Investigating Call Control using MGCP in conjunction with SIP and H.323

THESIS

Submitted in fulfilment of the requirements

for the degree of

MASTER OF SCIENCE

of Rhodes University

by

Ashley Jacobs

November 2004

Abstract

Telephony used to mean using a telephone to call another telephone on the Public Switched Telephone Network (PSTN), and data networks were used purely to allow computers to communicate. However, with the advent of the Internet, telephony services have been extended to run on data networks. Telephone calls within the IP network are known as Voice over IP. These calls are carried by a number of protocols, with the most popular ones currently being Session Initiation Protocol (SIP) and H.323. Calls can be made from the IP network to the PSTN and vice versa through the use of a gateway. The gateway translates the packets from the IP network to circuits on the PSTN and vice versa to facilitate calls between the two networks.

Gateways have evolved and are now split into two entities using the master/slave architecture. The master is an intelligent Media Gateway Controller (MGC) that handles the call control and signalling. The slave is a "dumb" Media Gateway (MG) that handles the translation of the media. The current gateway control protocols in use are Megaco/H.248, MGCP and Skinny. These protocols have proved themselves on the edge of the network. Furthermore, since they communicate with the call signalling VoIP protocols as well as the PSTN, they have to be the lingua franca between the two networks.

Within the VoIP network, the numbers of call signalling protocols make it difficult to communicate with each other and to create services. This research investigates the use of Gateway Control Protocols as the lowest common denominator between the call signalling protocols SIP and H.323. More specifically, it uses MGCP to investigate service creation. It also considers the use of MGCP as a protocol translator between SIP and H.323.

A service was created using MGCP to allow H.323 endpoints to send Short Message Service (SMS) messages. This service was then extended with minimal effort to SIP endpoints. This service investigated MGCP’s ability to handle call control from the H.323 and SIP endpoints. An MGC was then successfully used to perform as a protocol translator between SIP and H.323.

Acknowledgements

Throughout this project, Prof. Peter Clayton, my supervisor, has provided me with a great deal of advice and support. I would to thank him for his knowledgeable input, guidance, patience and enthusiasm throughout this research.

I would also like to thank my fellow postgraduate students and the Computer Science Department staff. In particular, I would like to acknowledge the valuable assistance from Jason Penton, Alfredo Terzoli and Ming Hsieh for his advice and being the springboard for my ideas.

I would also like to thank my friends and family for their encouragement. A special thanks to my parents, who encouraged me to overcome difficulty and stay focused.

I acknowledge the financial support of the National Research Foundation and the Computer Science Department.

Contents

Figures and Tables

Chapter 1

Introduction

Telephony used to mean using a telephone to call another telephone on the Public Switched Telephone Network (PSTN), and data networks were used purely to allow computers to communicate. However, with the advent of the Internet, telephony services have been extended to run on data networks and telephone networks are being used to provide data services. The net result is a large communication network.

On the PSTN, devices communicate by what is known as circuit switching. This refers to the days of analogue telephony when an end-to-end electrical circuit was formed to connect telephone sets at either end. With the arrival of digital switching and time division multiplexing, the underlying network is now divided into channels and an end-to-end connection is established by interconnecting these channels to form a media stream. [Bayer, 2001]

On data networks, to send information from one device to another, the information is put into packets and sent along the network to the destination. This type of network is known as a packet switched network. The most widely used protocol on this network is Internet Protocol or IP. Although other types of packet networks exist, Asynchronous Transfer Mode (ATM), Frame Relay and Packet Cable to name a few, the focus of this research is on packet networks which use IP as their underlying protocol. Telephone services implemented in an IP network are known as Voice over IP or IP telephony. To extend these telephony services from the IP network to the PSTN posses a problem because of the different networks involved, in other words from packet switched to circuit switched or vice versa. This is where gateways come in.

1.1 Introduction to Gateway Control Protocols

A gateway is a network element that connects two dissimilar networks [Ohrtman, 2003]. It processes the call signalling or call control information as well as switches the media from packet to circuit or vice versa. To understand how this works we look at the protocols used on the PSTN and how these are translated for the IP network.

On the PSTN, the signalling protocol used most commonly is Signalling System 7 (SS7) [Russel, 2000]. SS7 is not a single protocol per se, but rather it is a suite of protocols. To establish, maintain or tear down calls, the applicable SS7 signalling protocol is ISDN User Part (ISUP) [ITU-T, 1997]. When two people want to communicate on the PSTN, ISUP is used to establish a call between two parties after which they may talk to each other. When speech is transported from one telephone to another it is referred to as a media stream. The media stream on the PSTN is carried along the telephone lines using Time Division Multiplexing (TDM).

On the IP network, there are a number of protocols used for telephony. The most popular VoIP protocols currently are H.323 [ITU-T, 1998] and Session Initiation Protocol (SIP) [Handley *et al*, 1999]. The media transport mechanism used by these protocols is the Real-time Transport Protocol (RTP) [Douskalis, 2000]. A detailed explanation of these protocols will be done later. For now it is sufficient to understand that the SS7 signalling messages and media have to be translated into these formats to make a call from the PSTN to the IP network and vice versa.

Making a call from the PSTN to the IP network

Traditionally, a single gateway translated calls from the PSTN to the IP network and vice versa. The PSTN signalling carried by the ISUP messages as well as the PSTN media carried by the TDM trunks are translated by the gateway into a specific VoIP signalling protocol and RTP media stream for the IP network as shown in Figure 1.1.

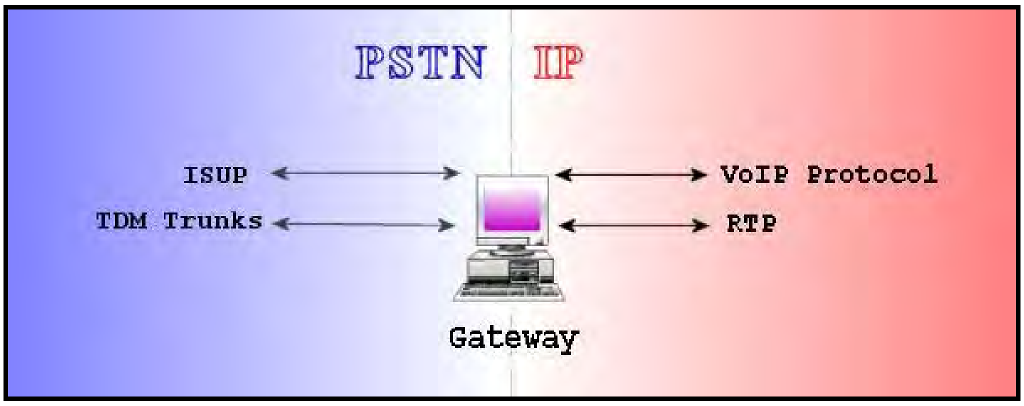


Figure 1.1 Gateway between PSTN and IP network

As networks grew, so did the number of VoIP protocols. As a result, the single gateway approach, which did not scale proportionately, made way for the decomposed gateway architecture. This approach splits a gateway into a signalling entity and a media entity. These entities are further decomposed into a functional gateway that translates media or signal and a controlling entity that manages this gateway known as an agent.

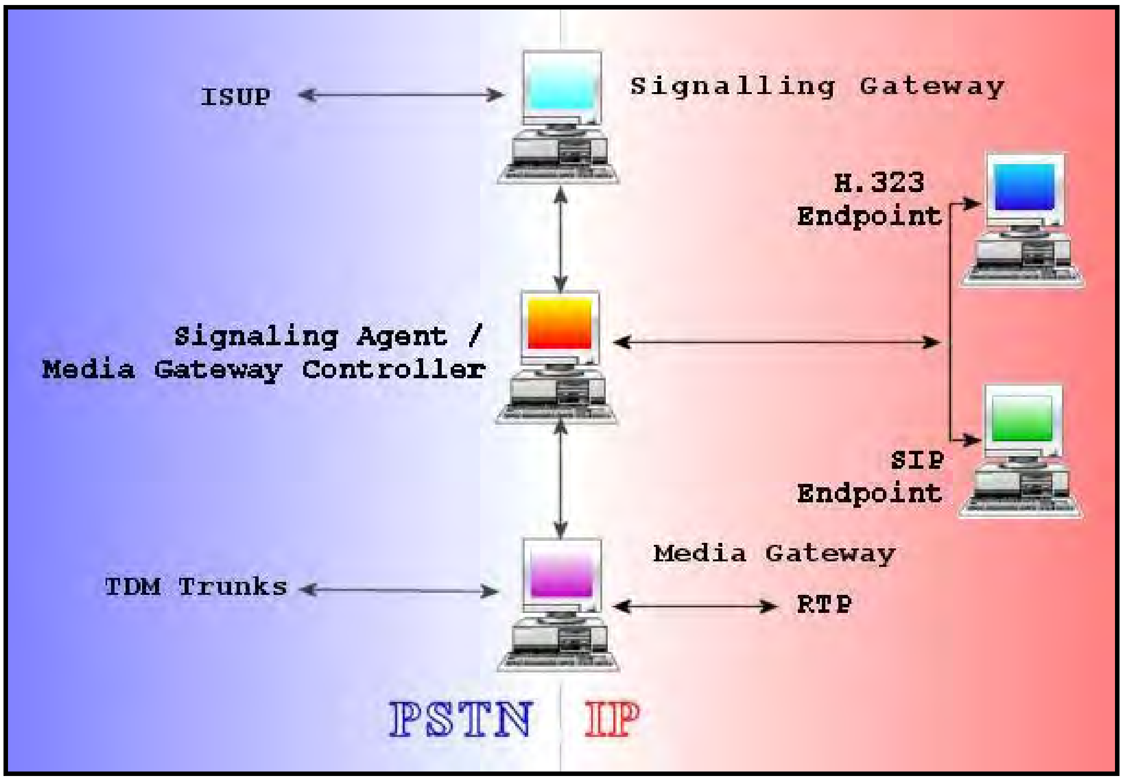


Figure 1.2 Decomposed gateway architecture

Thus, the gateway in Figure 1.1 can be decomposed into a Signalling Gateway (SG) and a Media Gateway (MG). The SG would be controlled by a Signalling Agent (SA) and the MG would be controlled by a Media Gateway Controller (MGC) or Call Agent (CA). Figure 1.2 shows the decomposed gateway architecture. The SG translates the signalling messages into VoIP signalling messages and the MG translates the media from the TDM trunks into RTP streams on the IP network.

