work without bugsand they can easily be customised for small environment scenarios.

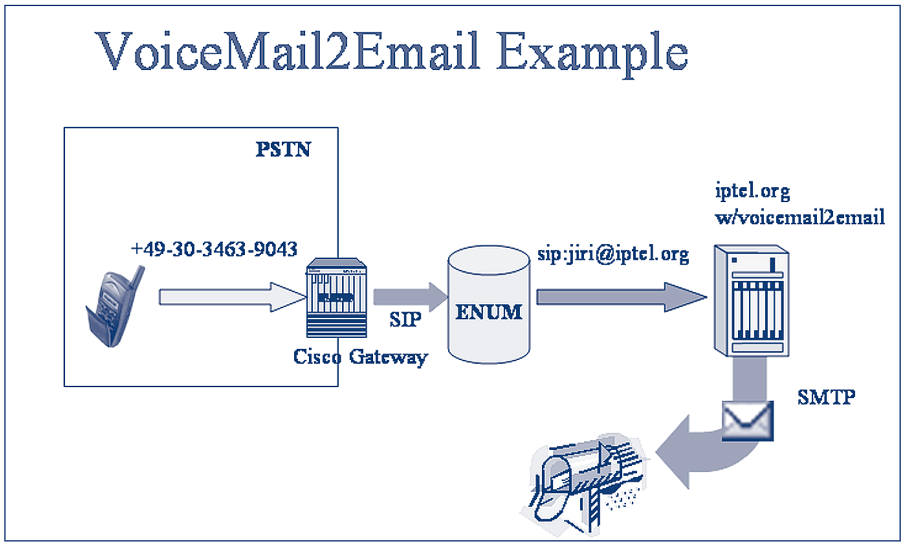


Figure 6.6 Voicemail

7 Integration of Global Telephony -}

The previous chapters described how to set up an IP Telephony site and how to use PSTN gateways to call external targets. With growing support and usage of IP Telephony, it becomes more and more interesting to interconnect IP Telephony sites and use the IP network to transport calls. Since dialling IP addresses is obviously not an option, inter-domain communication introduces the problem of call routing which can be dealt with in various ways. This chapter lists techniques and solutions for inter-domain call routing.

-} 7.1 Technology

This section starts with listing possible mechanisms and protocols to provide inter-domain address resolution.

-} 7.1.1 H.323 LRQ

With the H.323 protocol, even for intra-domain routing, there is a need to localise the terminals. Under normal operations, the user of an H.323 appliance configures the H.323 alias identifier and, eventually, the E.164 address number as they are assigned by the service administrator.

Later on, when a call is started, there is a need for the terminal to identify the logical channel of the called party (IP address and destination port) where to send the signalling messages.

If a gatekeeper is not present, call routing is possible using the IP address (that has to be known to the calling party) and a standard destination port. On the other hand, if a gatekeeper is present, both intra-domain and inter-domain routing can be performed using either the alias identifier or the E.164 address. If the called party is registered within the same domain as the calling party, alias mapping is performed internally by the gatekeeper itself, which replies to the calling party with the call signalling address (IP address and destination port) of the called party.

If the called party is registered within any other domain, the gatekeeper has to perform another step before replying to the calling party with the **Admission ConFirm** (ACF) message containing the call signalling address to be contacted. In order to localise the terminal within the domain, gatekeepers use the **Location ReQuest** (LRQ). The LRQ message is transmitted by the gatekeeper to other well-known neighbour gatekeepers (if any) or to a multicast address. When the LRQ message arrives to the gatekeeper where the terminal to be localised is registered, such a gatekeeper answers with **Location ConFirm** (LCF) message containing the information on the logical channel of the called party to be used for the signalling messages. At this point, the gatekeeper knows the call signalling address of the called party and, depending on the call model in use, it proceeds with the normal operation (see Chapter 2).

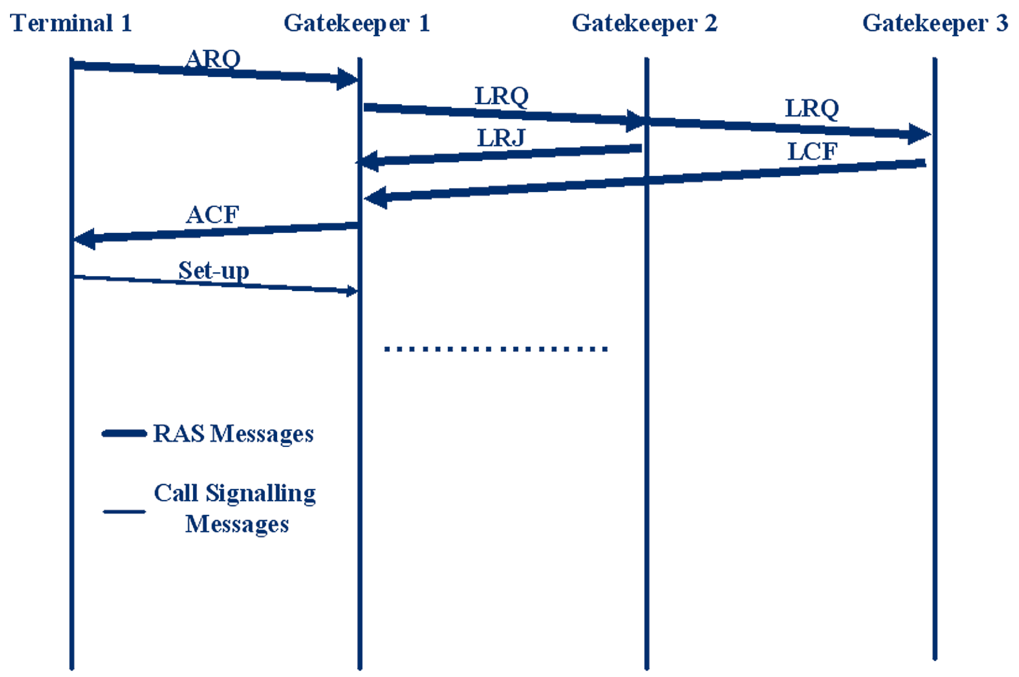


Figure 7.1 Location request mechanism

In the case that a gatekeeper reached by the LRQ message does not know the answer to such a location request, there are different possible behaviours depending on the channel which is being used by the LRQ message. If the LRQ message was sent on the RAS channel, then the gatekeeper replies with a **Location ReJect** (LRJ), whereas no action is taken if it was sent on the multicast channel. This simple mechanism is depicted in Figure 7.1, where the channel being used by the LRQ messages is the RAS one.

When handling calls directed to terminals on the traditional PSTN, the zone gatekeeper is in charge of registering the zone gateways. In such a case, the gatekeeper will take care of replying to the LRQ messages with a LCF message containing the call signal address of the gateway that is supposed to be the one routing the call to the PSTN network.

-} 7.1.2 H.225.0 Annex G

Another H.323-specific mechanism is defined by H.225.0 Annex G. Unlike the H.323 LRQ mechanism that simply provides a way to translate an address to an IP address and port number, Annex G allows the coupling of further information (e.g., pricing information) to an IP address and to use IP address prefixes for dialling (instead of just allowing complete addresses to be dialled). Furthermore, H.225.0 Annex G is a protocol used for communication between administrative domains, meaning a logical collection of gatekeeper zones (e.g., all gatekeepers used in a university would span the administrative domain of the university). The entities that provide H.225.0 Annex G services are called border elements.

The smallest information unit defined by H.225.0 Annex G is the pattern, which might be either a specific address, an address containing a wildcard or a range of telephone numbers. H.225.0

Annex G uses the **AliasAddress** structure to refer to addresses. Such usage is particularly useful when storing specific or wildcard addresses; thus it is able to carry any kind of address information used by H.323 (see Section 2.2.1.3.1).

Along with the patterns, H.225.0 Annex G stores some routing information containing additional data about pricing, necessary access requests, contact information (which IP/Port to contact and what kind of QoS is provided) and other protocol-relevant data.

Patterns and routing information are grouped in a so-called **address template**, along with the **time to live** to indicate how long the template is valid. The address template also contains information regarding the signalling protocols that may be used. Currently, signalling protocols include only the H.3xx protocol family. Templates are grouped together by an identifier, known as a descriptor. An administrative domain advertises templates by advertising descriptors to indicate the type of calls it can resolve.

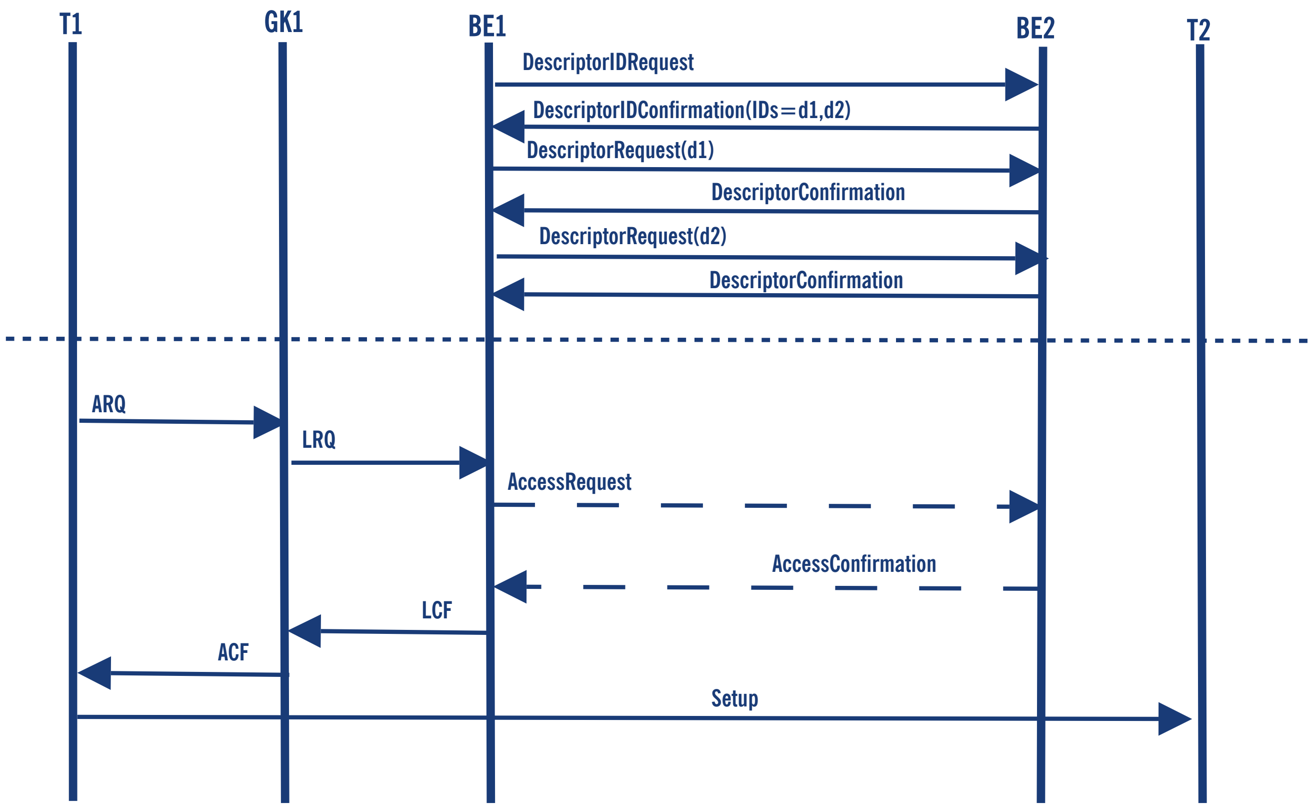


Figure 7.2 Use of H.225.0 Annex G

On the protocol side, H.225.0 Annex G defines unidirectional relationships between border elements. Such a relationship is established by sending a **ServiceRequest** to a well-known port (2099) of a configured peer. Upon receipt of a positive reply (**ServiceConfirmation**) the initiating peer sends a **DescriptorIDRequest** to the other border element, which replies by returning the IDs of all known descriptors (see Figure 7.2). The first border element now requests each descriptor by sending a **DescriptorRequest**. After all descriptors are sent, the initiating border element knows which addresses can be reached via the other border element.

When an endpoint, T1, tries to call another endpoint, T2, outside of T1's administrative domain, it sends the ARQ message to the gatekeeper GK1 as usual (see Figure 7.2). GK1 recognises the destination address in another administrative domain and asks its border element, BE1, by sending an LRQ. The border element knows by previous descriptor exchange with BE2 how to contact the given address. If BE2 requested that its authorisation is mandatory for all calls to that address template, BE1 sends an **AccessRequest** to BE2 before replying with a LCF (or LRJ) message. When GK1 receives a LCF message, the normal H.323 protocol flows apply.

The access requests from BE1 to BE2 might be enforced. The gatekeeper of the called endpoint in BE2's administrative domain might send a **ValidationRequest** to BE2, to check if the incoming call has been accepted by the border element.

-} 7.1.3 Telephony Routing Over IP (TRIP)

RFC 3219 'Telephony Routing over IP' (TRIP) defines the TRIP protocol that can be used to advertise reachability information for telephone numbers (e.g., E.164) between different administrative domains. The TRIP protocol design is similar to the Border Gateway Protocol (BGP) and uses a binary packet-encoding.

-} 7.1.3.1 Structure

TRIP defines communication between and within IP Telephony Administrative Domains (ITAD). Inter-ITAD communication uses peer relationships, which are considered to be setup between two sites upon a trust relationship.

ITADs are identified by a globally unique number. The Internet Assigned Numbers Authority (IANA) registers ITAD numbers to ensure that a number is globally unique. Taking the amount of registered ITADs as an indicator of the interest in this protocol, and since exactly one registered ITAD is listed on the IANA Website, TRIP does not seem to be widespread.