The SG, SA and MGC can be combined into one box [Hersent *et al*, 2000]. The resulting architecture is one that has two boxes. One box is the MG and the other is the MGC. In this architecture the MG translates the media while the MGC handles the signalling and controls the MG. The MG is controlled using a Gateway Control Protocol (GCP) like Media Gateway Control Protocol (MGCP)[1]. Figure 1.3 shows a PSTN to IP network call using the MGCP architecture.

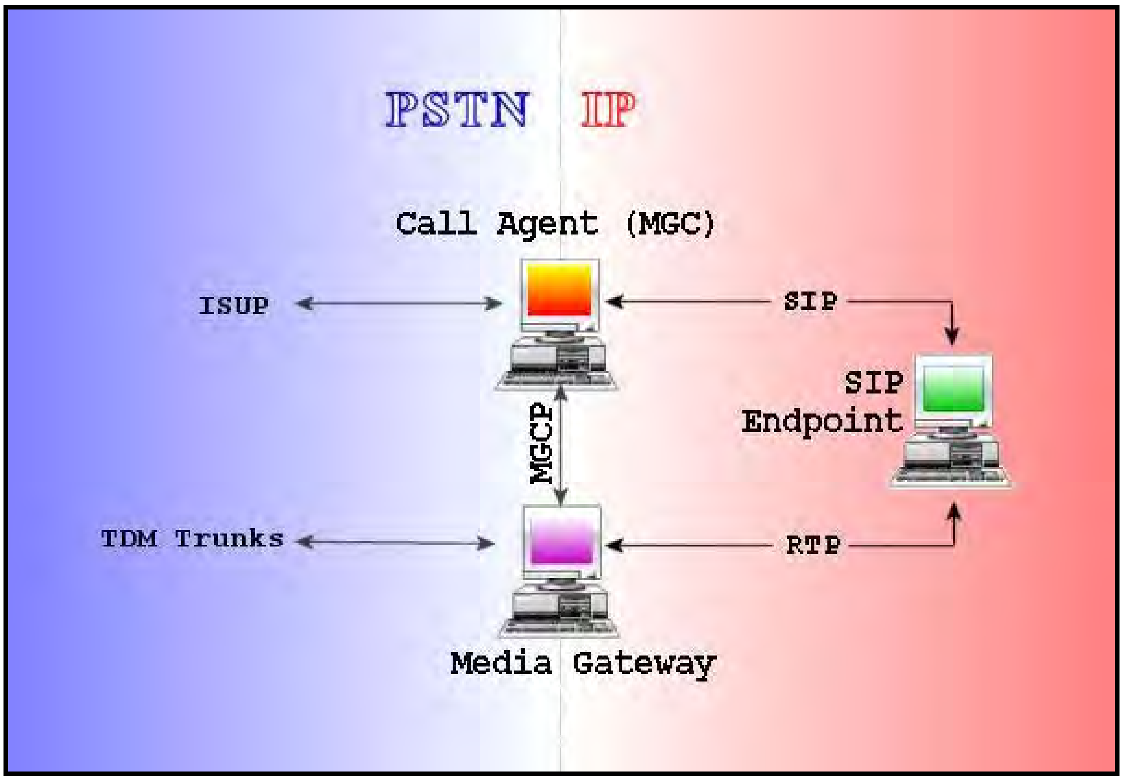


Figure 1.3 MGCP architecture

[1] MGCP assumes a call control architecture where the call control "intelligence" is outside the gateways and handled by external call control elements. MGCP assumes that these call control elements, or Call Agents, will synchronize with each other to send coherent commands to the gateways under their control. [Arango *et al*, 1999]

The relationship between the CA and the MG is a master/slave relationship. This design allows the CA to control one or many gateways as the need arises. The result is a scalable architecture that can process anywhere from tens to thousands of calls per minute. Interfacing between the PSTN and the IP network is one of the functions of a gateway, whose main function is to interface between heterogeneous networks. Having explained the basic function of a gateway and a gateway control protocol, let us look at the origin of gateway control protocols.

1.1.1 The origin of Gateway Control Protocols

The first protocol to emerge was the result of a drive by cable operators who wanted to become Competitive Local Exchange Carriers (CLEC), by using IP on top of their Hybrid Fibre/Cable infrastructure. [Hersent *et al*, 2000]. In May 1998, at a PacketCable^{TM} meeting, Cisco and Telcordia[2] proposed Simple Gateway Control Protocol (SGCP), as a better-suited and more cost effective alternative to H.323. Around the same time, an all IP backbone communications company, Level 3, had begun talks with Xcom. Xcom had developed a technology to allow a softswitch to communicate with a modem bank.[3] In April 1998, Level 3 bought out Xcom and renamed its newly acquired protocol Internet Protocol Device Control (IPDC). IPDC addressed roughly the same requirements as SGCP, but with a different approach to transport [Hersent *et al*, 2000].

Shortly thereafter, Telcordia and Level3 worked together in merging these two protocols into one. The resulting protocol was called Media Gateway Control Protocol. The MGCP effort was essentially a private effort, from a small group of companies and researchers. The MGCP proponents took their protocol to the Internet Engineering Task Force (IETF) intending to standardise MGCP. The IETF agreed to publish a version of MGCP as an ‘informational RFC’, which is not a standard, but publicly documents a de facto standard [Rosen, 2001]. There are a number of MGCP implementations and one major organisation, CableLabs, has adopted MGCP as its protocol for the packet over cable effort that is part of PacketCable [Rosen, 2001].

[2] Telcordia was formerly known as Bellcore
[3] A modem bank is the hardware on the ISP side that dial up Internet users dial into.

Independently of MGCP, the ITU began working on a similar protocol to MGCP. Lucent Technologies contributed to this effort by offering the Media Device Control Protocol (MDCP). After this, the ITU began a historic collaboration with the IETF using MDCP and MGCP as the basis. Although some of the collaborators would have liked to see the result as the next version of MGCP, it was published as a new protocol H.248 / Megaco [Rosen, 2001]. It is available as RFC 3015 [Green *et al.*, 2000] from the IETF and ITU-T recommendation H.248 [ITU-T, 2000].

Megaco is a newer, more flexible and extensible protocol than MGCP. However, MGCP's earlier deployment led to the availability of open source Application Programming Interfaces (APIs) at the beginning of this research. Similar packages for Megaco were not available, therefore the author chose to investigate MGCP. Since Megaco is conceptually an evolution of MGCP [RADVision, 2002], the findings of the research would be applicable to Megaco.

1.2 Why investigate Gateway Control Protocols?

As long as there is legacy equipment in use, the need to interface with it will require the use of gateways. In addition, the existence of different voice protocols within the IP network necessitates the use of interworking gateways. Therefore in an IP network with X number of VoIP protocols, X number of gateways are required to interface with the PSTN. Furthermore, interworking gateways are required to allow communication between these VoIP protocols. The result is that VoIP networks are becoming complicated and difficult to administer.

Gateway Control Protocols offer an approach to managing gateways that is easy to administer and highly scalable. Through the use of the Master/Slave architecture, a single MGC can control a number of media gateways greatly reducing the complexity of administering these gateways. This architecture, which is similar to the SS7 architecture, simplifies the monitoring and billing of calls. For these reasons GCPs receive support from telecommunication companies. Additionally, GCPs provide fine-grained control over media resources, and are not constrained by the limitations of signalling protocols such as SIP and H.323. When a signalling protocol is used to control a media server, the media server must act as a party in a call, rather than as a simple media resource. Consequently, the media server is forced into an “unnatural” and complex relationship with the media controller. [RADVision, 2002]. It is with this interest in GCPs, that an investigation into their capabilities and their role in telephony networks was thought to be worth investigating.

H.323 and SIP have gateways to interface with the PSTN. Adding a GCP potentially simplifies this process, but what is the overhead of adding another protocol to the network? What are the potential benefits and drawbacks and when is it appropriate to use a GCP? Furthermore, can GCPs offer any other benefits or functionality? These are some of the issues that this research sets out to address.

Lastly, as has been explained earlier, the various VoIP protocols use RTP to carry their media. Therefore their fundamental difference is in their signalling. Since MGCs set up calls between these VoIP protocols and the PSTN, perhaps MGCs could be used to setup calls between the VoIP protocols as well.

1.3 Goals of this project

The Telecommunications industry has spent a number of years researching IP telephony, but the research is far from over. As the search for an IP protocol that satisfies the demands of an emerging IP telephony market continues, more and more protocols are appearing. These protocols, that are competitive rather than complementary, add to the complexity of the VoIP arena, and yet do not satisfy all the needs completely. In addition, older VoIP signalling protocols cannot simply be replaced with newer ones. The result is that a number of VoIP protocols will continue to co-exist for some time. Hence the question being asked is not, “Which protocol is best?” but rather, "Which services do we want to deploy, and which VoIP protocols best support those services?" [Cisco, 2002]. Currently the two main signalling protocols are H.323 and SIP. In order for these protocols to make VoIP as appealing as the PSTN, they need to offer all the services that the PSTN does and more.

GCPs are complementary to the VoIP signalling protocols. Thus, the main aim of this research is to investigate GCPs through the creation of services, using MGCP. To begin with, MGCP is deployed on the Rhodes University Computer Science Department (RUCSD) VoIP network. Services are then developed to investigate the use of MGCP for service creation.

Services that are deployed for IP telephony are usually protocol specific. This means that if a service is developed for H.323 endpoints, SIP endpoints cannot use it unless it is remade specifically for SIP endpoints. However if a service is developed using MGCP, perhaps using an MGC that communicates with both H.323 and SIP can reduce the duplication of effort. This use of MGCP to create a generic interface will also be investigated.

SIP endpoints communicate with H.323 endpoints through a gateway, which translates the signalling between the two protocols. The gateway that does this kind of translating is known as a SIP-H.323 Interworking Function (IWF) [Schulzrinne & Agboh, 2004]. Once an MGC has been developed that communicates with H.323 and SIP endpoints, an investigation into the use of the MGC as a SIP-H.323 IWF will be conducted.

MGCP has already proven itself on the edge of the network. Therefore, the aim of this thesis is to test the limits of MGCP within the IP network. The focus of this study will be on call control and not media translation. This will be done through the creation of services and an IWF. Call control in this context refers to the signalling that is required to setup and tear down calls.

1.4 Structure of the thesis

- ❖ Chapter 2 introduces the protocols involved in this research, namely MGCP, SIP and H.323 with particular reference to call control.

- In Chapter 3, an SMS Service is developed using MGCP. This service is developed first for H.323 and then extended to SIP. This chapter describes the development and deployment of the two services.

- In Chapter 4, the issues that arose from the implementation of the H.323 and SIP SMS services are discussed. The interworking issues between SIP and MGCP as well as between H.323 and MGCP are explored. Lastly the two SMS services are compared.

- In Chapter 5 a SIP/H.323 MGC is developed and used to investigate the possibility of using MGCP to facilitate the interworking between SIP and H.323.

- In Chapter 6, closing remarks are made and a conclusion is drawn.

Chapter 2

Protocols Overview

There are four major IP-based call processing protocols currently being deployed in the industry: H.323, SIP, MGCP and Megaco [Black, 2001]. This research focuses on MGCP and its interaction with SIP and H.323. This chapter begins by describing how the three fit together conceptually. It will then introduce MGCP, its model and commands. The chapter will then introduce H.323 and SIP. The sections on SIP and H.323 are to give the reader an introduction to those parts of the protocols that are relevant to this research, and are not meant to give a complete introduction to them. Chapter one introduced the basic function of a gateway and a GCP. Before describing MGCP in detail, an overview of how MGCP, SIP and H.323 fit together with regard to their development will be in order.

H.323 and SIP are competing VoIP signalling protocols. The International Telecommunications Union-Telecommunications (ITU-T) developed H.323 version 1 (H.323v1) based on the H.320 standard for videoconferencing over ISDN. Its aim was to create very smart terminals that could invoke and create many services that are not supported by the network. H.323v1 began development in May 1995 and was approved in June 1996. After some minor changes, H.323v2 [ITU-T, 1998a] was approved in February 1998. Since then versions 3 and 4 have been approved and the current version of H.323 is five. For this research, all references to H.323 will be to version 2.

SIP on the other hand is an Internet Engineering Task Force (IETF) protocol. It started in 1996 for the purpose of initiating multicast sessions on the Internet and was later adapted to handle IP telephony and even later Instant Messaging and Presence [Levin, 2001]. SIP was designed to be a simple protocol that could be used in very simple devices and use the intelligence of the network in a similar model to services on the Internet.

As these two protocols compete for dominance, many of the modern VoIP networks have either one or both protocols. GCPs like MGCP allow H.323 and SIP to communicate with legacy equipment as well as extending their functionality. Although MGCP is primarily a gateway control protocol, it is not limited to this function and is also capable of simplifying heterogeneous protocols in a VoIP network, by adding commonality. This aspect of the MGCP model will be explored in chapters 3 and 4.

Figure 2.1 shows the MGCP architecture. The MGC/CA communicates with the SIP and H.323 endpoints using their native protocols and with the MG using MGCP messages.

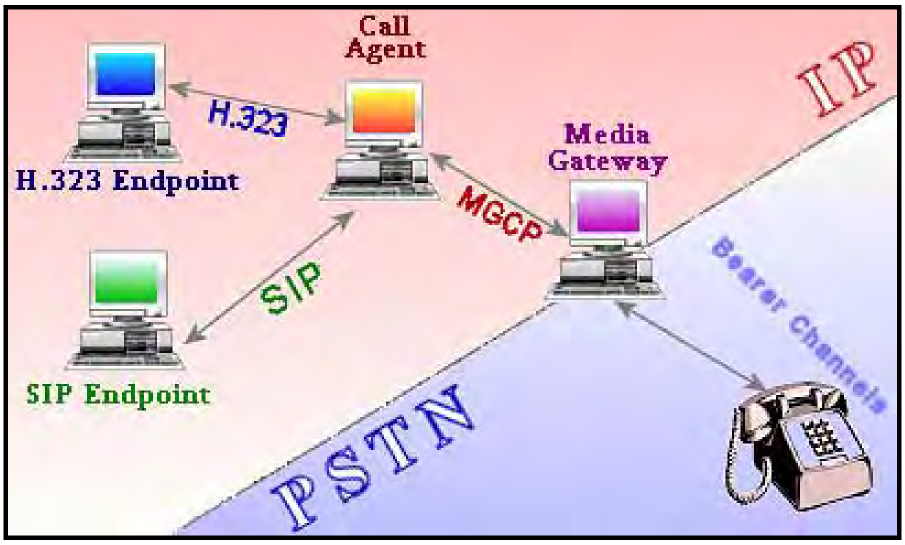


Figure 2.1 MGCP Architecture

2.1 Media Gateway Control Protocol

This introduction to MGCP is based mostly on RFC 2705, which specifies the requirements of MGCP version 1.0. MGCP assumes a connection model where the basic constructs are endpoints and connections. Endpoints are sources and/or sinks of data and can be physical or virtual [Arango *et al*, 1999]. Creation of physical endpoints requires hardware installation, while creation of virtual endpoints can be done by software [Arango *et al*, 1999]. Connections may be point-to-point or multipoint. Point-to-point connections are setup between two endpoints that wish to communicate exclusively. Multipoint connections are three or more endpoints that are engaged in a multipoint session. If two endpoints wish to communicate through a gateway, a connection has to be established between the first endpoint and the gateway and another connection between the gateway and the second endpoint. Once these two connections are linked, allowing media to flow between the endpoints, they are considered to be in a call. Please note that a call refers to both a media stream that is established between the endpoints and all the associated control information [Bayer, 2001]

2.1.1 MGCP Endpoints

Since endpoints may be implemented in a number of ways, MGCP simply assumes that media gateways support collections of endpoints. The type of endpoint determines its functionality. The following basic endpoint types have been identified:

- ❖ Digital channel (DS-0)
- ❖ Analog line
- ❖ Announcement server access point
- ❖ Interactive voice response access point
- ❖ Conference bridge access point
- ❖ Packet relay
- ❖ ATM trunk-side interface

Digital channels

Digital channels provide a 64 Kbps service. Such channels are found in trunk and ISDN interfaces and referred to as a DS-0 channel. Typically, a DS-0 will carry voice that is encoded in G.711 mu-law or G.711 A-law. In certain cases, the DS-0 may carry signalling instead of voice. Under such circumstances, the information has to be passed on to the call agent for processing [Collins, 2001]

Analog Lines

Analog lines typically interface with traditional telephone lines to provide a service to traditional telephones. They may also provide a service to gateways allowing them to send and receive analog calls. If the media gateway also supports a Network Access Server (NAS) service, the gateway can receive audio-encoded data from a modem and convert them into data packets.

Announcement server access point

An announcement server endpoint provides access to an announcement service. When requested by a CA, the announcement server will play a specific announcement. Since the server is not required to listen, the connection to the server is usually one way and one at a time. If more than one endpoint is connected to the server, then the same announcements would be played simultaneously over all the connections.

Interactive Voice Response (IVR) access point

An IVR endpoint provides access to an IVR service. When requested by a CA, the IVR server will play announcements and tones, and will listen to responses, such as DTMF input or voice messages, from the user.

Conference bridge access point

A conference bridge endpoint is used to provide access to a specific conference, where media streams from multiple callers can be mixed and provided to the callers.

Packet relay

A packet relay endpoint is a specific form of conference bridge that typically supports only two connections. This type of endpoint uses packet on both the connections.

Asynchronous Transfer Mode (ATM) trunk-side interface

An ATM trunk-side endpoint corresponds to the termination of an ATM truck.

Wiretap access point

This endpoint provides access to another endpoint to listen to media sent or received from that endpoint. It supports one connection at a time in half duplex mode.

2.1.2 MGCP Events and Packages

When you lift a telephone receiver, the telephone is said to go off-hook. Under MGCP, when endpoints go off-hook, it is known as an event. Events correspond to associated signals, which are grouped into packages [Black, 2001]. MGCP defines ten such packages. Figure 2.2 [Black, 2001] displays the conceptual view of events and packages.