An ITAD consists of one or more location servers, of which at least one has a peer relationship to a location server of another ITAD. While inter-ITAD communication is routed on a hop-by-hop basis, the intra-ITAD communication is done by flooding all internal peers.

-} 7.1.3.2 Addressing

TRIP can be used for SIP and H.323 and allows disclosure of which signalling protocol can be used to reach an address. For instance, you can advertise the 'reachability' of a phone number +49.421.2182972 via H.323 on host A, while the same address is reachable via SIP on another, host B. Since TRIP was intended to provide a mechanism that allows selection of egress gateways, the protocol is limited to phone numbers. It is not possible to advertise URIs and names (see Section 2.1.5) which makes TRIP unsuitable for all kinds of inter-domain call routing.

-} 7.1.3.3 Protocol

Initially, a TRIP Location Server (LS) knows just its local addresses.

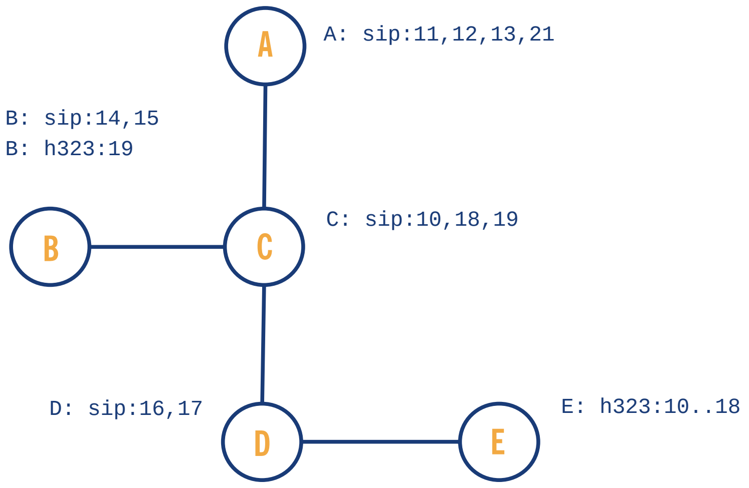


Figure 7.3 TRIP: Location Servers and their initial data.

After establishing connections with its peer LSs, each TRIP node advertises all the routes it already has knowledge of.

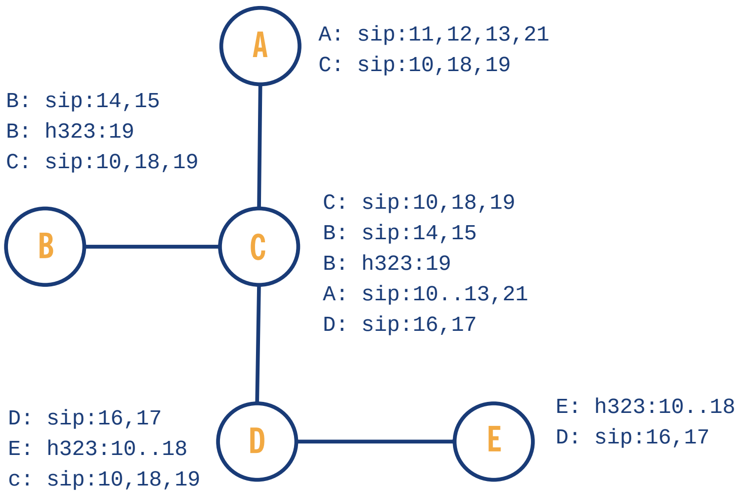


Figure 7.4 TRIP: LS tells peers their initial data

When an LS receives data from a peer LS, it stores it internally. This data is distributed the next time the LS sends an **UPDATE** message to its peers, but not before a minimum delay has elapsed (see Figure 7.5). Each node can decide whether to advertise itself (or better: the associated VoIP server) as the next-hop server of the new learned numbers (Node D) or to pass the information of the original contact (Node C).

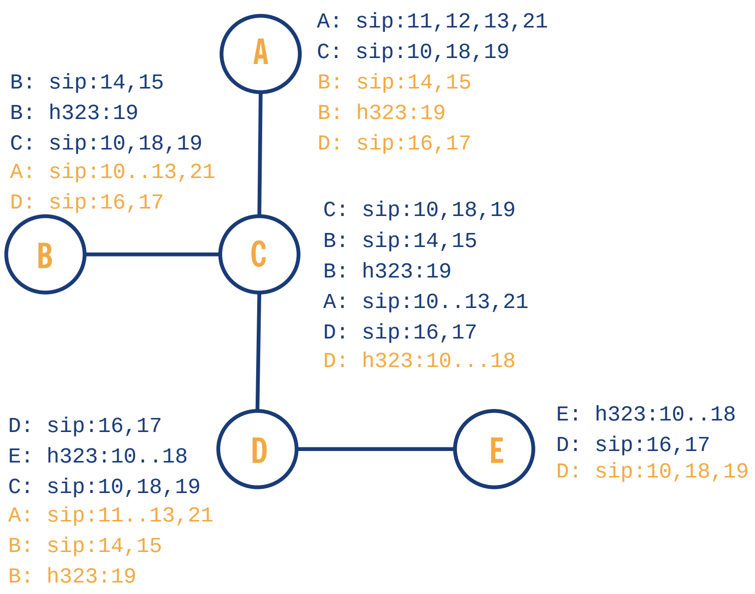


Figure 7.5 TRIP: Advertising gathered knowledge

When a LS collects a continuous range of telephone numbers (e.g., from 770 to 779), it can aggregate this information to a common prefix. In the example given in Figure 7.6, Node D knows how to reach the numbers from sip:10 to sip:19. Since E is not on the path to one of these numbers, it withdraws the routes sent previously and adds a new route containing the prefix '1'.

Node C could have done the same for h323:10 to h323:19 when talking to A, but since C was configured not to put itself in the chain of next-hop servers (or simply because the feature is not supported), it does not aggregate that information.

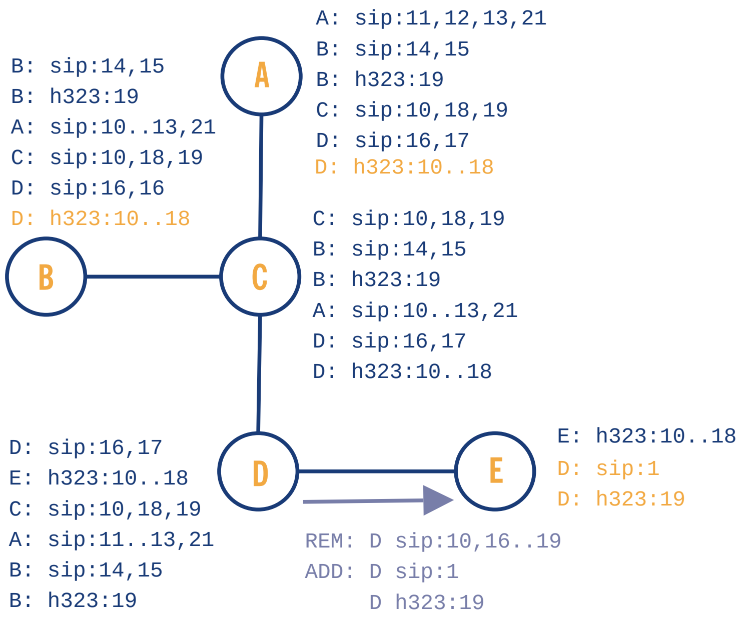


Figure 7.6 TRIP: Route aggregation

-} 7.1.4 SRV-Records

In February 2000, RFC 2782 was approved, describing a DNS RR for specifying the location of services (DNS SRV).Before then, there was a need (and sometimes there still is) for users to know the exact address of a server providing a specific service for a specific domain (i.e., GK or SIP Proxy). The first attempt at point-to-service-specific servers was MX record, used for mail servers. But, unfortunately, there was no feasible means of expressing a common service.

The SRV records allow administrators to easily manage the address propagation of servers providing specific services and could provide significant service-based routing advice. Several servers could be used for a single domain, with defined preference- and load-balancing. Users can easily ask for a desired service in a specific domain and obtain a name or list of server names that provide the service they requested.

The SRV RR is a structured collection of fields, each one with a name and a specific meaning:

- **Service** - Describes the service by its symbolic name (i.e., LDAP or SIP) registered by IANA Assigned numbers or locally defined. An underscore is prepended to avoid conflicts with common names that may occur in DNS. Service name is case insensitive;
- **Proto** - Describes the used protocol by its symbolic name. An underscore is prepended to avoid conflicts with common names that may occur in DNS. Service name is case insensitive. The most common values for this filed are _tcp and _udp at present;
- **Name** - Describes the domain name for which the service is specified;
- **TTL** - It is the time to live of the record. It specifies how many seconds the record will be valid in the cache of a questioner;
- **Class** - IN class is right for SRV records. The meanings of TTL and Class are described in detail in RFC 1035;
- **Priority** - Describes priority of the target host in range between 0 and 65535. The records with lowest number should be tried first;
- **Weight** - Describes a server selection mechanism for records with the same priority. Zero should be used if no server selection should be done. Otherwise, the higher number gives the record proportionally higher probability of being selected;
- **Port** - Describes the service port on the target host (i.e., 5060 for SIP). These numbers are often the same as registered IANA assigned numbers, if the service is running in a standard port;
- **Target** - Describes the name of the target host. The address for the name could be provided by one or more address records (even A or AAAA). Use of an alias as the name is prohibited.

If the client wants to find a server for a desired service, it first tries **SRV lookup**. If the result contains one record, the address of the server is resolved and used. In a case where there is more than one record, they are ordered by preference and weight and tried out. If no record is returned, **A** or **AAAA** lookup should be performed.

```
$ORIGIN domain.org.
_ldap._tcp SRV 0 1 389 ldap1.domain.org
           SRV 0 3 389 ldap2.domain.org
           SRV 1 0 389 ldap-old.domain.org
*._tcp           SRV 0 0 0     .
*._ucp           SRV 0 0 0     .
```

This example shows an SRV LDAP service for the domain '`domain.org`'. A user asking for LDAP service, using the TCP protocol, obtains three records. During the first round of attempts the user should try to connect to `ldap1.domain.org` or `ldap2.domain.org`. Three quarters of attempts should go to `ldap1` and one quarter to `ldap2`. If neither of those two is available, the user should then try to connect to `ldap-old`. The last two records determine that no other service, using either TCP or UDP, is provided in the domain '`domain.org`'.

More detailed information about DNS RR can be found in RFC 2782.

The two most-used VoIP protocols are SIP and H.323. The SIP protocol usually uses service names such as **_sip** and **_sips** and protocol names such as **_tcp** and **_udp**. H.323, in its Annex O, describes the use of SRV RR to locate specific services. Service names and protocols are more distinguishable in H.323 than it is the case in SIP:
- h323ls - Location Service, entity supporting H.225.0 LRQ;
- h323rs - Registration Service, entity supporting H.225.0 RRQ;
- h323cs - Call Signalling, entity that performs H.225.0 call signalling;
- h323be - Border Element, entity that supports communication as defined in Annex G.

Along with protocol symbolic names such as TCP and UDP, sctp and h323mux are also used. The contents of the Annex O will hopefully be implemented in new versions of H.323 products together with Protocol Specification, Version 5.

As can be seen, SRV record helps to find service-specific servers for desired domains. In the case of VoIP, that means lowering the need to maintain the address of distant server's information at gatekeepers and proxies and to ease the configuration of the clients.

-} 7.1.5 ENUM

One of the main issues with IP Telephony today is seamless integration with the PSTN. As more than one signalling protocol is used, integration should include them all. In general, it could be useful to have one identifier (business card contact) for all available services. By service, we could understand phone call (PSTN, SIP, and H.323), e-mail, Web and many others. Such a unifying identifier was chosen to be the E.164 number defined by ITU-T standards. It is represented as a number up to fifteen digits with a leading +. These numbers are used in the PSTN and very often by H.323 systems. The next issue was to find a reasonable worldwide, deployed database that would hold and provide such translation information. The DNS seems to be the right way at the moment, as it is used in probably all server and client machines in the Internet.

The mechanism of the E.164 to URI DDDS (Dynamic Delegation Discovery System RFC 3401) translation and its applications (ENUM) is described by RFC 2916 bis draft (currently 07) and by the RFCs listed as a reference in that draft.