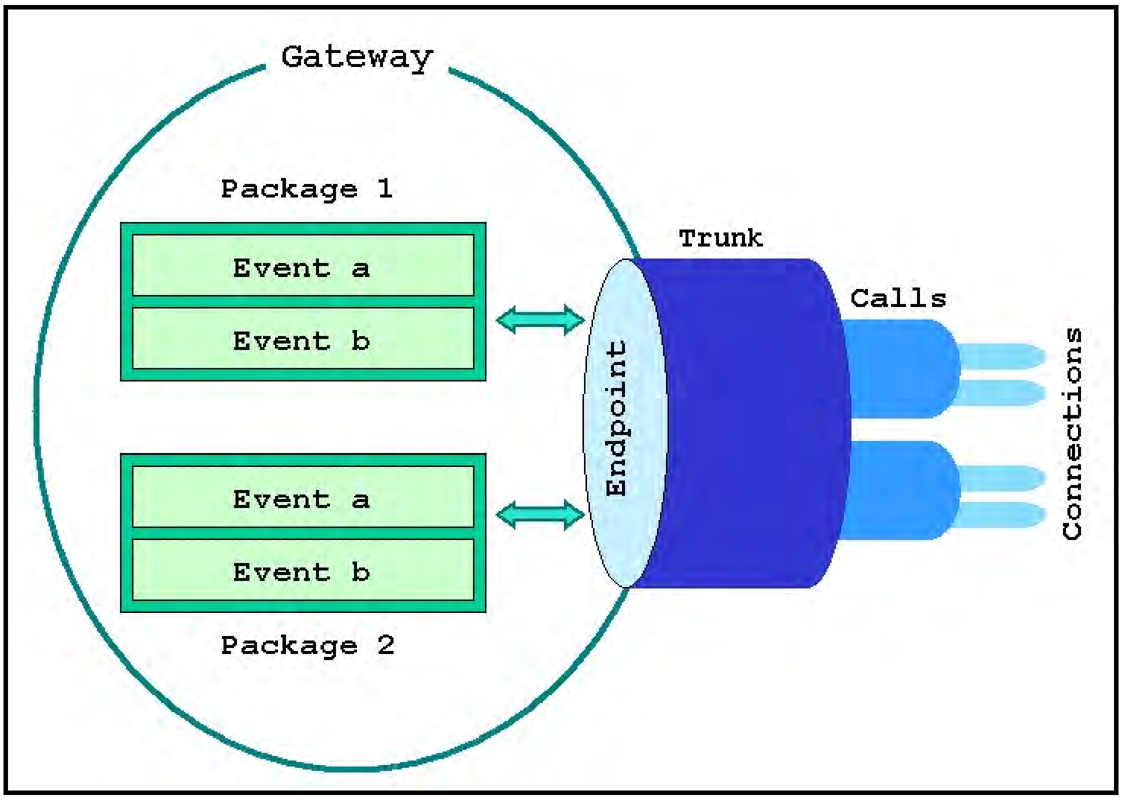


Figure 2.2 MGCP Events and Packages

The ten packages are:

Generic Media: describes events relating to media like detecting tones for example modem detected, fax tone detected and network congestion tone.

DTMF: describes telephone DTMF signals for example telephone numbers dialled.

MF: describes trunk MF signals

***Trunk**:* describe events that occur on a trunk circuit

***Line**:* describes line tones for example dial tone and alerting tone

***Handset**:* is an extension of the line package and is used by gateways capable of emulating handsets.

***RTP**:* describes RTP payload transmission like codec change or packet loss exceeded.

***Network Access Server**:* deals with status and diagnostic events like authorisation and status of a packet.

***Announcement Server**:* defines events related to an announcement server.

***Script**:* defines how to load scripts of type Java, Pearl, TCL and XML.

2.1.3 MGCP Commands

MGCP has a small command set, consisting of nine commands. Some commands are sent from the call agent to the media gateway and the rest from the media gateway to the call agent. The commands are text based and use another text-based protocol called Session Description Protocol (SDP) to describe resources used in connections. For example a connection between a gateway and an endpoint could describe the ports to be used and media capabilities using SDP. SDP is an IETF protocol and is specified as RFC 2327.

An MGCP command consists of the command name, a transaction identifier, the endpoint for which the command is intended or from which it originates and the protocol version (MGCP1.0). The command format is as follows:

```
MGCPCommand TransactionId EndpointID MGCP 1.0
```

Details of MGCP’s nine messages are described below. The four letters in brackets are the API commands for the message.

EndpointConfiguration (EPCF)
A call agent issues this command to a gateway to inform it about the coding characteristics of the line side of one or more endpoints. At the moment the only characteristic is whether the encoding is A-law or mu-law.

CreateConnection (CRCX)
A call agent issues this command to a gateway to create a connection that terminates in an "endpoint" inside the gateway.

ModifyConnection (MDCX)
A call agent issues this command to the gateway to change some characteristics of an existing connection. It is quite often used to convey information about the other end of the connection.

DeleteConnection (DLCX)
A call agent issues this command to instruct the gateway to terminate a connection. A gateway may also send this command to a call agent when it detects an event, like the loss of line-side connectivity.

NotificationRequest (RQNT)
A call agent issues this command to the gateway to request notification when the gateway detects a specific event.

Notify (NTFY)
A gateway issues this command to a call agent to notify certain events and may be in response to a *NotificationRequest* message.

AuditEndpoint (AUEP)
A call agent issues this command to the gateway to request information about a certain endpoint.

AuditConnection (AUCX)
A call agent issues this command to the gateway to query a specific connection.

RestartInProgress (RSIP)
A gateway issues this command to inform a call agent, that the gateway or a group of endpoints are being taken out or being placed back in service.

The messages above constitute the MGCP message set. Within these messages certain parameters can be passed with the message. These include bearer information, connection modes, connection IDs, connection descriptors and connection options to name a few such parameters. Next, we look at the MGCP responses to these messages.

2.1.4 MGCP Responses

Every MGCP command requires a response. A response consists of a response line and may also contain a number of response parameters. The response line contains a return code, a transaction identifier and is optionally followed by a commentary or reason phase [Collins, 2001]. The transaction identifier is used to correlate responses to the commands that triggered them. The command format is as follows:

```
ReturnCode TransactionId Commentary
```

The return code is an integer that falls into one of four categories:

100 – 199 - temporary responses to be followed by concluding response

200 – 299 – command completed successfully

400 – 499 – Failure caused by transient error

500 – 599 – Failure caused by permanent error

2.1.5 MGCP Call Scenario

Having defined the MGCP requests and responses, we look at a simple call scenario to exemplify some of these concepts. The following scenario is based on the VOCAL[4] MGCP example. In this scenario we assume that one call agent controls two gateways and each one has an endpoint attached to it. We also assume that the call agent has received external call signalling to initiate a call between the two gateways, the call setup will be as follows.

1. The Call Agent (CA) sends Media Gateway Alpha (MGA) a *CreateConnection* command to instruct it to create a connection on an endpoint. This may be a specific endpoint or left to the gateway to select one.
2. MGA creates the connection and responds to the CA. The response contains the LocalConnectionDescriptor. The LocalConnectionDescriptor contains the session description, which is the IP address, port number, media type and coding.

[4] VOCAL stands for Vovida Open Communications Application Library

3. The CA then sends Media Gateway Beta (MGB) a *CreateConnection* Command, but also includes the RemoteConnectionDescriptor, which corresponds to the session description of MGA.
4. MGB creates the connection and responds to the CA with its LocalConnectionDescriptor.
5. The CA then sends MGA a *ModifyConnection* command containing the RemoteConnectionDescriptor. The RemoteConnectionDescriptor contains the session description for MGB.
6. MGA responds to the CA.

Now both MGA and MGB know each other's media requirements and the media can now be exchanged. Figure 2.3 shows the message flow between the gateways and call agent.

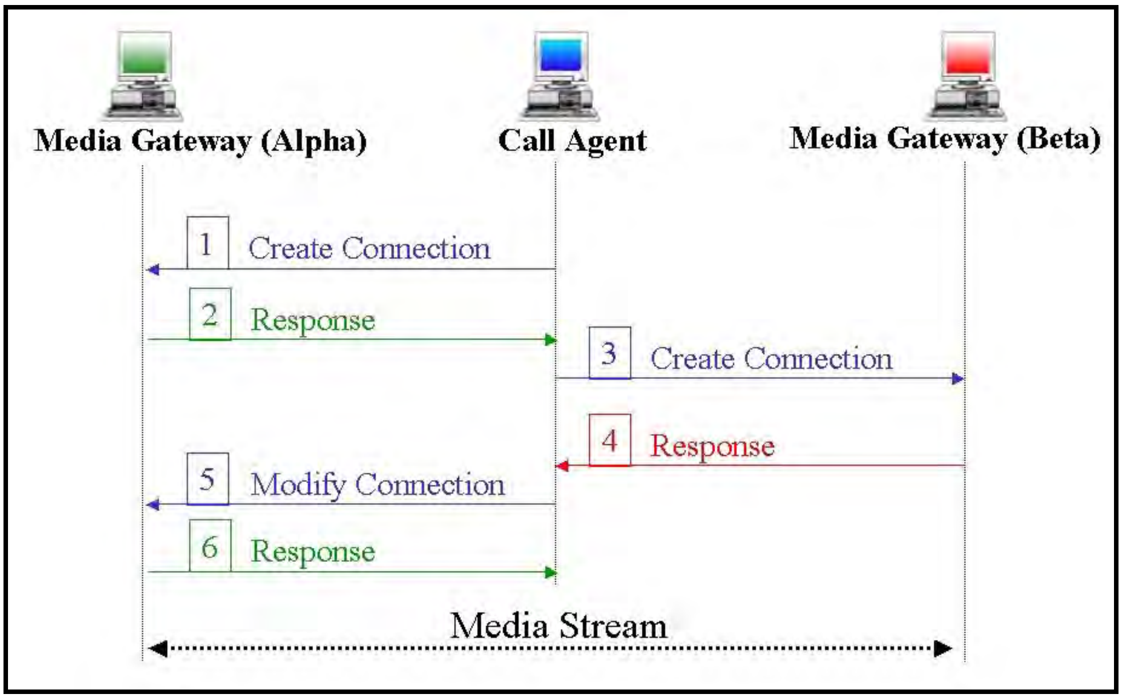


Figure 2.3 MGCP call between two gateways

At this point a more in-depth look at the parameters passed with the messages is in order. Table 2.1 shows the details of the six MGCP messages used to setup the call between the two gateways and explains each message.