The NAPTR RR is a structured collection of fields used to store translation information. The most important fields are the following:

- **Name** – Represents an E.164 number encoded as a domain name. The conversion is done by the following algorithm:

  ```
  o All non-digit characters are removed.
  +420-123456789 is transformed to 42123456789
  o Dots are inserted between each digit
  42123456789 is transformed to 4.2.0.1.2.3.4.5.6.7.8.9
  o Order of digits is reversed
  4.2.0.1.2.3.4.5.6.7.8.9 is transformed to 9.8.7.6.5.4.3.2.1.0.2.4
  o To the end is appended string e164.arpa
  4.2.0.1.2.3.4.5.6.7.8.9 is transformed to
  4.2.0.1.2.3.4.5.6.7.8.9.e164.arpa
  ```

  The e164.arpa domain is used worldwide as the domain designed for the ENUM purpose only;

- **Order** - Specifies the order of NAPTR rules processing. The ordering is from lowest to highest, i.e., records with the same order value are processed according to a combination of Preference and Service;

- **Preference** - Equivalent to the Priority field in the SRV RR. The main difference between Order and Preference is that once a match is found, the client has to work with records within only one Order, but it can use records with different Preference values within the selected Order. The use of SRV records or a multiple address record is important for load balancing;

- **Flags** - These are strings consisting of characters (A-Z, 0-9) that control the way of rewriting and interpreting a record. Flags are defined by RFC 3404. S, A and U flags are used as terminal flags terminating the DDDS loop (RFC 3402) and determining what should be the next action. The U flag is used to define that the output of the rule is an URI. The A flag means that the output is the domain name and that it should be resolved by using address records (A, AAAA, A6). It is expected that the output of the rule with the S flag is the domain name for which one or more SRV records exists. The most-used flag seems to be the U flag. The use of the U flag does not deny **SRV lookup** at questioner for the domains returned in URI. These two mechanisms can cooperate at a very large scale;

- **Service** - Service parameters have the following form: 'E2U+service-type:subtype', where E2U is the mandatory and non-optional value determining E.164 to URI translation. The service-type and subtype (ENUM-service) define what the record can be used for. URI schemes, service-types and subtypes are not implicitly mapped one to another. This mapping could be done by specification of the ENUM-service. Most important for VoIP use, are SIP (draft-ietf-sipping-e164-04.txt) and H.323 (draft-ietf-enum-h323-01.txt) scheme specifications. The application decides what record to use, comparing its own capabilities and the user request with offered ENUM service types;

- **Regexp** - This is a string containing a regular expression that the questioner applies to the original string. Note that **regular expression** is a very powerful tool and therefore should be well-constructed and tested to avoid errors leading to partial 'unreachability' of the user. The easiest regular expression is '!^.*$!', which covers the whole original string;

- **Replacement** – This field is used when the regular expression is a simple replacement and its value is the domain name that will be queried next.

A NAPTR record for number +420123456798 could look like this:

```
$ORIGIN 9.8.7.6.5.4.3.2.1.0.2.4.e164.arpa
   IN NAPTR 10 100 "u" "E2U+sip"        "!^.*$!sip:smith@domain.org!"    .
   IN NAPTR 10 101 "u" "E2U+h323"       "!^.*$!sip:smith@domain.org!"    .
   IN NAPTR 10 102 "u" "E2U+msg:mailto" "!^.*$!mailto:smith@domain.org!" .
```

Domain 9.8.7.6.5.4.3.2.1.0.2.4.e164.arpa (user with E.164 number +420123456789) is contacted first by SIP, second by H.323 and third by e-mail.

A wildcard could be used to express prefixes. Use of wildcards is under discussion. The example could then look like this.

```
$ORIGIN 7.6.5.4.3.2.1.0.2.4.e164.arpa
*  IN NAPTR 10 100 "u" "E2U+sip"  "!^\+420(.*)$!sip:$\1@domain.org!" .
```

All numbers with `prefix +4201234567` will be translated into `sip:1234567[suffix]@domain.org` and `contacted by SIP`.

How will the whole process work? In the example the user dials `+420123456789` in his SIP client; a NAPTR query is performed and, according to the service provided by the SIP client, the URI `sip:smith@domain.org` is formed. Then the client tries to find the SIP server for the domain, `domain.org` via an SRV or `A` (`AAAA, A6`) query and connects to the obtained address.

Using ENUM along with SRV helps to provide flexible management of available services and provides a single contact point that could be used even from the PSTN. This is the major task of ENUM. Even ENUM has some problems. The main problem could be security. Especially if DNS is used as record storage, information could be easily retrieved from the database and used, for example, for spamming.

-}7.2 Call routing today

Since some of the protocols or mechanisms, especially those described in Section 7.1.5, are quite young, there is no widespread support in existing equipment. An institution that invested in VoIP equipment in 2002/2003 will probably support only few mechanisms. To describe how global call routing has been implemented so far, one needs to distinguish between H.323 and SIP, since they originate from different backgrounds and therefore have different approaches..

-} 7.2.1 Using SIP

SIP's IETF origin has lead to an early adoption of the usage of SRV-records to resolve destination addresses. Despite being younger than H.323, SIP products have supported this feature since the beginning while most H.323 equipment still does not.

The reasons might be that the usage of DNS is more natural to a protocol from the IETF than it is to one from the ITU-T. On the other hand, calling telephone numbers also required static peer configurations.

SIP products often provide the possibility to use regular expressions to ease the management of routing tables - at least for those that are familiar with the concept.

-} 7.2.2 Using H.323

In the first version of H.323, the only way to implement address resolution was the usage of Location Requests (see Section 7.1.1). This mechanism required either peer relationships or the reachability of all gatekeepers in the Internet via multicast. Since the latter is obviously only usable for intra-domain address resolution, sending requests directly to peers was the only way to perform address resolution..

Using peer relationships works well as long as there are only a small number of servers involved, e.g., if you want to connect branch offices to the main site (see Figure 7.7). The mechanism is the same as described in Section 4.1.1.2.1.

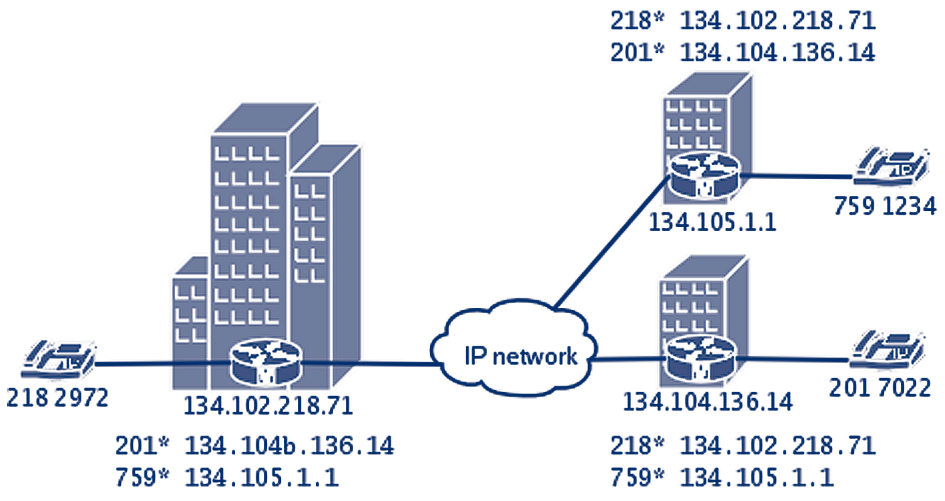


Figure 7.7 Using peers to route external calls.

This structure does not scale to a large number of connected sites. Manually configuring each server and its prefixes is error-prone and exhausting at best. The solution to this problem could be the usage of SRV-Records (see Section 7.1.4) or even TXT-Records that where defined for H.323 since version 2, but since most IP Telephony solutions where used for intra-domain communication and as a PBX replacement, dynamic address resolution was not implemented for a long time.

For this reason and because of 'legacy' VoIP equipment, the idea of a gatekeeper hierarchy was born. A gatekeeper that cannot resolve an address itself sends a location request to a higher level gatekeeper that acts as a clearing house for this request. This gatekeeper may also need to ask a higher-level gatekeeper.

The gatekeeper hierarchy is oriented at political or organisational structures. On top, there is a world gatekeeper that can route calls to the top level gatekeepers of all nations. The sub-structuring within a nation may vary. A dialling scheme must be applied to address the gatekeepers. To provide a testbed, ViDeNet came up with the Global Dialling Scheme (GDS) that is explained in more detail in Section 7.2.2.1. The problem with the GDS is that, in its original form, it differs from the E.164 numbering space except in country codes.

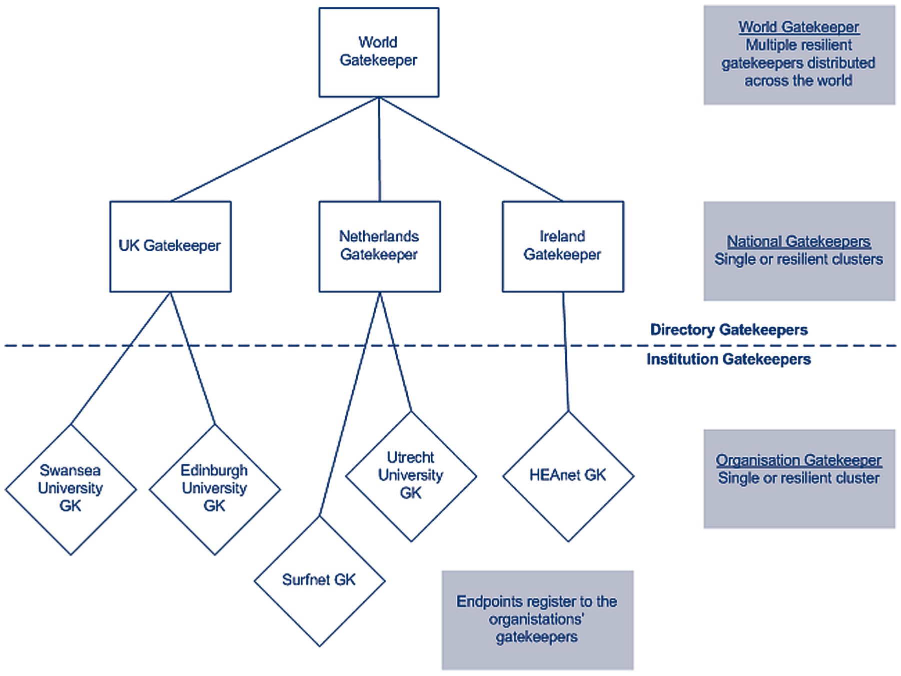


Figure 7.8 Gatekeeper hierarchy

-} 7.2.2.1 Global Dialling Scheme

The Global Dialling Scheme (GDS) is a new numbering plan for the global video and voice over IP network test bed, originally developed by HEANet, SURFnet and UKERNA. It resembles the international E.164 telephone system numbering plan, with some exceptions. With the GDS, you can number each participating videoconferencing endpoint, MCU conference and gateway. GDS provides easy, uniform dialling throughout the world.

7.2.2.1.1 The Global Dialling Scheme explained

Each basic number consists of four parts: <IAC><CC><OP><EN>

International Access Code (IAC)
This is also called the world gatekeeper prefix. It is defined as 00;

Country Code (CC)
This follows the ITU international access code system. For instance, the country code for the Netherlands is 31. See the following PDF document for country codes:
`http://www.itu.int/itudoc/itu-t/ob-lists/icc/e164_717.pdf`

Organisational Prefix (OP)
Many national research organisations follow the telephone number system in their country and use their area code and organisational telephone exchange prefix. For instance, SURFnet's OP is 302305. However, there are other possibilities. Some organisations use their administration number or make one up. National research organisations or videoconferencing service providers could instead supply you with an OP, as was the case with the old ViDeNet system. In any case, your OP must be unique within a country. If you do not know your OP, please contact your videoconferencing service provider, your national gatekeeper or the NASM working group (see below);

Endpoint Number (EN)
Your EN can be any number and it is decided by each organisation. However, we recommend that it is no longer than seven digits. Each endpoint number MUST be unique within the organisation. Both 305 and 1234567 are fine examples as long as they are unique.

7.2.2.1.2 Examples

The Megaconference informal test MCU:
00(IAC) 1(CC) 189(OP) 7201234(EN)

Typed into your videoconferencing endpoint, the number would simply look like:
0011897201234

7.2.2.1.3 Alphanumeric dial plan

The GDS also defines an alphanumeric dial plan. This part is equal to the alphanumeric dial plan of the old ViDeNet and should be in the form: `<station ID>@<fully qualified domain name of the institution>`

An example is: `egon.verharen@surfnet.nl`

7.2.2.1.4 More information

More information on the GDS and the Numerical Addressing Space Management (NASM) working group overseeing its development can be found at:
- `http://www.wvn.ac.uk/support/h323address.htm`
- `http://www.vide.net/workgroups/nasm/index.shtml`

-} 7.2.2.2 Problems

The use of H.323 **LRQ**s and of a gatekeeper hierarchy is an easy way to enable all kinds of H.323 equipment to reach other sites. On the other hand, there are some problems that make this solution less desirable:

- **Latency** – The address resolution process may include multiple hops (e.g., from an organisational gatekeeper to another gatekeeper in another country, there are at least four hops). Each hop increases the time needed to process the request and eventually forward it. So it may take several seconds to resolve an address even before the call setup can begin. Speaking of call setup, one should consider that it is possible for a gatekeeper to put itself into the call signalling path. By changing the destination address of a **LCF** message to its own address, the higher level gatekeeper forces the requesting gatekeeper to pass the call to him. This may be useful for security reasons, e.g., when a gatekeeper's policy only accepts incoming external calls that originate from a trusted higher-level gatekeeper to avoid being spammed from an uncontrollable source. But again this adds some latency to the call setup.

- **Bottlenecks and single point of failure** – Every higher-level gatekeeper resolves addresses or routes of calls for its subordinate gatekeepers. It is critical that a gatekeeper is operational and can handle all incoming requests. To avoid bottlenecks or complete loss of higher-level gatekeepers, these must have a redundant layout, should have different locations and should use load-sharing mechanisms.

- **Cost** – Running a higher-level gatekeeper is expensive, because one must provide hardware that guarantees continuous availability..

- **Feature limit** – The usage of a hierarchy limits the availability of protocol features to the feature set of the 'dumbest' gatekeeper in the chain. Current versions of H.323 do, for example, carry more attributes than just **AliasName** and **IPAddress**, as it is the case in the **Location Request** (**LRQ**); such additional information specifies desired protocols or a hop count. A gatekeeper that uses a lower-protocol version will ignore those attributes and will not add that information in its reply. While this is annoying, it is still tolerable since advanced address resolution is not that important. But it is even worse when the call signalling is routed through the gatekeepers as well. In this case, it might happen that two communication partners using new equipment that supports a special feature (e.g., transmitting an image of the calling party on call setup) can not use this feature because of a gatekeeper in the call path that is not aware of the feature. So, hierarchical address resolution and call routing hinder research on IP Telephony.

- **Protocol limitation** – By nature, the gatekeeper hierarchy is limited to H.323 and is difficult to merge with the SIP world. It is, of course, possible to provide gateways to SIP proxies but the infrastructure and the configuration will get complex and more error-prone.

-}7.3 Utopia: setting up global IP Telephony

To set up an infrastructure to provide address resolution for international addresses that supports H.323 and SIP, not all of the protocols described in Section 7.1 are useful:

- H.323 LRQs – Among other reasons this solution is not suitable because it does not support SIP;
- H.225.0 Annex G – Does not support SIP;
- TRIP – Supports H.323 and SIP but can be used for telephone numbers only;
- ENUM + SRV-Records – Supports SIP and H.323 and can be easily configured to support more protocols. It does not impose a network of peer relationships because of its decentralised nature. It supports URIs and telephone numbers.

It is obvious that the combined use of ENUM for mapping telephone numbers to URIs and SRV-Records to resolve URIs upon a well-understood and scalable mechanism like DNS is an ideal solution to this problem.

-}7.4 Towards Utopia

The use of ENUM and SRV-Records might be the ideal solution, but it still has some obstacles that need to be overcome. Most available VoIP products do not support both features. There are some products, especially in the SIP world, that support SRV-Records but just a few that support ENUM as well. Products that are already in use probably will not be able to resolve telephone numbers via ENUM.

Some vendors will offer ENUM/SRV-Records capabilities by software upgrades, but not everyone will. To protect investments that have already been made, it is necessary to find a solution that integrates older IP Telephony equipment.

-} 7.4.1 Call Routing Assistant (CRA)

To enable an ENUM-unaware H.323 Gatekeeper or SIP Proxy to use this feature, one can make use of the ability to configure a default server (further on called Call Routing Assistant (CRA)) that all unknown calls shall be routed to. Such a configuration option is provided by nearly all IP Telephony servers. The Call Routing Assistant will act as the VoIP equivalent of a default gateway and perform the call routing instead of the ENUM-unaware server.

Such a Call Routing Assistant could also be used as a firewall to protect the internal IP Telephony server by opening just a single hole in the CRA as a gateway for external communication.

If the internal IP Telephony server only supports one signalling protocol, the CRA could include SIP/H.323 Gateway functionality.

The Call Routing Assistant is not a special product. In theory, every IP Telephony server that can be configured to route or resolve calls, without registering the other server, can be used in this place. Efforts are being made by the Center for Computing Technology in Bremen, Germany to provide an easy-to-use, open source CRA implementation.

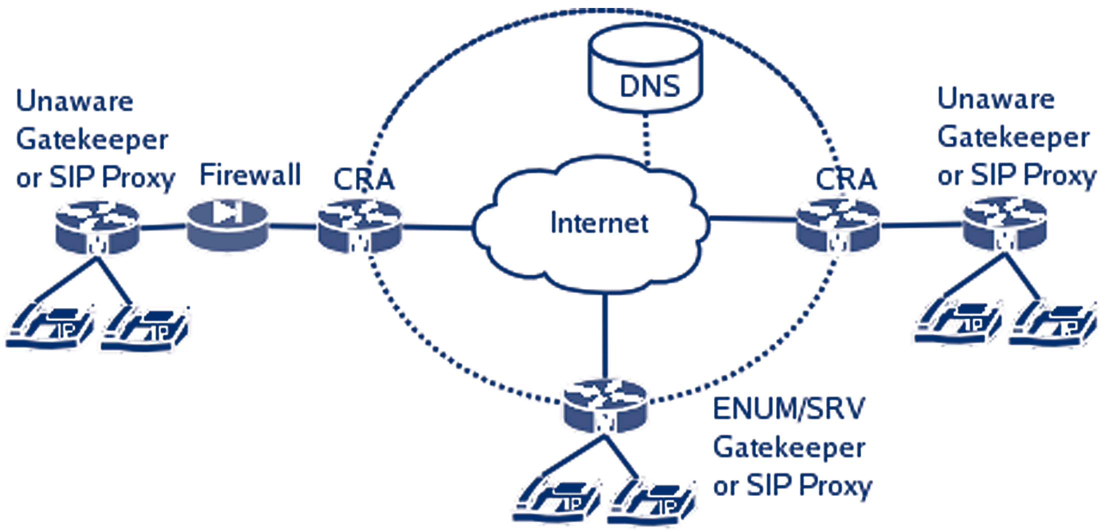


Figure 7.9 Use of a Call Routing Assistant

8 Regulatory/Legal considerations -(.

-(.8.1. Overall

For a legal classification of Voice over IP, several aspects have to be taken into consideration. Where regulation focuses on technology – and there on the regulation of voice telephony – the definition of voice telephony is the starting point for all further legal and practical considerations. But when it comes to Voice over IP, terminology is used quite inconsistently. The lack of a clear definition often leads to misunderstandings and also problems in the exact legal classification.

Internet telephony (or Voice over IP, VoIP) can be defined as a collection of Internet applications for real-time voice traffic over data networks using the Internet protocol (IP) whereby the quality of the transmitted voice depends on various factors such as available network capacity, gateways, audio codecs used, etc.

For Oftel, "Voice over Internet Protocol (VoIP) is the generic name for the transport of voice traffic using Internet Protocol (IP) technology. The VoIP traffic can be carried on a private managed network or the public Internet or a combination of both. A wide range of applications and services could use VoIP technology, from traditional telephone services to interactive games." 'Internet telephony' (also referred to as Voice over the Internet) – according to Oftel – "is a specific type of VoIP service that uses the public Internet to carry the IP traffic."[4]

The ITU [5] uses different definitions depending on the nature of the principle underlying the means of transmission. Therefore, Internet Protocol (IP) telephony is the transmission of voice, fax and related services over packet-switched IP-based networks. Internet telephony and VoIP are – according to the ITU – specific sub-sets of IP Telephony. Internet telephony is therefore IP Telephony in which the principal transmission network is the public Internet. Internet telephony is also commonly referred to as 'Voice-on-the-Net' (VON), 'Internet Phone,' and 'Net telephony' – with appropriate modifications to refer to fax as well, such as 'Internet Fax'. Voice-over-IP (VoIP) IP Telephony is then telephony in which the principal transmission network or networks are private, managed IP-based networks (of any type).

-(.8.2 What does regulation mean for Voice over IP?

Regulation in telecommunications intends to transfer a monopolistic market into a competitive one. Therefore, especially in the beginning of the opening of the market, former monopolists face strong obligations e.g., concerning access rights to their networks or price regulation. The target is to give new entrant operators the chance to gain market share. For incumbent operators (the former monopolists), regulation is therefore interfering with their business models and is a considerable cost factor.

4. *http://www.ofcom.org.uk/static/archive/oftel/publications/internet/2003/voip1103.pdf*

5. *See ITU Internet Reports 2001: IP Telephony, http://www.itu.int/ITU-D/ict/publications/inet/2000/index.html.*

Furthermore, regulation is based on public (provision of basic services to everybody) and state interest (license fees).

Where Voice over IP services are exempted from regulation, Voice over IP faces some benefits in comparison to classical voice services. In Asia and America, classical voice telephony is substituted more and more by Voice over IP whereby positive cost factors of Voice over IP are partly supported by regulation of classical voice telephony. To balance this - and to further uphold public and state interests - a trend can be recognised that tends to also impose regulatory measurements on Voice over IP services, the more mature and the more common those services become.

-(.8.3 Regulation of Voice over IP in the European Union

Looking at regulatory measures in the European Union relating to Voice over IP, there have been different approaches for quite a long time. On the one hand, there was the well- regulated PSTN world and, on the other, the parallel world of the Internet or more generally of IP, where there has been hardly any regulation perceived. In other words, telecommunications regulators were used to concentrating regulatory interventions on traditional switched networks whilst packet -oriented networks have been out of the telecom's legal focus.

-(. 8.3.1 Looking back into Europe's recent history in regulation

In the old regulatory framework, Voice over IP was exempted from regulation. The European Commission stated in a Communication dating back to 1998 that Voice over IP services do not face the common regulation set forth for voice telephony. The reason for that was the definition of voice telephony in article 1 of the 1998 Voice Telephony and Universal Service Directive.

According to Article 1 of the ONP Voice Telephony and Universal Service Directive [6] "*voice telephony service*" means a service available to the public for the commercial provision of direct transport of real-time speech via the public switched network or networks such that any user can use equipment connected to a network termination point at a fixed location to communicate with another user of equipment connected to another termination point'. In other words, commercially, directly transported and switched Internet telephony provided for the public in real time is considered as being voice telephony, if the quality of service can be compared to ordinary fixed or mobile telephony and if it is enabling any user to communicate with every other user in fixed or mobile networks.

In its considerations, the Commission concluded that for the time being (which meant the time of publishing the Communication) VoIP was not in commercial use and direct transport of real-time speech via the public switched networks enabling any user to use equipment connected to a fixed network termination point to communicate with another user of equipment connected to another termination point was not available. The Communication was put under examination by the Commission in 2000. The results were published in summer 2000 stating that, despite technological developments and market trends, Voice over IP still does not underlie the regulation for voice telephony. The reasons were low quality and reliability compared to traditional voice

6. *Directive 98/10/EC of the European Parliament and of the Council of 26 February 1998 on the application of open network provision (ONP) to voice telephony and on universal service for telecommunications in a competitive environment, OJ L 101/24 as of 1st of April 1998.*

services and a lack of a distinct market for VoIP services (given the fact that VoIP services are not offered as such, but – according to the Commission – always in combination with e.g., data transmission).

Between 2000 and 2003 (when the New Regulatory Framework was published), the European Commission did not publish any document on VoIP. A Communication is, as such, 'soft law', meaning that it does not have the same legal consequences than a Directive or a Regulation (e.g., there is no need for the Member States to transfer it into national law), even though it is a means that it is often used to clarify things. Regarding Voice over IP there was legal uncertainty whether VoIP could be considered as voice telephony and therefore being held as voice service with all legal and practical consequences such as licensing and interconnection. With its Communication the European Commission clarified that point for the time being and stated that VoIP was not seen as voice service and therefore did not face voice telephony regulation.

In the New Regulatory Framework – which aims to be technologically neutral – the chance that the European Commission will work on another Communication to clarify the legal status of VoIP has been decreased. Electronic communications networks and electronic communications services are facing the same regulation no matter which technology they are using. Therefore, the basic answer to the question whether VoIP-services are facing the same regulation than PSTN-voice telephony, will be yes provided that the service is offered to the public.

-(. 8.3.2 The New Regulatory Framework - Technological Neutrality

With the European Union's New Regulatory Framework things have been changing.
The Framework states the principle of technological neutrality, meaning that there is no longer a distinction in the regulations made based upon technology between switched- or packet-based networks and/or services. The new rules are applied to all electronic communication services and networks. In other words, regulators today neither impose, nor discriminate in favour of the use of a particular type of technology except where necessary.

At first glance, it seems that technological neutrality leads to less regulation. It seems as though there were only generic rules and the market, as such, is developing according to market principles. Looking more closely, a technology-independent approach might lead to even more regulation, as potentially every network and every service then faces regulation in one form or another.

Voice over IP is a good example of the covergence between the well-regulated world of switch -fixed and mobile telephony and the Internet world, which has traditionally claimed independence from any regulation. If regulation is applied in a technologically-dependent manner and Voice over IP is not seen as voice telephony that falls under regulation (as it has been the case in the European Union up till now), we are in a grey area where Voice over IP operators do not face the same rights and obligations as traditional operators. This means, for example, that they do not have to apply for licenses, do not get access to the numbering resources, are not permitted to interconnection with others, etc. Where regulation is technology-neutral there is no differentiation made. In such a system (lately introduced in the European), Voice over IP might be considered as just another (voice) service facing the same rules as any other service based on another technology. Where the absence of regulation in the past seemed to foster the deployment of Voice over IP, it is likely that the application of regulation now will slow down that process. But in the long run, it seems to be almost certain that Voice over IP will win against traditional, switched fixed and mobile telephony as sole voice communication service.

-(. 8.3.3 New Regulatory Framework - an overview

The above mentioned New Electronic Communications Regulatory Framework consists of one general Directive, Directive 2002/21/EC of the European Parliament and of the Council of 7 March 2002 on a common regulatory framework for electronic communications networks and services (Framework Directive)[7] and four specific Directives:

- Directive 2002/20/EC of the European Parliament and of the Council of 7 March 2002 on the authorisation of electronic communications networks and services (Authorisation Directive)[8];
- Directive 2002/19/EC of the European Parliament and of the Council of 7 March 2002 on access to, and interconnection of, electronic communications networks and associated facilities (Access Directive)[9];
- Directive 2002/22/EC of the European Parliament and of the Council of 7 March 2002 on universal service and users' rights relating to electronic communications networks and services (Universal Service Directive)[10];
- Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications (Directive on privacy and electronic communications)[11] sector.

The Framework Directive provides the overall structure for the new regulatory regime and sets out the policy objectives and regulatory principles that National Regulatory Authorities must follow. It also requires that market analyses be carried out before regulation is imposed. The Authorisation Directive establishes a new system whereby persons do not require prior authorisation before providing electronic networks and services. It includes provisions relating to enforcement of conditions and the specific obligations which can be imposed. The Universal Service Directive deals with the obligation to provide a basic set of services to end-users. The Access Directive sets out the terms on which providers may access each others' networks and services with a view to providing publicly available electronic communications services. Finally, the Privacy Directive establishes users' rights with regard to the privacy of personal data.[12] The New Framework for the regulation of electronic communications came into force in April 2002 and had to be transferred into national law by the EU Member States by 25 July 2003. Only six Member States (Finland, Denmark, Sweden, United Kingdom, Ireland and Italy) made it in time.[13] The EU launched infringement proceedings against the other Member States. Germany said it will transpose the Directives by May 2004, Belgium by January 2004, Greece has still no timetable, the law in France is still discussed in the Parliament, Luxemburg does not foresee a date, and the Netherlands expect the new law in the spring and Portugal just approved the law in December.