Table 2.1 MGCP call setup between two gateways

	Command	Explanation

1	`CRCX 1000`
	`EPAlpha@MGA.ict.ru.ac.za`
	`MGCP 1.0`
	`C: 007`
	`M: recvonly`
		The CA sends MGA a CRCX message to create a connection on endpoint EPAlpha. The TransactionId is 1000. The CallId is 007. The mode is receive only since the EPAlpha does not know the session description for the remote end.

2	`200 1000 OK`
	`I: AlphaOne`
	`v=0`
	`c=IN IP4 146.231.121.141`
	`m=audio 22000 RTP/AVP 0`
		MGA responds to the CRCX message with the same TransactionId(1000). The ConnectionId is AlphaOne. EPAlpha has an IP address of 146.231.121.141. It supports RTP payload 0. This corresponds to codec G.711 mu-law

3	`CRCX 2000`
	`EPBeta@MGB.ict.ru.ac.za`
	`MGCP 1.0`
	`C: 007`
	`M: sendrecv`
	`v=0`
	`c=IN IP4 146.231.121.141`
	`m=audio 22000 RTP/AVP 0`
		The CA sends MGB a CRCX message to create a connection on endpoint EPBeta. The TransactionId is 2000. The same CallId (007) is used. The CA includes the session description for EPAlpha. Since EPBeta now has this information, the mode is set to sendrecv.

4	`200 2000 OK`
	`I: BetaOne`
	`v=0`
	`c=IN IP4 146.231.123.22`
	`m=audio 22000 RTP/AVP 0`
		MGB responds to the CRCX message with the same TransactionId(2000). The ConnectionId is BetaOne. EPBeta has an IP address of 146.231.123.22. It supports RTP payload 0. This corresponds to codec G.711 mu-law

5	`MDCX 3000`
	`EPAlpha@MGA.ict.ru.ac.za`
	`MGCP 1.0`
		The CA sends MGA a MDCX message with the remote session description.

	`I=AlphaOne`
	`M=sendrecv`
	`v=0`
	`c=IN IP4 146.231.123.22`
	`m=audio 22000 RTP/AVP 0`
		The TransactionId is 3000 and the ConnectionId id AlphaOne. Now that EPAlpha has EPBeta’s IP, port and media information, the mode is set to sendrecv.

6	`200 3000 OK`
	`I=AlphaOne`
		MGA responds positively to the CA.

2.2 Session Initiation Protocol (SIP)

This introduction into SIP is by no means complete, as a full introduction is beyond the scope of this research. Instead this section aims to introduce the basic concepts of SIP, with particular reference to call signalling, to equip the reader with adequate knowledge of SIP for later chapters.

Protocol Overview

Session Initiation Protocol, or SIP as it is abbreviated to, is an IETF protocol. It was designed to be a part of the IETF multimedia data and control architecture. As such, SIP is used in conjunction with other IETF protocols, such as Session Description Protocol (SDP), Session Announcement Protocol (SAP) and Real-Time Streaming Protocol (RTSP) [Collins, 2001]. SIP is defined in RFC2543 (Request For Comments 2543) [Handley *et al*, 1999].

2.2.1 SIP Architecture

SIP is a call signalling protocol. It sets up, modifies and tears down multimedia sessions between users. SIP does not define any specific transport protocol to carry the session traffic, but more often than not, RTP is used. SIP calls involve call signalling and media. SIP separates the call signalling from the media. For example, when two endpoints are in a call, the signalling messages may go through a number of entities to reach the other endpoint, but the media will take a more direct path. One of SIP’s advanced features is its support of mobile users [Black, 2001]. A user may register their location or a number of locations with a server. Calls to them will then be routed to the location(s) they are registered at. SIP’s novel approach to mobility is keyed to the individual and not their communications equipment.

SIP supports five steps in establishing and terminating multimedia communications:

- User location: determination of the end system to be used for communication;
- User capabilities: determination of the media and media parameters to be used;
- User availability: determination of the willingness of the called party to engage in communications;
- Call setup: establishment of call parameters at both called and calling party;
- Call handling: including transfer and termination of calls.

Before proceeding further let us familiarise ourselves with some terms and concepts as defined in RFC 2543.

2.2.2 SIP Terminology

- ❖ *Invitation*: A request sent to a user (or service) requesting participation in a session.
- ❖ *Caller*: Party that initiates a call.
- ❖ *Callee*: Party that is called.
- ❖ *User Agent*: An application that contains both a User Agent Server (UAS) and a User Agent Client (UAC), also known as a SIP User Agent. SIP User Agents (SIP UA) are the endpoints used to make and receive calls [Dang *et al.*, 2002]
- ❖ *Server*: A server is an application program that accepts requests and replies to them by sending back a response to the requesting entity. Examples of servers are proxy, redirect, user agent and registrars.
- ❖ *Proxy server*: An intermediary program that acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them on, possibly after translation, to other servers. A proxy interprets, and if necessary, rewrites a request message before forwarding it.

- *Redirect server*: A redirect server is a server that accepts a SIP request, maps the given address to new addresses and returns these addresses to the client. Unlike a proxy server, it does not initiate its own SIP request. Unlike a user agent server, it also does not accept calls.
- *Registrar*: A registrar is a server that accepts REGISTER requests. A registrar is typically co-located with a proxy or redirect server and may offer location services.
- *Location service*: Used by a SIP redirect or proxy server to obtain information about a callee's possible location(s). Location servers offer location services. Location servers may be co-located with a SIP server.

2.2.3 SIP Addressing

In SIP, users may be contacted using their IP address, if this is known, or by their SIP address. A SIP address is known as a SIP URL (Universal Resource Locator). The syntax is similar to an email address and would look like:

```
sip:someUser@someAddress
```

The *sip:* before the remainder of the address indicates that it is a SIP URL. Optionally the SIP URL may contain a number of other parameters that contain additional information. These parameters include elements like port numbers, type of device and type of media transport to use. A SIP URL allows the calling endpoint to locate a SIP server. This server will be the entity to be contacted first to begin communicating. It may also be the final destination; if it is not, it will redirect the request to the called endpoint.

2.2.4 SIP Network Entities

SIP is a client-server protocol. A SIP endpoint, also known as a user agent consists of a User Agent Client (UAC) and a User Agent Server (UAS). The UAC sends SIP requests while the UAS responds to these requests. Since the user agent contains both the UAC and UAS, SIP can operate as a peer-to-peer operation while using the client server model [Black, 2001]. SIP has four types of servers: proxy server, redirect server, registrar and a UAS.

SIP proxy server

A SIP proxy server receives requests from clients and forwards them either to another server, a redirect server or a UAS. The proxy acts as both a client and a server to make requests on behalf of clients. If the proxy deems it necessary, it may rewrite a request before forwarding it. Figure 2.4 shows a proxy server operating between two user agents, Homer and Marge. Homer sends a request to the proxy server. The proxy server proxies the request to Marge. Marge sends a response to the request to the proxy server. The proxy server then proxies the response to Homer.

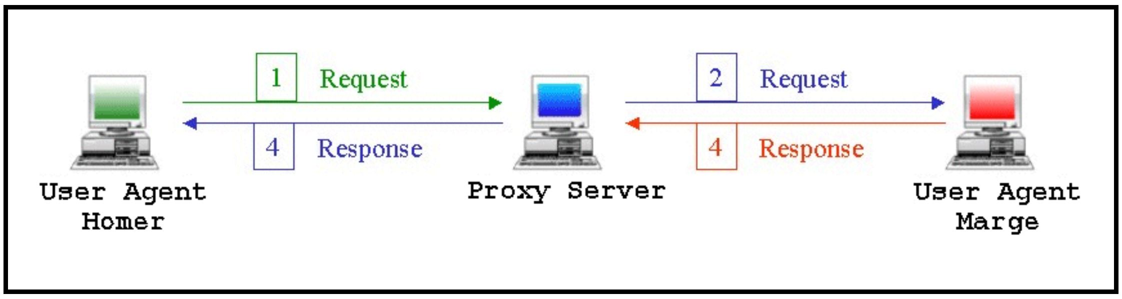


Figure 2.4 Example of SIP proxy server operating between two user agents.

Thus replies to messages sent through a proxy server are also relayed through the proxy server, except that the path is reversed.

Redirect Server

A redirect server is a server that accepts a SIP request, maps the given address to new addresses and returns these addresses to the client. Unlike a proxy server, a redirect server cannot initiate any of its own SIP requests. It also cannot act as a UAS and accept calls. The redirects servers task is simply to provide the information requested. This then allows the requesting client to perform the task without the involvement of the redirect server. Details of the different types of requests that a client can make are explained in 2.2.5. Figure 2.5 shows the operation of a redirect server. A user agent Homer is trying to contact a user agent Marge.