7. *http://europa.eu.int/information_society/topics/telecoms/regulatory/new_rf/documents/l_10820020424en00330050.pdf*
8. *http://europa.eu.int/information_society/topics/telecoms/regulatory/new_rf/documents/l_10820020424en00210032.pdf*
9. *http://europa.eu.int/information_society/topics/telecoms/regulatory/new_rf/documents/l_10820020424en00070020.pdf*
10. *http://europa.eu.int/information_society/topics/telecoms/regulatory/new_rf/documents/l_10820020424en00510077.pdf*
11. *http://europa.eu.int/information_society/topics/telecoms/regulatory/new_rf/documents/l_20120020731en00370047.pdf*
12. *http://www.oftel.gov.uk/ind_info/eu_directives/index.htm*
13. *See: http://europa.eu.int/information_society/topics/ecomm/all_about/implementation_enforcement/country_by_country/index_en.htm*

-(. 8.3.4 Authorisation system instead of licensing system

Under the new regime, communications providers can offer (electronic) services and networks without first having to seek permission or authorisation in terms of a license (exceptions are possible for the allocation of scarce resources). The new system is based on the principle of a general authorisation containing general conditions outlining the minimum obligations for providers. This means that companies can offer electronic communication networks and services (including Voice over IP) to end users without first having to notify or to seek permission of the regulator.

With the new framework, it has generally become easier for network and service operators to start off with the provision of their services. Especially for Voice over IP operators, this change means a simplification compared to the previous situation where they had to take a decision whether to provide a service that is not considered as voice service or whether to go through the hassle of the (voice telephony) licensing process. Under the new regime, Voice over IP providers and other (voice) providers have equal rights and duties e.g., concerning interconnection, numbering, directory entries or emergency calls.

With the introduction of the new general authorisation regime on 25 July 2003, all classes of telecommunications licenses were revoked. There are no different licenses e.g., for voice telephony provided over fixed networks, for voice telephony provided over mobile networks or for offering of leased lines. The differentiation between regional and national licenses also ceased to exist as well as different terms and conditions in these licenses. Existing licenses, expired at the end of July 2003, and were replaced by general authorisations. General authorisations, by definition, (see Annex to the Authorisation Directive[14]) contain just general conditions. General conditions may be financial contributions to the funding of universal service, administrative charges, rules concerning Interoperability of services and interconnection of networks and environmental and town and country planning requirements. Other specific conditions, such as effective and efficient use of frequencies, can be imposed on individual communications providers if they are using scarce resources.

-(. 8.3.4.1 Example: VoIP in the New Framework in the United Kingdom

The United Kingdom's regulator, Oftel[15], recently took a closer look at the regulation of VoIP in the new framework and published its findings. In summary, Oftel states that it is regulating VoIP. The reason given for regulating VoIP services is that those services are ‘electronic communication services’ for the purposes of the British Communications Act 2003 ('the Act'). The Act regulates, amongst other things, the provision of ‘electronic communications networks’, ‘electronic communications services’ and ‘associated facilities’.[16] Oftel also concludes that interconnection is likely to be relevant for electronic communication networks and services irrespective of the underlying technology (e.g., circuit-switched networks or IP networks). It is still (just as in the previous legal framework) possible to make a regulatory distinction between publicly available

14. *http://europa.eu.int/information_society/topics/telecoms/regulatory/new_rf/documents/l_10820020424en00210032.pdf*

15. *Oftel is now part of Ofcom. Ofcom began its regulatory duties on 29 December 2003. It replaces UK's five previous regulators - the Broadcasting Standards Commission, the Independent Television Commission, Office of Telecommunications (Oftel), the Radio Authority and the Radio Communications Agency.*

16. *http://www.ofcom.org.uk/static/archive/oftel/publications/internet/2003/voip1103.pdf* telephone services and services that are not publicly available. Oftel says that specific requirements for publicly available telephone services can be set forth in general authorisations. Where VoIP services are not considered to be publicly available telephone services, those conditions will not apply.

Oftel states that a VoIP service should be regulated as, a publicly available telephone service if any of the following conditions apply:

- the service is marketed as a substitute for the traditional public telephone service, or
- the service appears to the customer to be a substitute for the traditional public telephone service over which they would expect to access emergency numbers, directory enquiries etc. without difficulty; or;
- the service provides the customer's sole means of access to the traditional circuit-switched public telephone network.

Where a VoIP service is clearly being offered as an adjunct to a traditional telephone service or as a secondary service it is, according to Oftel likely not to be considered as publicly available telephone service. Oftel would, however, expect VoIP providers selling secondary services to ensure that the customers and third parties using the VoIP service are fully aware of the nature and limitations of the service.[17]

-(. 8.3.5 Numbering

A full E.164 telephone number is a string of up to fifteen decimal digits that uniquely identifies a termination point in the global public telephone network. Every end-user connected to the public telephone network, fixed or mobile has an individual number and all these numbers are managed within the different national, regional or global numbering plans. The ITU deals with the overall management of the world's numbering resources and assigns country codes to individual countries (e.g., 49 for Germany, 33 for France, 44 for the UK, etc.), regional codes to regions who request them (e.g., '3883' for Europe) and global numbers for worldwide use (e.g., the 00800 worldwide free phone code).

Traditionally, fixed network telephone numbers were used to identify endpoints and to route calls to those endpoints.

For the deployment of Voice over IP services, numbering and access to numbering resources is important. It is highly unlikely that the addressing system will switch from E.164 numbers to URIs within a short time-period. Therefore it is important that VoIP services and endpoints can be addressed using E.164 numbers. For a worldwide Voice over IP service that allows the subscriber to connect from different points, it would be useful to have a worldwide service number. That would need a decision taken by the ITU to install such a number.[18]

One has to bear in mind that national numbering plans have historically been evolving. A sophisticated alignment of these numbering plans would cause major changes and problems to the existing numbering plans. Such changes would also involve enormous costs. Therefore, it is more likely that VoIP services will use numbers within the national numbering ranges than getting distinctive numbers out of a newly-defined numbering range.

17. *http://www.ofcom.org.uk/static/archive/oftel/publications/internet/2003/voip1103.pdf*

18. *In the past ITU-T SG2 had assigned the Universal Personal Telephone code +878 878 to be used for test reasons until 22. October 2000.*

The European Commission states that the availability of numbers for existing and new services is of crucial importance for competition and innovation. Member States should therefore design their numbering plans in such a way that they can cope with increased future demand and they should manage the existing numbering space to encourage efficient and effective use of numbers. [19]

With Voice over IP being treated as any other electronic communication service, Voice over IP providers are also entitled to get numbers from the national numbering range.

The European Commission says that numbers must be assigned to any undertaking providing or using electronic communications networks or services, within three weeks after receipt of a request. Procedures for assignment must be open, transparent and non-discriminatory. Short codes, e.g., carrier selection codes, and so-called golden numbers, e.g., numbers that are easy to remember, deserve special attention as they may represent a specific economic value. Member States may decide to assign such numbers or codes via competitive or comparative selection procedures, in which case, the assignment period may be extended until up to six weeks (Article 5 of the Authorisation Directive).[20]

Oftel says that communication providers may apply for public numbers for VoIP services in accordance with the requirements set out in the General Conditions for the allocation, adoption and use of numbers. Oftel considers about an appropriate number range specifically allocated for VoIP services.[21]

-(. 8.3.6 Access

According to the Access-Directive (2002/19/EC), access means the making available of facilities and/or services, to another undertaking, under defined conditions, on either an exclusive or non-exclusive basis, for the purpose of providing electronic communications services. It covers access to network elements and associated facilities, which may involve the connection of equipment, by fixed or non-fixed means (in particular this includes access to the local loop and to facilities and services necessary to provide services over the local loop), access to physical infrastructure including buildings, ducts and masts; access to relevant software systems including operational support systems, access to number translation or systems offering equivalent functionality, access to fixed and mobile networks, in particular for roaming, access to conditional access systems for digital television services; access to virtual network services.

Access is a generic concept covering any situation where one party is granted the right to use the network or facilities of another party, on either an exclusive or shared basis. As defined in the Access Directive, interconnection is a special form of access.[22]

The basic question here is whether VoIP providers can demand access from other operators and if so, to what facilities and at what price.

19. *http://europa.eu.int/information_society/topics/ecomm/all_about/todays_framework/public_resources/index_en.htm#Numbers*
20. *http://europa.eu.int/information_society/topics/ecomm/all_about/todays_framework/public_resources/index_en.htm#Numbers*
21. *http://www.ofcom.org.uk/static/archive/oftel/publications/internet/2003/voip1103.pdf*
22. *http://europa.eu.int/information_society/topics/ecomm/all_about/todays_framework/interconnection_interoperability/index_en.htm#access*

The Access Directive lays down a procedural framework for regulators to follow, and identifies factors to be taken into account when granting access, but does not specify precise access obligations. In general, access obligations are only imposed on operators that have significant market power in specific markets. Taking into consideration the above, VoIP providers are offering electronic communication services. Therefore they are entitled to ask other operators for access and enter into negotiations. Only operators with significant market power are obliged to offer access in a transparent, non-discriminative and cost-oriented way, whereby the regulator has the rights to control prices.

-(. 8.3.7 Interconnection

Interconnection covers the physical and logical linking of networks, and is an essential element in any multi-network environment. It allows the users on one network to communicate with users on other networks, or to access services provided on other networks. In a newly liberalised market, terms and conditions for interconnection to the incumbent operators' network are critical for successful market opening.

All operators of public communications networks in the EU have both a right and a duty to negotiate interconnection with each other. In the event of a dispute, the national regulatory authority may intervene.[23]

According to the Access Directive, interconnection is defined as the physical and logical linking of public communications networks used by the same or a different undertaking in order to allow the users of one undertaking to communicate with users of the same or another undertaking, or to access services provided by another undertaking. Services may be provided by the parties involved or other parties who have access to the network. Interconnection is a specific type of access implemented between public network operators: interconnection being defined as physical and logical linking between networks has to be understood technologically-neutral. In the past, interconnection was PSTN-to-PSTN interconnection, meaning connecting homogeneous networks based on SS7. In today's framework, interconnection can mean much more. Packet-oriented networks must be enabled to be interconnected with line-switched networks and vice versa. The basic questions to be answered in practical cases over the next months will be who is going to be obliged to pay eventually for additional equipment needed to make interconnection between heterogeneous networks possible. From a legal point of view, it will not be possible to reject a request for interconnection based upon technology. Between operators that do not have significant market power, interconnection pricing is a question of negotiation and insofar not restricted. There is also only limited power for regulators to intervene. As soon as one of the two operators involved in the case has significant market power, this one has to offer interconnection based on transparent, non-discriminative and cost-oriented criteria.

-(. 8.3.8 Quality of Service

In traditional telecommunication networks (PSTNs), Quality of Service was the key point and standardised quality requirements had to be fulfilled. In the PSTN, quality expectations

23. *http://europa.eu.int/information_society/topics/ecomm/all_about/todays_framework/interconnection_interoperability/index_en.htm* concerning services have traditionally been very high. There are a lot of features that are not regulated by law. Instead, features have been regulated by standardisation organisations. When it comes to voice, there are a lot of requirements to be fulfilled, e.g., answering times and priority routing for emergency calls.

Taking a closer look at the legal requirements, quality of service parameters, definitions and measurement methods can be found in Annex III of the Universal Service Directive (2002/22/EC). According to Article 11 of the same Directive, national regulatory authorities are entitled to specify additional quality of service standards. In the UK, for example, the regulator can impose technical interface standards to ensure end-to-end connectivity and interoperability. Annex III of the Directive sets forth quality of service parameters, definitions and measurement methods only. Operators are therefore obliged to measure the specific service-quality in their networks and provide that data to the respective regulator. There are no specific quality -requirements to be fulfilled by networks or services, nor threshold values that have to be met.

In other words, one could say that quality is not a regulatory obligation but a market request.

One has to consider the fact that there may be demand for cheaper services that may be of a lower quality. Accordingly to that, Oftel expressed an interesting thought regarding quality expectations. Communications providers should note that when VoIP services are provided using traditional E.164 telephone numbers, calling parties may not be aware, in advance, that they are calling a customer connected to a VoIP service. When providing a VoIP service that uses E.164 numbers, communications providers should take account of the quality of service that a calling party would normally expect when calling an E.164 telephone number.

-(. 8.4 Voice over IP in the United States

Just as in Europe, the legal status of voice over Internet protocol (VoIP) services depends on the decision whether to classify them as 'traditional telecommunications services' or 'information services'. Should the Federal Communication Commission (FCC) decide on the former classification, VoIP service providers will need to apply for licenses and to ensure that their operations comply with the same state and federal regulations as their counterparts in the traditional wire-line telephony arena.

On 1 December 2003, the FCC held a public forum on VoIP which was open to the public. [24]

In his opening remarks, FCC Chairman Michael Powell stressed his belief that IP-based services such as VoIP should evolve in a regulation-free zone. Even though topics like availability of emergency services, contributions to the Universal Service Fund, full access for persons with disabilities, safety for consumers, law enforcement and national security must be targeted and discussed, nevertheless VoIP was already in the attention of regulators in some of the States. In August 2003, the Minnesota Public Utilities Commission (MPUC) became the first US state regulator to make a ruling on the issue, deciding that VoIP market leader Vonage should be classified as a telecoms service provider, thereby needing to apply for a concession to operate in the state.

Vonage provides a service that permits voice communications over the Internet. It sells a service called Vonage DigitalVoice that enables its customers to engage in voice communications, with

24. *http://www.fcc.gov/voip/*

broadband Internet connections, using voice over Internet protocol (VoIP). It has customers in the state of Minnesota.

In October 2003, the U.S. District Court (DMinn) overturned this decision and issued its Memorandum and Order in Vonage v. Minnesota Public Utilities Commission, holding that Vonage is an information service provider and that the MPUC cannot apply state laws that regulate telecommunications carriers to Vonage.[25]

In Virgina, the State Corporation Commission has also taken notice of Vonage and is of the opinion that it is subject to its jurisdiction.

In Ohio, the State Public Utilities Commission started an inquiry into how telecommunications providers are using VoIP in Ohio to provide telecommunication services to Ohio consumers. It turns out that under Ohio law, companies that are, in fact, 'transmitting telephonic messages' are a common carriers subject to the State Public Utilities Commission's laws.

The Florida Public Service Commission is awaiting on answers to the pending AT&T petition to the FCC with regard to whether or not VoIP providers should be responsible for paying access charges to local phone companies when they offer similar services. There is a bill pending in Florida that will affect consumers living in Florida [26].

-(.8.5 Conclusion and Summary

Regulation in telecommunications intends to transfer a monopolistic market into a competitive one. In the past, telecommunications regulators were used to concentrate regulatory interventions on traditional switched networks whilst packet-oriented networks have been out of the telecom's legal focus.

In Europe, Voice over IP was exempted from regulation up to mid-2003. The European Commission stated in a Communication dating back to 1998 that Voice over IP services do not face the same regulatory burden than other voice services because they were not in commercial use and direct transport of real-time speech was not possible.

With the European Union's New Regulatory Framework, things have been changing. The Framework states the principle of technological neutrality, meaning that there is no longer a distinction in regulation made based upon technology between switched-or packet-based networks and/or services. Under the new regime, Voice over IP providers and other (voice) providers have equal rights and duties concerning authorisation, interconnection, access, numbering, directory entries or emergency calls. In other words, publicly offered Voice over IP is now regulated in Europe.

In the US, the legal status of voice over Internet protocol (VoIP) services depends on the decision whether to classify them as 'traditional telecommunications services' or 'information services'. The US regulator, FCC, has not taken a decision on whether to classify it one or the other way. Should the FCC decide on the former classification, VoIP service providers will need to apply for licenses and to ensure that their operations comply with the same state and federal regulations as their counterparts in the traditional wire-line telephony arena.

25. *http://www.techlawjournal.com/topstories/2003/20031016.asp*
26. *The Florida State Senate's VoIP Bill is available in the Internet, see http://tinyurl.com/b5nb/.*

A Appendix European IP Telephony Projects

A. 1 Evolute

http://evolute.intranet.gr

The Evolute project addresses the issue of supporting multimedia services in general and SIP-based Voice over IP service in particular in heterogeneous networking environments. The focus here is on supporting secure access to such services regardless of the used access authentication technology. In this context, a SIP platform was enhanced with acAAA functionalities as well as gateways to SMS and Jabber and user credentials saved in SIM cards were used for authentication of the user. The project finished in 01.2004.

A. 2 6NET

http://www.6net.org

6NET is a three-year European project to demonstrate that continued growth of the Internet can be met using new IPv6 technology. It also aims to help European research and industry play a leading role in defining and developing the next generation of networking technologies. In the context of Voice over IP, the 6NET partners are deploying and experimenting with Voice over IP solutions, both SIP and H.323, using IPv6. This work involves porting of open source Voice over IP solutions such as the SIP Express Router to IPv6, provisioning of components to allow transparent Voice over IP communication between IPv4 and IPv6 networks and finally testing and deploying SIP and H.323 IPv6 capable components over a European wide IPv6 network.
The project will be finished in 12.2004.

A. 3 Eurescom P1111 (Next-Gen open Service Solutions over IP (N-GOSSIP)

http://www.eurescom.de/public/projects/P1100-series/P1111/default.asp

This project was funded by various telecommunication companies in Europe including Deutsche Telecom, Telecom Italia, Elisa Communication, OTE and France Telecom. The major goal of the project was to investigate the major problems hindering a vast and fast deployment of SIP such as QoS and NAT traversal, provisioning of intelligent services over SIP and demonstration of SIP -based services. The project finished in 2001.

A. 4 HITEC

The project HITEC (H.323-based IP Telephony Control) have dealt with the planning, the implementation and the prototyping of 'PBX VoIP', a managing system for the packet voice communication into an administratively, homogeneous Voice over IP network. The project aims to the design an intelligent 'PABX VoIP' system, which includes the functions of the H.323 terminal, the MCU system and the H.323-PSTN gateway in a 'GateKeeper Brain' functional element.

The prospective of an integration of telephone- and Web-based services enabled by the new technology. 'Voice over IP' has produced the realisation of various commercial equipments offering the voice/data integration at different levels; the peculiar feature of these solutions is that they are in compliance with the standards in a more-or-less pronounced manner. In this framework, the HITEC project, carried out by META center of Consorzio Pisa Ricerche (`http://www.meta.cpr.it and funded by Fondazione Cassa Risparmio di Pisa` (`http://www.fondazionecaripisa.it/`), aims to design a 'PABX VoIP' system, which includes the functions of the H.323 terminal, the MCU system and the H.323-PSTN gateway. Besides these, the system is characterised by the addition of 'gatekeeper' traditional management functions, new auxiliary functions in an entity denoted as 'GateKeeper Brain' (GKB).

These new functions are necessary to support the H.323 entity in the implementation of services either already defined in the H.323 systems, such as the supplementary services of the H.450 series, or services not yet standardised. The innovative approach of the HITEC project is the realisation of a PABX VoIP prototype in compliance with the H.323 standard, and characterised by limited hardware requirements (a simple PC with Linux O.S. and cards needed for the implementation of the gateway and GKB functions). These features should permit to the designed prototype to provide, at lower cost, the same functions as commercial systems already present in the market (legacy PABX or VoIP architectures, often based on proprietary solutions), to guarantee the interoperability with system-compliant to H.323 standard, and to make available a platform where users can develop new services exploiting the voice and data integration. An example of such integration is the introduction of call-centre services for supporting traditional Web-based services such as home banking and electronic commerce. This enhancement permits to the service operators to offer vocal support, which can give an important added value to the customer.

The project aims also at overcoming the limits of the H.323 architecture maintaining one rigid adhesion to the standard mechanisms, thanks to the introduction of new control centres of the H.323 zones. The intelligence of H.323 architecture is distributed on the gatekeepers, whose management operations (authorisations management of the H.323 clients, accounting of the calls, etc.) are not detailed by the H.323 standard. Thus, at the state-of-the-art, the implementations of gatekeepers is characterised either by a rather ingenuous management plan (for example, it is not possible to insert policy on the customers enabled to register themselves or to use the communication services), or by proprietary control architectures (an example is the AVVID architecture of the Cisco Systems).

The innovative contribution of this project is the definition of a system where the management functions (the 'GKB') are located. The GKB is defined in order to increase the flexibility and the capillarity of the management plane of the VoIP architecture while maintaining compatibility with the H.323 standards. In the project vision, the gatekeeper interacts with the GKB and depends on it for whichever decision related to the normal procedures of H.225/H.245 protocols. The transfer of some typical functions of gatekeeper in the auxiliary subsystem GKB and their integration with management function of the H.323 zone produces two main advantages:

- Access to the management system (configuration, logging or accounting procedures) is allowed to several client levels (administrator or normal client), without influencing the standard mechanisms of the gatekeeper, by means of disparate management interfaces (e.g., Web interface);
- The subsystem GKB constitutes an interaction point among H.323 zones allowing the improvement of inter-zone communication processes.

The realisation of the PABX VoIP system is based on the approach of open architecture 'Openh323', on platform IA32 with Linux O.S., according to the paradigm application/server. The advantages of this solution are the high reliability, flexibility and robustness of the Linux O.S. and in the perfect conformity to the H.323 standards implemented by the Openh323 library. Furthermore, this library constitutes a powerful development environment of H.323 applications, and, therefore, it is considered, not only by academic and standardisation entity, but, above all, by several manufacturers.

A. 5 The GRNET/RTS project

`http://rts.grnet.gr`

The Greek Research Network (GRNET) has been developing central videoconferencing services at the national level for the academic and research community. The main points of action of the project are: a scheduling interface for facilitating the organisation of meetings through automated invitations to conferences and for managing MCU and Gateway resource reservation. The Web interface is based on Apache/PHP/MySQL/LDAP. Authentication mechanisms for securing endpoint participation in specific, scheduled meetings are applied. The techniques involve gatekeeper interaction with RADIUS, MySQL, and LDAP, as well as the Cisco MCM GKAPI which has been explored as a means of controlling ARQs targeting scheduled MCU calls from legitimate, registered participants. Also, limited SNMP management of MCU conferences through a Web interface has been implemented.

A. 6 SURFWorks

`http://www.surfnet.nl/innovatie/surfworks/voip`

The SURFnet project called 'SURFWorks' has a component to stimulate the use of Voice over IP and VC, obtain knowledge and disseminate it and provide an H.323 and SIP service.

A. 7 VC Stroom

`http://www.surf.nl/projecten/index2.php?oid=136`

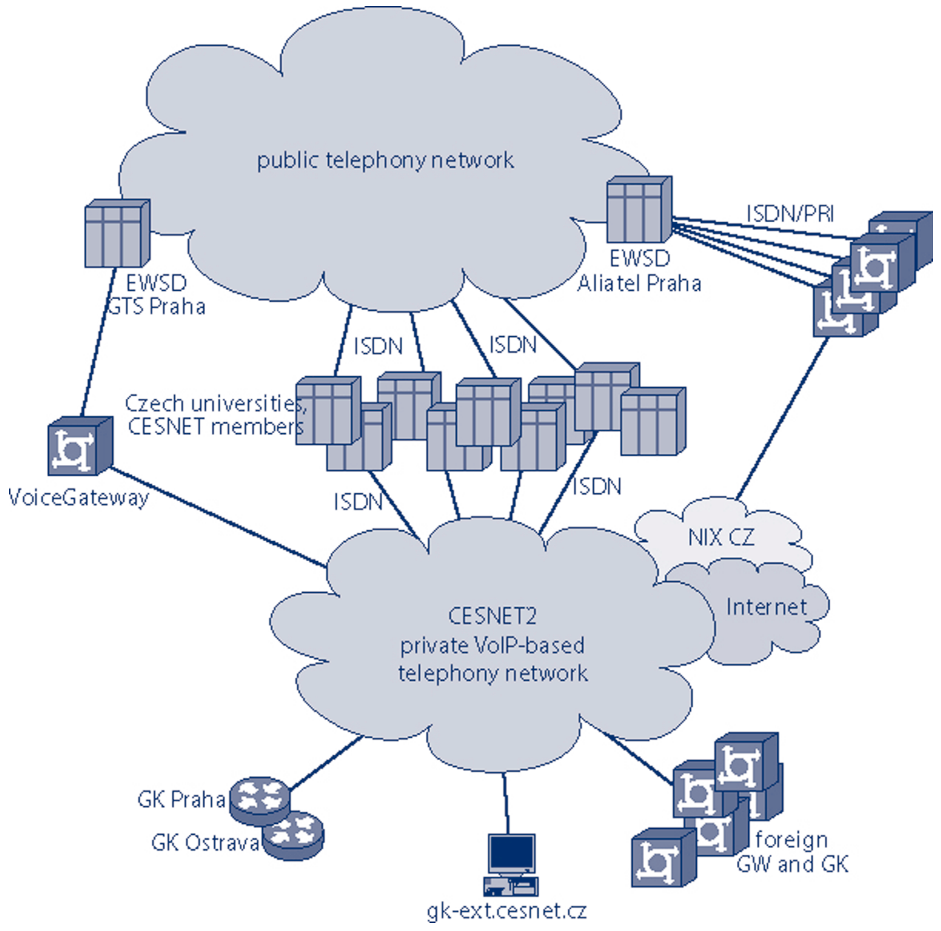
VC Stroom is a project of multiple institutions in the Netherlands to promote the use of VC in the educational setting.

A. 8 Voice services in the CESNET2 network

`http://www.cesnet.cz/english/project/iptelephony`

The CESNET IP Telephony project started in 1999 with the goal of investigating the possibilities of data and voice network convergence. CESNET operates the Czech NREN. A lot of experience was acquired about the optimal interconnection configurations of various telephony devices (PBXs with different interfaces, voice gateways, gatekeepers, IP phones, etc.). It turned out that most popular IP Telephony service has been least-cost routing. Over time, the 24 PBXs in universities and research institutes have been interconnected over the Czech NREN with two gateways to PSTN. The network structure is illustrated in Figure A.1.

Figure A.1. CESNET IP Telephony Network



B Appendix IP Telephony Hardware/Software

> Product Name: Windows Messenger
Product URL: `http://www.microsoft.com/windows/messenger/`
Vendor: Microsoft
Supported Protocols: SIP
Platform: Windows
Description: A well-know SIP softphone from Microsoft. There are different versions of the software and not all of them support SIP. This is unfortunately confusing. The most standard -compliant seems to be version 4.7. The phone is easy to use, but there are several violations of specifications. It supports audio, video, instant messaging and white board.

The client sends **BYE** instead of **CANCEL**. Microsoft uses a 'proprietary extensions of the presence' document and some other software vendors implemented the extensions as well. The realm of digest credentials must be same as the hostname of the server; otherwise the client refuses to authenticate.