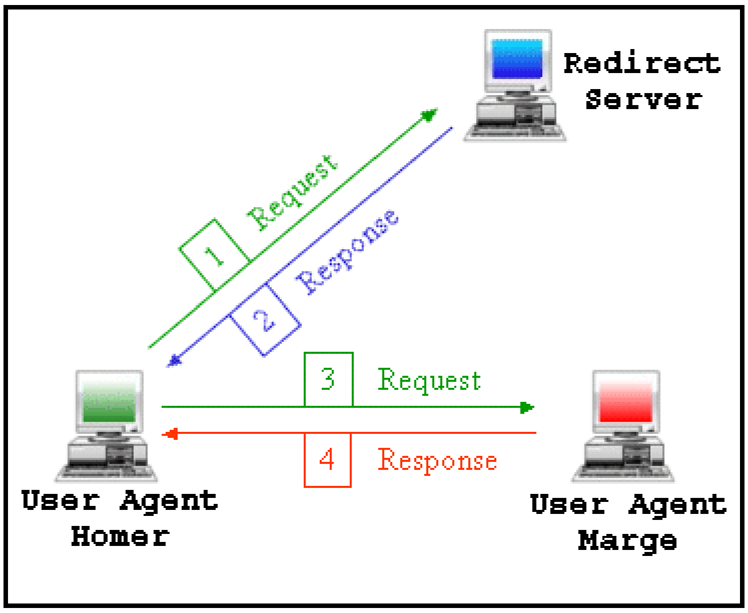


Figure 2.5 Operation of a SIP redirect server.

1. Homer sends a request for the address of Marge to the redirect server.
2. The server maps all known addresses of Marge and returns them to Homer. If Marge cannot be contacted directly, then the address of the next hop server is returned. This would be the next server in the chain of one or more servers between Homer and Marge.
3. Homer sends a request to Marge
4. Marge replies to Homer's request.

SIP Registrar server

This server is usually combined with a proxy or redirect server [Collins, 2001]. A SIP user registers with a registrar server to indicate the address at which the user is available as well as the duration that the user can be reached at this address. The registrar server maintains a mapping between the user's SIP URL and their IP address.

Having looked at the entities that constitute a SIP network, next we look at SIP messaging. It should be noted that like MGCP, SIP too uses the Session Description Protocol (SDP) to describe media.

2.2.5 SIP Messaging

SIP messages are either requests or responses. Clients make requests to servers. Servers respond to clients with responses that are also known as status messages. The structure of a SIP message is shown in Figure 2.6. The start-line contains either a request-line or a status-line. The request-line states the type of request being made. The status-line declares that either the request was a success or in the case of a failure the reason for it and type of failure. The message header provides additional information regarding the request or the response [Collins, 2001]. This includes information like the originator and intended recipient of a call and may include the subject of a call to indicate the reason for the call. The body of a message describes the type of session to be established and the type of media to be used. SIP does not define the structure or content of the message body. Thus it is commonly described using SDP but may be described using other protocols like ISUP.

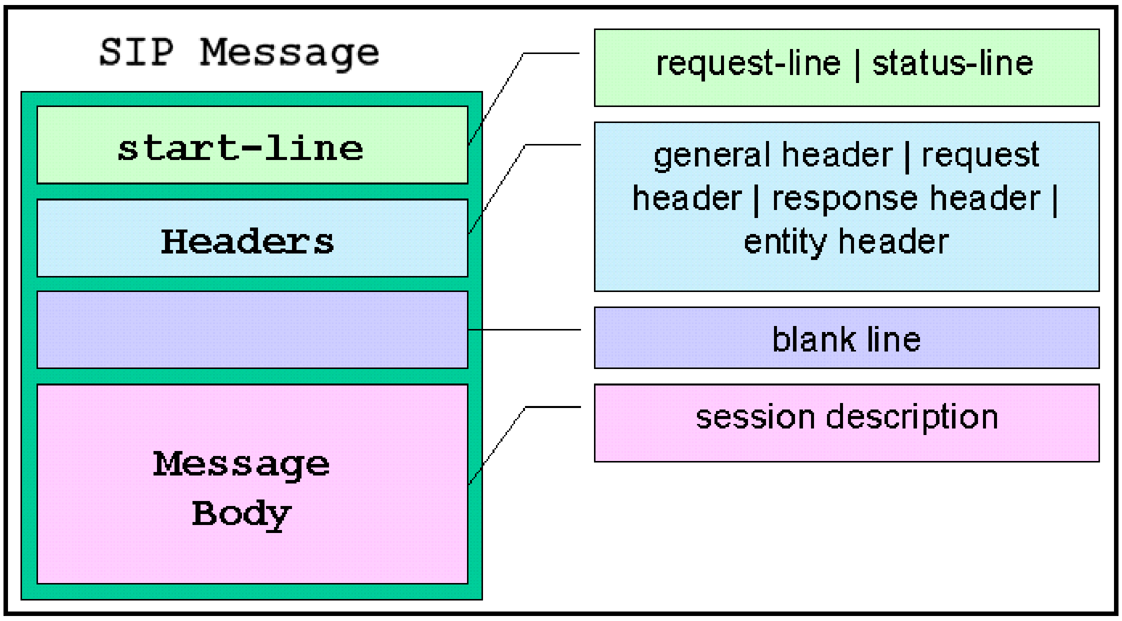

SIP Message
start-line
Headers
Message Body
request-line | status-line
general header | request header | response header | entity header
blank line
session description

Figure 2.6 Structure of SIP message

SIP Requests

According to RFC 2543, SIP has six different types of requests: INVITE, ACK, OPTIONS, BYE, CANCEL AND REGISTER.

- ❖ The *Invite* message is used to initiate a call. It may be used to initiate a call between two endpoints or a multipoint conference.
- ❖ The *Ack* message is sent by a client to a server to confirm that it has received a final response from a server.

- The *Options* message queries a server about its capabilities. It may be used to query a user agent as to whether the user agent supports a particular type of media. It may also be used to anticipate what the response from a user agent would be if it was sent an *Invite* message.
- The *Bye* message is used to end a call. It maybe issued by either party in that call.
- The *Cancel* message is issued to annul a previous request. A cancel message may only be sent to a server if the server has not already replied to the original request.
- A client registers their location with a registrar server by issuing a *Register* message. This lets the registrar server know where the user can be reached. A user may register with several servers and may also have several registrations with one registrar [Collins, 2001]. This occurs when a user registers with a registrar using several SIP endpoints.

SIP Responses

A SIP response contains a status code and a reason phrase. The status code is a three-digit number that indicates the outcome of the request. The reason phrase is an explanation of the response code. The status codes defined in RFC 2543 have values between 100 and 699 with each interval of 100 numbers representing a response class. The six classes of response codes and their meaning are displayed in table 2.2.

Code	Meaning
1XX	Informational: request received, continuing to process the request
2XX	Success: The request has been received, understood and accepted.
3XX	Redirection: Further action must be taken in order to complete the request.
4XX	Client error: The request contains bad syntax or cannot be fulfilled at this server. The request is rejected.
5XX	Server error: The server failed to fulfil an apparently valid message owing to an internal error with the server.
6XX	Global failure: The request is invalid at any server.

Table 2.2 SIP response codes

2.2.6 SIP Message Sequence

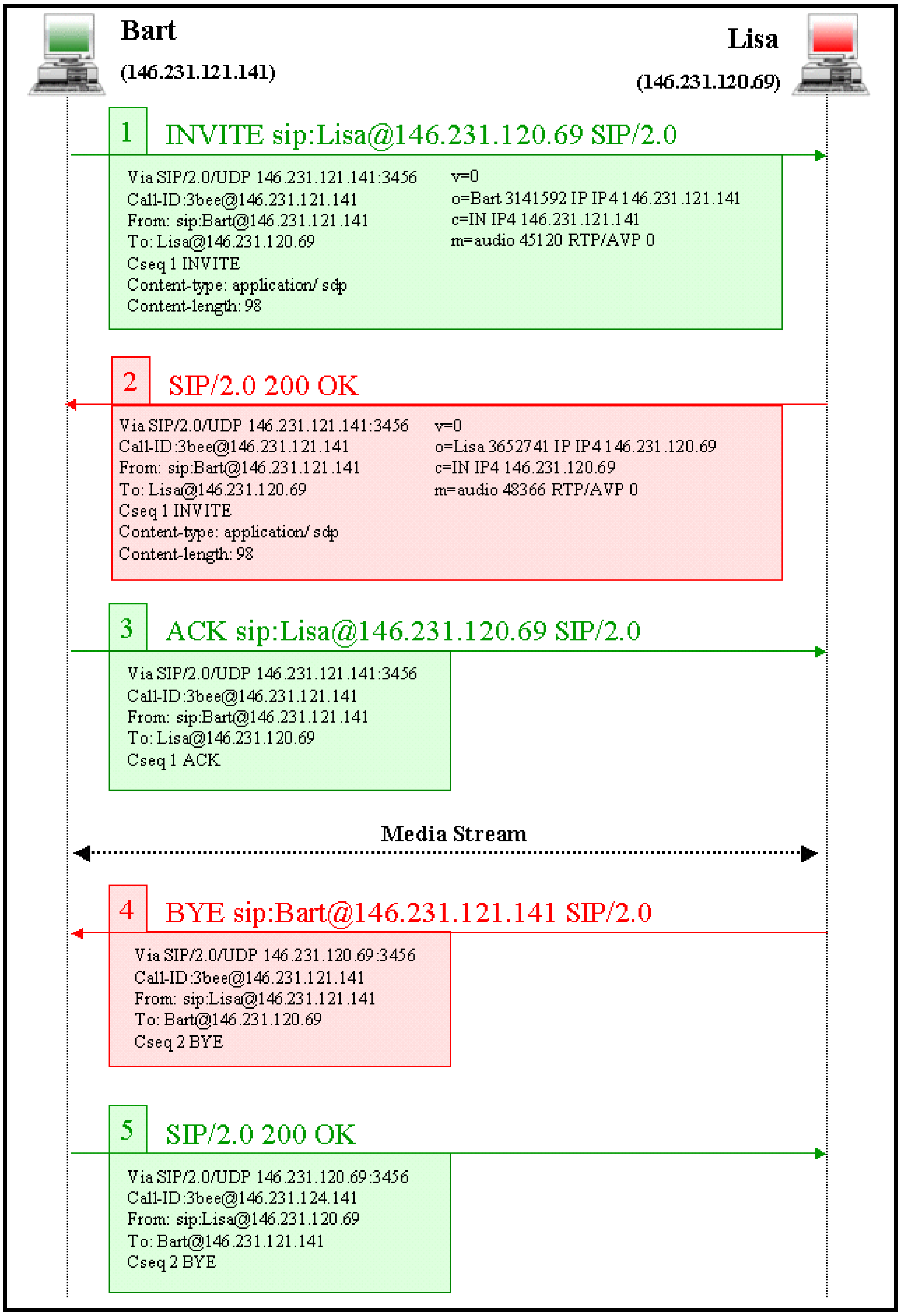
To conclude this introduction to SIP, we look at a simple call scenario. The complete call signalling messages are shown with all the mandatory headers. In this scenario a SIP user Bart calls another SIP user Lisa. The message sequence is as follows:

1. Bart initiates the call by sending an *Invite* message.
2. Lisa accepts the call and returns a *200 OK* message.
3. Bart sends Lisa an *Ack* message.
4. When they have completed their conversation Lisa sends Bart a *Bye* message to terminate the call.
5. Bart responds to this with a *200 OK* message and the call is ended.