New versions of Windows Messenger use **INVITE** to establish a session before an instant message is sent and thus are incompatible with other user agents, even with older versions of Windows Messenger. It does not support any NAT traversal techniques but it can be used from behind NATs with some aid from a SIP server in the public Internet because it implements symmetric media and signalling.

> Product Name: kphone
Product URL: `http://www.wirlab.net/kphone/`
Vendor: Wirlab
Supported Protocols: SIP
Platform: UNIX/Linux
Description: A user agent for Linux. Versions up to 4.0 were linked with KDE, starting from 4.0 kphone. It does not require KDE libraries anymore and uses qt only. The phone supports presence and instant messaging based on SIMPLE standards. Video conversation is supported using vic. The phone is not very stable yet and crashes from time to time. It supports G.711, GSM, and iLBC codecs.

> Product Name: Linphone
Product URL: `http://www.linphone.org/`
Vendor: Simon Morlat
Supported Protocols: SIP
Platform: UNIX/Linux
Description: It is a simple SIP user agent for Linux and uses the GNOME frameworks. The phone is unstable and crashes from time to time.

> Product Name: X-Pro, X-Lite
Product URL: `http://www.xten.com/`
Vendor: Xten Networks
Supported Protocols: SIP
Platform: N/A
Description: It is a good softphone with many interesting features. It has sophisticated NAT detection mechanisms. SIP support is good. There seem to be no interoperability issues (as far as we have tested). X-lite is a free version; X-Pro must be purchased. The softphone is also available also for WindowsCE-based PDAs.

B. 2 Hardphones

> Product Name: Cisco 7960
Product URL: `http://www.cisco.com/warp/public/cc/pd/tlhw/prodlit/7960_ds.htm`
Vendor: Cisco Systems
Supported Protocols: H.323, SIP
Platform: N/A
Description: Operational Experience: It supports H.323 under the assumption of having a Call Manager, which is a Cisco software product to manage Cisco hardphones. The communication between Call Manager and Cisco 7960 is done using the Cisco proprietary Skinny protocol (not a standard H.323 protocol!)and therefore the standard H.323 communication with an external client is only performed with the Call Manager and not with the phone itself (limited standard compliance of the phone). It supports also SIP protocol but from a standard point of view. With the following firmware (Application LoadID: P0S30203; BootLoadID: PC03A300. It was found to be perfectly interoperable with VOCAL system and with MSN messenger. Please pay attention that Power Cord is not included in default selling configuration. Take care when buying it or delays will occur before you can effectively use it (because of Cisco vendor's slowness). Overall Evaluation: Good hardphone. It is a little bit too expensive and not completely H.323 compliant. The SIP part is very good and standard-compliant.

The phone has excellent design. It is very comfortable to use, even during long conversations. The display is big enough to be seen even from a great distance. The phone has six lines. A SIP image has to be loaded into the phone before it can be used as a SIP phone. SIP-standard compliance is very good up to 6 SIP accounts can be registered simultaneously.

Unfortunately it is not possible to decline an incoming call. The phone can be switched into Do-not-disturb mode, though. Short packets containing just four zeroes (often used to keep the NAT bindings alive) freeze the phone. This should hopefully be fixed in the latest SIP model (not tested yet).

> Product Name: Adtech SI-150 IP Phone
Product URL: `http://www.adtech.be/text/si150.php`
Vendor: Adtech
Supported Protocols: H.323, SIP
Platform: N/A
Description: Operational Experience: It supports H.323 but with a particular version of the

firmware we have experienced some interoperability problems with standard H.323 clients (NetMeeting). Regarding the SIP part, with the firmware version SIP 2.07.06 CS 49AC, it showed some interoperability problems with VOCAL systems and MSN messenger. Some bug fixes are needed in order to avoid hook and crash problems of called clients. Overall Evaluation: Not yet a mature hardphone (at least with the firmware version we tested). It is quite inexpensive.

> Name: Komodo Fone (now Cisco ATA)
Product URL: `http://www.cisco.com/warp/public/cc/pd/as/180/186/index.shtml`
Vendor: Cisco (formerly Komodo)
Supported Protocols: SIP, H.323
Platform: N/A
Description: Black-phone-2-Ethernet/analogue-line adapter, both H.323 and SIP (currently SIP only for the adapter). The adapter supports two lines, only one of them at a time can use G.729 codec; the other one can use G.711 only. It supports symmetric signalling and media and can work from behind NATs. Standard-compliance is good.

One of the nice features is the possibility to create dialling plans using regular expressions. Any analogue phone that supports DTMF dialling can be connected to the adapter.

> Product Name: BudgetTone-100
Product URL: `http://www.grandstream.com`
Vendor: Grandstream
Supported Protocols: SIP
Platform: N/A
Description: This is the cheapest SIP phone available at the time of completing this document. The price is very low, but you get what you paid for. The phone is mostly standard-compliant; from time to time there are some interoperability problems but the manufacturer fixes them quickly. The biggest problem of the phone seems to be its HW. The phone can dial numbers only; it supports STUN and symmetric media and signalling. Many codecs are supported. iLBC support has been announced. If you need the cheapest phone available then BudgetTone is your choice. Be prepared to receive a phone that looks like a toy with a not very friendly user interface.

> Product Name: 5055 SIP Phone
Product URL:
`http://www.netergymicro.com/products/reference_designs/ip_t2.html`
Vendor: Mitel Networks
Supported Protocols: SIP
Platform: N/A
Description: The 5055 is a high-quality SIP phone that meets the price/performance needs of business users. Also available is the 5305 desktop conference unit and the 5310 boardroom conference unit. Mitel is a traditional manufacturer of telephones. 5055 has very solid design and is comfortable to use. The phone has many programmable buttons. The display is very small but functional. SIP implementation is good without any serious interoperability problems. The phone is rather expensive compared to other SIP phones available on the market. Audio quality is very good.

> Product Name: XPressa
Product URL: `http://www.pingtel.com/products.jsp`
Vendor: Pingtel
Supported Protocols: SIP
Platform: N/A
Description: A hardphone with unusual design. We had some problems with record routing (loose routing) which we were unable to solve so we did not use the phone much.

> Product URL:
Vendor: Siemens
Supported Protocols: H.323
Platform: N/A
Description: The former HiNet LP 5100â’¢ (in 1999) comes without an integrated switch and supports only 10 MBit/s. The supported codecs include G.711 and G.723. While in theory multiple firmwares can be used on this phone, the most popular was the H.323 (V2) firmware.

Depending on the firmware/application version the phone can be remotely configured via HTTP (using port 80 or 8080 - depending on firmware) and (sometimes only) using a special Windows Deployment Tool that allows the configuration of multiple ones at once.

To upload a new firmware, older versions required the use of DHCP and TFTP to update the firmware and FTP to access the application file.

New firmware versions have been seen booting the phones once in a while - regularly, when the phones aren't registered to a gatekeeper and without obvious reason while registered.

The telephones support the H.323 feature of Registration KeepAlive, but ignore the registration timeouts from the gatekeeper and resend their registrations every 60 seconds. If they are not currently registered the interval decreases to 30 seconds. The telephone does not support the FastStart protocol feature (at least not in the old versions).

> Product Name: optiPoint 300 basic
Product URL:
Vendor: Siemens
Supported Protocols: H.323
Platform: N/A
Description: It is a H.323 phone with a limited display (numbers only), less function keys that its big brother 300 had and without G.723 support. It supports H.323 FastStart procedures (10 MBit/s only).

> Product Name: optiPoint 400 standard
Product URL:
`http://www.siemens.com/index.jsp?sdc_rh=&sdc_flags=3&sdc_sectionid=2&sdc_secnavid=110&sdc_3dnvlstid=&sdc_sid=10240442142&sdc_countryid=0&sdc_mpid=0&sdc_unitid=0&sdc_conttype=3&sdc_contentid=1004068&sdc_ggid=17&sdc_langid=1&sdc_m4r=`
Vendor: Siemens

Supported Protocols: H.323, SIP
Platform: N/A
Description: This is a phone with integrated 10/100 Mbit/s mini switch, display and numerous features. Like the predecessor optiPoint 300 advanced, this phone can run different firmwares: HFA (CorNet), H.323 and SIP The H.323 protocol behaviour and the available features are pretty much the same as for the predecessor (see above).

> Product Name: Snom 100
Product URL: `http://www.snom.com/snom100_en.php`
Vendor: Snom.de
Supported Protocols: SIP, H.323
Platform: N/A
Description: The phone is easy to use with Graphical LCD display and four Softkeys. Caller Id, Hold, Divert and Transfer, Call Waiting Indication, Message Waiting Indication, Speed Dial, Phone Book, Call and Deny List, HTTP Server, Echo cancellation. The firmware can be upgraded over http.

> Product Name: SoundStation IP 5000
Product URL:
`http://www.polycom.com/products_services/0,1443,pw-182-3073,00.html`
Vendor: Polycom
Supported Protocols: H.323 (SIP and Skinny announced)
Platform: N/A
Description: This is a conference phone supporting multiple signalling protocols (depending on firmware). The H.323 firmware supports FastStart and Tunnelling (since version 2.5) but up to now (2.8) no H.450. It has very good audio quality and supports Power-over-LAN.

> Product Name: IP Phone 7905
Product URL:
Vendor: Cisco
Supported Protocols: H.323
Platform: N/A
Description: Very simple H.323 phone without H.450 support or integrated switch. It exists in two versions: Either with AC adapter or only with Power-over-LAN support.

This is a cheap brother of popular 7960. The phone has smaller display but the design is as comfortable as in 7960. The SIP image is quite simple but standard compliant. The phone allows registration of just one line, it can dial numbers only and it is not possible to dial a domain (@iptel.org for example). It can be configured through a Web interface but the Web interface is very basic and has some shortcomings.

If you are looking for a cheap but well designed phone and do not mind that you will be not able to dial SIP URIs, 7905 is a good choice. By default it is not shipped with a power supply (Cisco assumes power over ethernet) so do not forget to order one.

> Product Name: IP 200
Product URL:
Vendor: innovaphone
Supported Protocols: H.323
Platform: N/A
Description: Excellent, full-featured (H.450, ...) H.323 phone with a fair sized display. Comes with an alpha-numerical keyboard and eases entering URL addresses. It can access phone-books via LDAP (not tested). Supports overlap dialling.

Early protocol firmwares had a problem when an incoming call has been cancelled by the calling party (receiving a RELEASE COMPLETE while still ringing) when the phone continued ringing and showing a TRAP/error dump on the display when going off-hook. (This problem might be long fixed).

Early hardware had a problem of an incorrectly applied capacitor leading to a fading display (contrast). Contact Innovaphone for a replacement.

> Product Name: i2eye DVD-1000
Product URL: `http://www.d-link.com/products/?pid=8`
Vendor: D-Link
Supported Protocols: H.323
Platform: N/A
Description: A broadband videophone hardware with built-in camera. Video and audio signal can be accessed via cinch connectors to connect television hardware. An external microphone can be connected.

The DVC-1000â”¢ has a very minimal H.323 support. H.323 is used to setup a call, while address resolution is done using D-Links LDAP server. The address of the LDAP server is fixed and can not be configured. The D-Link LDAP server can be used for free for all DVC users but doesn't offer resolving other addresses.

To make really good use of the DVC hardware, one needs a proxy instance for dialling that calls the target and the DVC (set to auto reply) at the same time and passing messages through.

> Product Name: VCON Escort
Product URL:
`http://www.vcon.com/solutions/videoconferencing/desktop/Escort_Cruiser/index.shtml`
Vendor: VCON
Supported Protocols: H.323
Platform: Windows
Description: The VCON Escort hardware H.323 and H.320 client is a PCI-based card that can be installed on any standard PC running a Windows OS. It allows connections at up to 1.5Mbps and includes features for data collaboration (T.120), quality of service (QoS) and interactive multi-cast for allowing viewers to watch a conference over a multicast network. It generally performs well, but compatibility with PC video cards seems to be crucial for trouble-free operation, as in some set-ups, sudden crashes during long operation are not uncommon.

> Product Name:VCON Falcon
Product URL: http://www.vcon.com/solutions/videoconferencing/group/Falcon/index.shtml
Vendor:VCON
Supported Protocols: H.323
Platform: N/A
Description:The VCON Falcon is a set-top-box H.323 and H.320 client. It includes a quality camera and microphone. It has a wide array of audio and video connectors to allow it to interoperate with projectors, screens, multiple video and audio sources, as most often found in a group videoconferencing settings. It is a reliable device, but the remote controlled management interface is difficult to work with and a bit limited (e.g. H.323 aliases cannot include special chars at all).

B. 3 Servers

> Product Name: OpenH323 Gatekeeper - GnuGK
Product URL: http://www.gnugk.org/
Vendor: Open Source
Supported Protocols: H.323
Platform: Any platform where you can compile the OpenH323 Library (Linux, Windows, FreeBSD, Solaris, etc.)
Description: Operational Experience: There are some minor problem with Netmeeting. Netmeeting does not support Gatekeeper Discovery, thus you should directly configure the gatekeeper address in the Advanced Calling Options. Netmeeting requests an incorrect bandwidth; disable bandwidth management to avoid problems with GnuGK. It has Radius, MySQL and LDAP support. The manual is written in English, Chinese and Portuguese. It is used in many commercial applications and it has nice graphical interfaces to configure it, to monitor the registrations, to define groups of endpoints, to do call management, etc. Overall Evaluation: it is simply the best Open Source gatekeeper available nowadays. For technical support there is the Gatekeeper Users mailing list.

> Product Name:VOCAL (Vovida Open Communication Application Library)
Product URL: http://www.vovida.org/
Vendor: Open Source
Supported Protocols: H.323, SIP, MGCP
Platform: Refer to http://www.vovida.org/applications/downloads/vocal/platform/1_5_0.html for an updated list (Linux, etc.)
Description: Operational Experience: The VOCAL software includes a Session Initiation Protocol (SIP)-based Redirect Server (RS), Feature Server (FS), Provisioning Server (PS), Marshal Server (MS) and Voice Mail Server (vmserver). Other applications are not included in the current release (1.5) and are: 1. SIP to MGCP translator 2. Policy server 3. Inet/Conference proxy server. 4. SNMP/NetMgnt 5. SIP to H323

Translator: It has IPv6 Support and a lot of subsidiary application (from the site http://www.vovida.org). Unfortunately, provisioning currently requires valid IPv4 addresses. For an updated list of know limitation please refer to:
http://www.vovida.org/downloads/vocal/1.5.0/doc/LIMITATIONS.txt.

Overall Evaluation: It is a al-in-one solution for small-medium size business unit. It needs complementary solutions to be deployed in tandem in order to become a distributed system (the management is still centralised with no possibility of interfacing to other domains).

> Product Name: OpenMCU (H.323 Conferencing Server)
Product URL: `http://www.openh323.org/code.html`
Vendor: Open Source
Supported Protocols: H.323
Platform: Any platform where you can compile the OpenH323 Library (Linux, Windows, FreeBSD, Solaris, etc.)
Description: Operational Experience: It can accept multiple simultaneous connections. From the first four clients that get connected, it accepts audio and video capabilities, from the following ones only audio. It determines which conference is required via the 'rooms' feature and adds the call to that conference. New rooms are created automatically and there is a default room for people who do not specify a room or cannot specify a room (e.g. NetMeeting). It initiates calls from the MCU to remote endpoints and supports audio loop-back mode in a specific room (ideal for setup of audio hardware and testing network performance). No dynamic configuration is possible; once the program is started the client is configured using the command line options (only statistics on call in progress and initiating new calls is possible).

Overall Evaluation: MCU software requiring a performing hardware infrastructure in terms of shared memory it is using. High customisation in terms of parameters (video compression and quality) makes it scalable in terms of computational requirements. It is really useful in case of multipoint conferencing. A drawbacks is the one of multipoint conferencing, i.e., it does not use multicast but multiple unicast connections.

> Product Name: SIP Express Router
Product URL: `http://iptel.org/ser`
Vendor: iptel.org
Supported Protocols: SIP
Platform: POSIX-like systems
Description: SER or SIP Express Router is a very fast and flexible SIP (RFC3621) proxy server. Written entirely in C, ser can handle thousands calls per second even on low-budget hardware. A C Shell like scripting language provides full control over the server's behaviour. Its modular architecture allows only required functionality to be loaded. Currently, the following modules are available: digest authentication, CPL scripts, instant messaging, MySQL support, a presence agent, radius authentication, record routing, an SMS gateway, a Jabber gateway, a transaction module, registrar and user location.

The server has been optimised for speed and is being used on a couple of major SIP servers. One drawback might be the quite complicated configuration file. The configuration requires good knowledge of SIP.

> Product Name: AppEngine
Product URL: http://www.dynamicsoft.com/prod_sol/AppEngine/appenginev4.php
Vendor: dynamicsoft
Supported Protocols: SIP

Platform: Sun Solaris
Description: It is a SIP application server written in Java. The server implements Java SIP servlets which allow creation of even complex SIP applications called servlets. The servlets can interact with HTTP servlets as well. Good documentation is available with many examples.

> Product Name: Cisco IP/VC 3510 MCU
Product URL: `http://www.cisco.com/univercd/cc/td/doc/product/ipvc/ipvc2_2/`2_2mcurn.htm
Vendor: Cisco
Supported Protocols: H.323
Platform: N/A
Description: This MCU is an older product, identical to the RADVISION OnLAN MCU, but OEMed by Cisco and currently not supported any more, as it has been replaced by the 3511 MCU. The 3510 is capable of connecting twelve participants at 384Kbps each, or a combination of conferences at different rates and different numbers of participants. The default hardware does not provide audio/video transcoding between participants, so conference settings must be matched by all. Continuous presence is supported, but with asymmetric video rates, i.e., each participant sends 384Kbps video but receives 4x384Kbps back from the MCU, for viewing four participating sites simultaneously.

Check the IP/VC Products page at
`http://www.cisco.com/univercd/cc/td/doc/product/ipvc/ipvc2_2/`
2_2mcurn.htm

> Product Name: Cisco MCM Gatekeeper
Product URL:
`http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/`
`122cgcr/fvvfax_c/vvf323gk.htm`
Vendor: Cisco
Supported Protocols: H.323
Platform: Cisco router - IOS based
Description: The Cisco Multimedia Conference Manager (MCM) is the name of the gatekeeper product bundled in special IOS feature sets (H.323 feature set) and can be installed on most Cisco routers that may be performing as gateways at the same time. The MCM is mostly geared towards VoIP and in supporting basic H.323 interoperability, but it has a number of extra Cisco-proprietary features as well. It can also act as an H.323 proxy for serving H.323 clients behind firewalls. The API that has been developed by Cisco allows extensive control of gatekeeper events by an external application, but it requires significant development effort to bear fruit.

Check the Cisco Gatekeeper External Interface Reference page at
`http://www.cisco.com/univercd/cc/td/doc/product/software/`
`ios122/rel_docs/gktmp4_1/index.htm`

B. 4 Gateways

> Product Name: OpenISDN (H.323 Call Generator)

Product URL: `http://www.gae.ucm.es/~openisdngw/home_en.php`
Vendor: Open Source
Supported Protocols: H.323
Platform: Any platform where you can compile the OpenH323 Library (Linux, Windows, FreeBSD, Solaris, etc.)
Description: Operational Experience: It requires ISDN cards to be properly installed and configured on the local machine in order to make connections with the ISDN. It works only with ISDN lines (no PSTN support) managing n calls simultaneously, as many as the ISDN channels available. The gatekeeper can be in a well-known IP address or it could be discovered in the network with broadcast RAS. It gives information about the call progress state to the user of the Switched Circuit Network that calls to the Gateway. This information is made with tones similar to those sent by the telephone offices. Support and development has now stopped and it requires a special old version of OpenH323 library to compile. No dynamic configuration is possible; once the program is started the client is configured using the command line options.

Overall Evaluation: It is a simple H.323/ISDN Gateway. It needs to be better investigated for complete H.320 compatibility for ISDN conferencing. Right now it seems to be only an audio gateway.

> Product Name: Asterisk Open Source PBX
Product URL: `http://www.asterisk.org/`
Vendor: Digium
Supported Protocols: SIP, H.323
Platform: N/A
Description: Asterisk is an Open Source, full featured hybrid TDM and VoIP PBX and IVR platform. It allows you to seamlessly integrate TDM (T1, PRI, FXS, FXO) and VoIP (IAX, SIP, H.323) technologies in a single PBX while providing full IVR functionality through any scripting language available on Linux.

> Product Name: Cisco IP/VC 3525 PRI Gateway
Product URL: `http://www.cisco.com/univercd/cc/td/doc/product/`
pvc/ipvc2_2/2_2prirn.htm
Vendor: Cisco
Supported Protocols: H.323
Platform: N/A
Description: This H.320 to H.323 Gateway is an older product, identical to the RADVISION OnLAN Gateway, but OEMed by Cisco and currently not supported any more, as it has been replaced by the 3526 Gateway. The 3525 is capable gatewaying 16 voice channels, or 8 128Kbps participants with H.261 video, or a combination of other rates for multiple BRI bonding. The default hardware does not provide audio/video transcoding, but there existed a hardware add-on for audio transcoding. Check the IP/VC Products page at
`http://www.cisco.com/univercd/cc/td/doc/product/ipvc/ipvc2_2/2_2mcurn.htm`

B. 5 Testing

> Product Name: CallGen323 (H.323 Call Generator)
Product URL: `http://www.openh323.org/code.html`
Vendor: Open Source
Supported Protocols: H.323
Platform: Any platform where you can compile the OpenH323 Library (Linux, Windows, FreeBSD, Solaris, etc.)
Description: Operational Experience: It can make and receive an exact number of calls, adjust the delay between each batch of calls and set the number of batches to repeat. It only produces signalling traffic (no audio data traffic). Support and development has now stopped and it requires special old version of a OpenH323 library to compile. No dynamic configuration is possible; once the program is started the client is configured using the command line options.

Overall Evaluation: It is a simple H.323 Call Generator. It is very customisable using a number of parameters. It is really useful in testing environments where servers need to be tested under stress. Drawbacks are static configuration, no dynamic management and limited support.

> Product Name: sipsak
Product URL: `http://sipsak.berlios.de/`
Vendor: iptel.org
Supported Protocols: SIP
Platform: N/A
Description: Free Diagnostic and Stress Utility. sipsak is a simple utility that can be used to test various functions of a SIP server. It includes proxy, registrar and digest authentication tests. It can also generate a load of SIP messages to stress a server.

> Product Name: SIPStone
Product URL: `http://www.sipstone.org`
Vendor: Columbia University and Ubiquity
Supported Protocols: SIP
Platform: N/A
Description: Currently, this is a draft about measuring SIP performance
`http://www.sipstone.com/. A measurement tool is available from Columbia University - see http://www.cs.columbia.edu/IRT/cinema/sipstone/`
See SIPstone mailing list for a discussion.

B. 6. Miscellaneous

> Product Name: OpenAM (H.323 Answering Machine)
Product URL: `http://www.openh323.org/code.html`
Vendor: Open Source
Supported Protocols: H.323
Platform: Any platform where you can compile the OpenH323 Library (Linux, Windows, FreeBSD, Solaris, etc.)
Description: Operational Experience: It can accept multiple connections simultaneously and runs

a user-defined program after each call, which can be used to automatically send the recorded message as a MIME-encoded e-mail attachment to a known e-mail address. If the recorded message is encoded using G.723.1 codec, it requires equipping with the PC with additional cards (Quicknet). No dynamic configuration is possible; once the program is started, the client is configured using the command line options.

Overall Evaluation: It is a simple answering machine using the H.323 protocol. It is really useful in unified messaging scenarios. It needs to operate in an environment where supplementary services are implemented. Drawbacks are static configuration and no dynamic management.

> Product Name: Yxa
Product URL: `http://www.stacken.kth.se/projekt/yxa/`
Vendor: Open Source
Supported Protocols: SIP
Platform: Any platform where you can use Erlang programming language (Linux, Windows, FreeBSD, Solaris, etc.)
Description: Operational Experience: Yxa is a bunch of library-like functions for receiving, processing and sending SIP messages, and a couple of small programs that can do various things. The operational experience is poor right now because Yxa is not widely known (even if it is gaining popularity). It is a SIP server open source software written in Erlang (a programming language from Ericsson with open source releases). Basically, it has built-in some software for performing a number of functionalities. One of the main applications is the incoming proxy; it can handle REGISTER requests and authorise different users to make calls to different classes of PSTN numbers. It can proxy requests from UACs to other parts of the Yxa system, relay requests to remote servers/domains Routing features, do ENUM lookups of things that looks like E.164 phone numbers and do lookup addresses in LDAP.

Overall Evaluation: The project just produced some nice results. No release has been made yet and downloading is done only through CVS. A mailing list is available for questions and inquiries.

Glossary

CPL
See Call Processing Language.

Call Processing Language
An XML application to define rules for call processing (e.g., to route all calls to a cell phone during lunchtime). Decisions can be made upon the calling party number or the called party number, date or time - just to name a few, while actions usually include rejection or redirections of calls.

Extensible Markup Language
A standard methodology with formal syntax for adding information to a document relating to its structure and/or content by applying identifiers for elements of information in a neutral way, stored in a neutral form, independent of systems, devices and applications.

Gatekeeper
In the H.323 world, the gatekeeper provides several important functions. First, it controls access to the network, allowing or denying calls and controlling the bandwidth of a call. Second, it helps with address resolution, making possible e-mail-type names for end users, and converting these into the appropriate network addresses. A gatekeeper also handles call tracking and billing and call signalling.

GK
See Gatekeeper.

Gateway
A gateway is a communication instance that translates (call) data between different networks (e.g., IP and PSTN) or protocols. A signalling gateway translates between two or more signalling protocols (like H.323 or SIP), while a media gateway usually performs transcoding of media streams (e.g., G.711 to GSM).
Of course a gateway may as well combine all functionality described above.

GW
See Gateway.

H.323
ITU standard for videoconferencing over packet-switched networks such as Internet.

Media Gateway Control Protocol
A protocol for IP Telephony that enables a calling party with a PSTN phone number to locate the destination device and establish a session.

MGCP
See Media Gateway Control Protocol.

PBX
See Private Branch eXchange.

PSTN
See Public Switched Telephone Network.

Private Branch eXchange
A device used by organisations to allow a single access number to offer multiple lines to outside calling parties and to allow internal staff to share a range of external lines.

Public Switched Telephone Network
The worldwide voice telephone network.

QoS
See Quality of Service.

Quality of Service
Measure of performance for a transmission system that reflects its transmission quality and service availability.

Real-time Protocol
RTP is designed to provide end-to-end network transport functions for applications transmitting real-time data, such as audio, video or simulation data over multicast or unicast network services.

RTP
See Real Time Protocol.

SIP
See Session Initiation Protocol.

Session Initiation Protocol
IETF standard for session initiation in multi-purpose communication systems.

Voice over IP
The transmission of voice over data networks that use the Internet Protocol (IP).

VoIP
See Voice over IP.

XML
See Extensible Markup Language.