The actual messages used are shown in Figure 2.7. For simplicity, it is assumed that Bart knows Lisa’s IP address. Bart’s messages are in green and Lisa’s in red.

The following should be noted:

- ❖ The Call-Id of *3bee@146.231.121.141* remains the same for the entire call.
- ❖ The Cseq number which every request has to have only increments when a new request is made.
- ❖ Bart can receive µlaw PCM data (RTP/AVP 0) at 146.231.121.141 on port 45120
- ❖ Lisa can receive µlaw PCM data (RTP/AVP 0) at 146.231.120.69 on port 48366
- ❖ RTP uses even numbered ports and RTCP uses odd numbered ports.



1 INVITE sip:Lisa@146.231.120.69 SIP/2.0

```
Via SIP/2.0/UDP 146.231.121.141:3456      v=0
Call-ID:3bee@146.231.121.141              o=Bart 3141592 IP IP4 146.231.121.141
From: sip:Bart@146.231.121.141            c=IN IP4 146.231.121.141
To: Lisa@146.231.120.69                   m=audio 45120 RTP/AVP 0
Cseq 1 INVITE
Content-type: application/ sdp
Content-length: 98
```

2 SIP/2.0 200 OK

```
Via SIP/2.0/UDP 146.231.121.141:3456      v=0
Call-ID:3bee@146.231.121.141              o=Lisa 3652741 IP IP4 146.231.120.69
From: sip:Bart@146.231.121.141            c=IN IP4 146.231.120.69
To: Lisa@146.231.120.69                   m=audio 48366 RTP/AVP 0
Cseq 1 INVITE
Content-type: application/ sdp
Content-length: 98
```

3 ACK sip:Lisa@146.231.120.69 SIP/2.0

```
Via SIP/2.0/UDP 146.231.121.141:3456
Call-ID:3bee@146.231.121.141
From: sip:Bart@146.231.121.141
To: Lisa@146.231.120.69
Cseq 1 ACK
```

4 BYE sip:Bart@146.231.121.141 SIP/2.0

```
Via SIP/2.0/UDP 146.231.120.69:3456
Call-ID:3bee@146.231.121.141
From: sip:Lisa@146.231.121.141
To: Bart@146.231.120.69
Cseq 2 BYE
```

5 SIP/2.0 200 OK

```
Via SIP/2.0/UDP 146.231.120.69:3456
Call-ID:3bee@146.231.124.141
From: sip:Lisa@146.231.120.69
To: Bart@146.231.121.141
Cseq 2 BYE
```

Figure 2.7 Complete call signalling messages

2.3 H.323

As with the introduction to SIP, this brief introduction to H.323 too does not attempt to be complete, but rather, it affords the reader the necessary H.323 fundamentals for the later chapters.

Protocol Overview

H.323 is an International Telecommunications Union (ITU-T) recommendation. The recommendation specifies a methodology and architecture that incorporates many other recommendations [Collins, 2001]. Two of the more important recommendations included are H.245 [ITU-T, 1998b] and H.225 [ITU-T, 1998c] which will be considered later.

2.3.1 H.323 Architecture

Based on the success of RTP as a media transport protocol, H.323 was developed to provide call signalling for these multimedia sessions. H.323 has undergone several revisions and is currently in its fourth version.

From its inception, the H.323 suite of protocols specified an architecture to allow its endpoints to communicate. The most basic element of an H.323 environment is a terminal. It is the fundamental component responsible for generating and terminating audio, video, and data streams [Penton *et al*, 2001]. A simple H.323 environment may consist of a number of terminals without any of the other components. Such an environment would only be capable of single point-to-point multimedia calls within an IP network.

An H.323 network may contain a gateway, which facilitates the communication with other types of networks, like the PSTN. The gateway operates in the same way that MGCP gateways operate, as described in 2.1. An H.323 network may also contain a gatekeeper. This network element controls a number of H.323 terminals, gateways and Multipoint Controllers (MCs) [Collins, 2001]. A gatekeeper controls the network access, bandwidth allocation and may support aliases by providing address translation. An MC is an H.323 endpoint that manages multipoint conferences between three or more terminals and/or gateways [Collins, 2001]. The collection of gateways, MCUs and terminals that a gateway controls is known as a zone. An example of an H.323 zone is shown below.

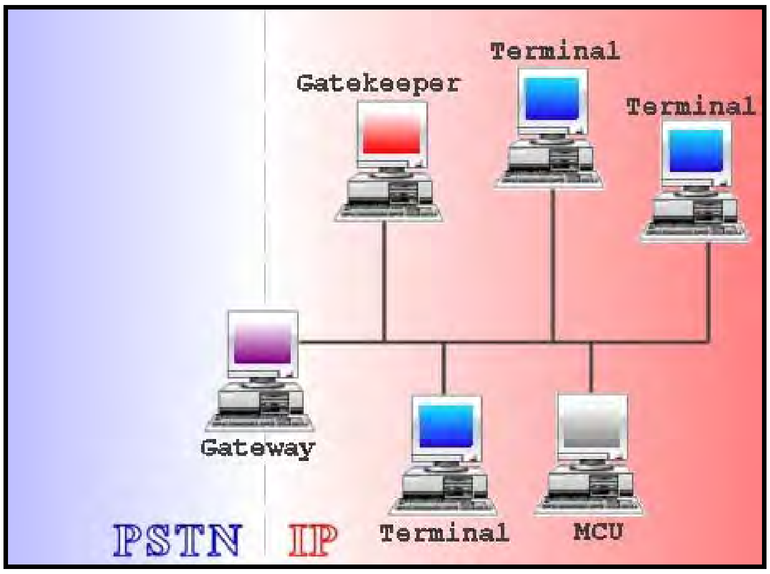


Figure 2.8 H.323 architecture

2.3.2 H.323 Protocols

As mentioned earlier, H.323 is composed of a number of protocols. Table 2.3 shows the H.323 protocol stack. Starting from the bottom of the stack, there is the physical layer, data link layer and then the network layer with IP. Above this H.323 uses both UDP and TCP. It uses TCP or UDP for call and control signalling and Data communications. It uses UDP to transport the media. As with SIP, the media is transported using RTP. The Real Time Control Protocol (RTCP) is used to provide feedback information to the communicating parties [Collins, 2001]. To setup and tear down calls, H.323 uses two protocols H.225 and H.245.

Audio	**VIDEO**	Data	Terminal / Application control

Audio codecs	Video codecs	T.120	H.225 Call signalling	H.225 RAS signalling	H.245 Control signalling
RTP/RTCP					
UDP		UDP or TCP			
Network Layer (IP)					
Data link layer					
Physical Layer					

Table 2.3 H.323 protocol stack

H.225 has two parts to it, call signalling and RAS (Registration, Admission and Status) signalling. The call signalling part of H.225 is a variant of Q.931 and is used to setup and tear down calls between H.323 endpoints. This research may use the terms 'H.225 call signalling' and 'Q.931 signalling' interchangeably. The RAS signalling is used between terminals and a gatekeeper to allow the gatekeeper to manage its zone. Terminals register with the gatekeeper using RAS, which is also used by the gatekeeper to allow or deny them access to the network.

H.245 is used to manage the media stream in a call between two or more endpoints. First it establishes logical channels between the calling entities. It then establishes the type of media to be exchanged and ensures that the parties in the call use the selected format. More details on the use of H.245 will be explained in section 2.3.5.

Note to the reader: H.323 signalling is specified using Abstract Syntax Notation 1 (ASN.1), which is easily interpreted by software tools, but is not human-readable. For this reason, text descriptions are given instead of the ASN.1 syntax.

2.3.3 H.323 Addressing

Entities on an H.323 network have a unique address based on their IP address. This address may also be represented as a URL if a Domain Name Service (DNS) is present. H.323 entities may also be identified using Transport Service Access Point (TSAP) identifiers. TSAP identifiers use the entity's IP address and its signalling port number [Collins, 2001]. Lastly, H.323 entities may use one or more aliases. Since

H.323 messages must contain the IP address of the called entity, the alias address must be translated to reveal the real address. A gatekeeper performs this function.

2.3.4 RAS Signalling

RAS signalling is exchanged between a gatekeeper and the endpoints that it controls within its zone [Collins, 2001]. However, since a gatekeeper is an optional entity, RAS signalling is also optional. It is included in this introduction as it will be required in chapter 5. RAS signalling is quite extensive, therefore only those functions used in this research will be covered, namely admission, disengage, registration and endpoint location.

Registration

Endpoints register with a gatekeeper by issuing a *RegistrationRequest* (RRQ) message. The message is sent to a gatekeeper that either the endpoint has been pre-configured to register with or one that has been discovered by a procedure known as gatekeeper discovery. The *RRQ* message contains the endpoint's address for RAS messages as well as its address for call signalling [Collins, 2001]. The *RRQ* message may also contain aliases and alternative addresses at which the user can be reached.

A gatekeeper can respond to a *RRQ* message, with either a *RegistrationConfirm* (RCF) message or a *RegistrationReject* (RRJ) message. If the gatekeeper accepts the registration, it will assign the endpoint a unique identifier, which must be used for all subsequent RAS messages. It may also assign the endpoint an alias if the endpoint had not specified one in the *RRQ* message.

Endpoint Location

Endpoint location is a service that translates aliases to real addresses. The service is provided by gatekeepers. If an endpoint wants to know the address of another endpoint, it sends a *LocationRequest* (LRQ) message to its gatekeeper. If the gatekeeper knows the endpoint's address, it replies with a *LocationConfirm* (LCF) message. If the gatekeeper does not know the address, it responds with a *LocationReject* (LRJ) message. The gatekeeper may then send a *LRQ* message to another gatekeeper or it may multicast the message to the gatekeeper discovery multicast address 224.0.1.141.

Admission

Admission is a request from an endpoint to a gatekeeper for permission to participate in a call. The endpoint does this by sending an *AdmissionRequest (ARQ)* message to the gatekeeper. The *ARQ* message contains many parameters including the type of call, a call identifier, a call-reference value and information about the other parties in the call like their signalling address. The most important mandatory parameter in the *ARQ* message is the bandwidth parameter [Collins, 2001]. This specifies how much bandwidth the media in the call will require.

The Gateway responds to the *ARQ* message with an *AdmissionConfirm (ACF)* message. This message contains many of the parameters of the *ARQ* message, but stipulates the bandwidth to be used in the call as well as the call model. The call model specifies how the call signalling is to be sent. If it is to be sent directly from one endpoint to the other endpoint, it is known as direct call signalling. If the call signalling is to be sent via the gatekeeper, it is known as gatekeeper-routed call signalling.

Disengage

When terminals wish to end a call, they close their media streams and terminate their session. They then send a *DisengageRequest (DRQ)* message to their gatekeeper(s). The *DRQ* message contains the call reference, call identifier and the reason for disengaging. The Gatekeeper responds to this with a *DisengageConfirm (DCF)* message.

2.3.5 H.323 Call setup

Call setup can be viewed as a four-stage process. By looking at these stages the call signalling necessary to establish a call will be explored. To establish a call between two H.323 endpoints, first a TCP connection is made between the endpoints. This connection carries the setup messages using Q.931 signalling described earlier. Next a second TCP connection is made that carries the call control H.245 messages. Once this second connection has been established the Q.931 channel is not required and may be closed by either endpoint [Hersent *et al*, 2000]. The H.245 channel is used to establish the media capabilities of the terminals and a master / slave relationship between the endpoints. The channel then opens logical channels for the media to be transported using RTP. The H.245 channel stays open for the duration of the call and is used to signal the end of the call.

Stage 1: Initiating the call

The following Q.931 messages are used to setup a call. There are a number of optional messages that have not been included here.

***Setup*:** an initial message used to begin a call.

***Alerting*:** sent by the called endpoint to indicate that the user is being alerted.

***Connect*:** sent by the called endpoint to indicate that the user has accepted the call.

***Release Complete*:** used to end a call.

***Status Facility*:** a request for the remote end of the call to report on the status of the current call.

A call is initiated with a *Setup* message and accepted with a *Connect* message. These messages are sent using a TCP connection.

Stage 2: Establishing the control channel

This occurs via the second TCP connection. This connection is setup when one of the endpoints specifies an H.245 transport address. This channel is also known as logical channel 0. Once the channel 0 has been established, the *TerminalCapabilitiesSet* message is exchanged to determine the media capabilities of the endpoints. This message is acknowledged using a *TerminalCapabilitiesSetAck* message. Next the terminals determine who is master and who is slave in the call. This notion of master and slave was devised to determine who should do what when two endpoints are both capable of performing a task that has to be performed by only one of them. Master is determined by exchanging *MasterSlaveDetermination* messages. The messages contain a terminal type value and a random number. The terminal type value reflects the capabilities of an endpoint. The priority in ascending order is terminal > gateways >gatekeepers > MCU.

Stage 3: Establishing media channels

This is done using H.245 *OpenLogicalChannel* messages. These messages contain parameters specifying the type of media, codec to be used, UDP address and port, logical channel number and RTP information. Logical channels are one way, with the exception of ones that carry data; therefore both endpoints in a call need to establish at least one logical channel.

Stage 4: Dialogue and ending the call

Once the logical channels have been setup, the users can communicate. The media is sent using RTP streams that are monitored with RTCP messages. When the users wish to end the call, they need to close all the logical channels that they opened using *CloseLogicalChannel* messages. These messages are acknowledged with *CloseLogicalChannelAck* messages. After the logical channels have been closed, they have to close the H.245 channel, which is done by sending an H.245 *endSessionCommand* message. Lastly a Q.931 *Release complete* message is sent if this channel is still open. This closes the channel and ends the call. Although the procedure stated above is the prescribed way to end a call, many terminals simply terminate the call.

2.3.6 H.323 Call Scenario

We now look at two call scenarios to demonstrate the call signalling and control messages described above. The first call is a point-to-point call without a gatekeeper shown in Figure 2.9. The call in this scenario is from Terminal 1 to Terminal 2. Terminal 1 initiates the call with a *Setup* message. Terminal 2 accepts the call with a *Connect* Message. The terminals then exchange media capabilities and communicate in the medium of their choice. When their communication is over they close the H.245 session and then the H.225 session to end the call.

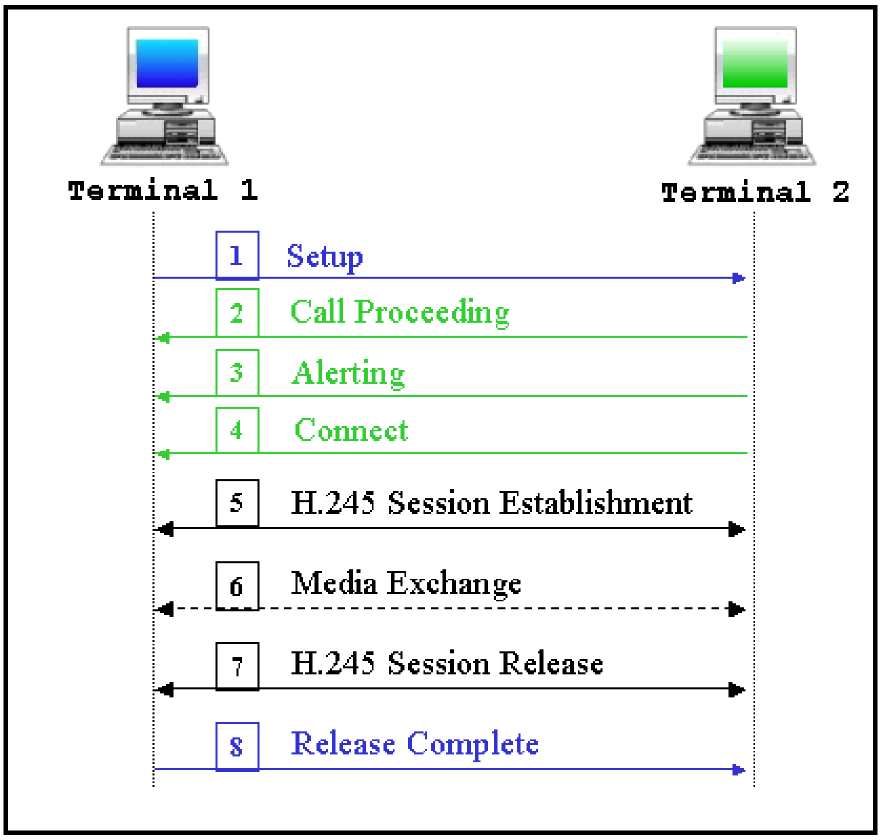


Figure 2.9 Basic point-to-point call setup

Next, we look at a more complex scenario in which two terminals (Terminal Alpha and Terminal Beta) that are registered with different gatekeepers communicate. Gatekeepers may chose whether or not they want to route the call signalling or let the endpoint handle it. In this scenario adapted from [Collins, 2001], Alpha's gatekeeper opts to route the call signalling, while Beta's gatekeeper opts not to.

Alpha begins the call by sending the gatekeeper an *ARQ* message. The gatekeeper responds with an *ACF* message and indicated that *callmodel* will be gatekeeper-routed call signalling. Alpha then sends a *Setup* message to its gatekeeper. The gatekeeper immediately responds with a *Call Proceeding* message. The gatekeeper then sends a *Setup* message to Beta. Beta responds with a *Call Proceeding* message and then sends an *ARQ* message to its own gatekeeper. The gatekeeper responds with an *ACF* message indicating that call signalling should be routed directly.

Beta then alerts Alpha by sending an *Alerting* message to Alpha's gatekeeper. Alpha's gatekeeper forwards this message to Alpha. When Beta accepts the call with a *Connect* message it is routed in the same way as the *Alerting* message. The H.245

messages are then exchanged directly between the terminals. If either gateway wished to be in the path of the messages, it could be. Closing the H.245 channel and the H.225 channel ends the call. The endpoints then send their respective gatekeepers *DRQ* messages. The gatekeepers end the session with *DCF* messages to their respective terminals.

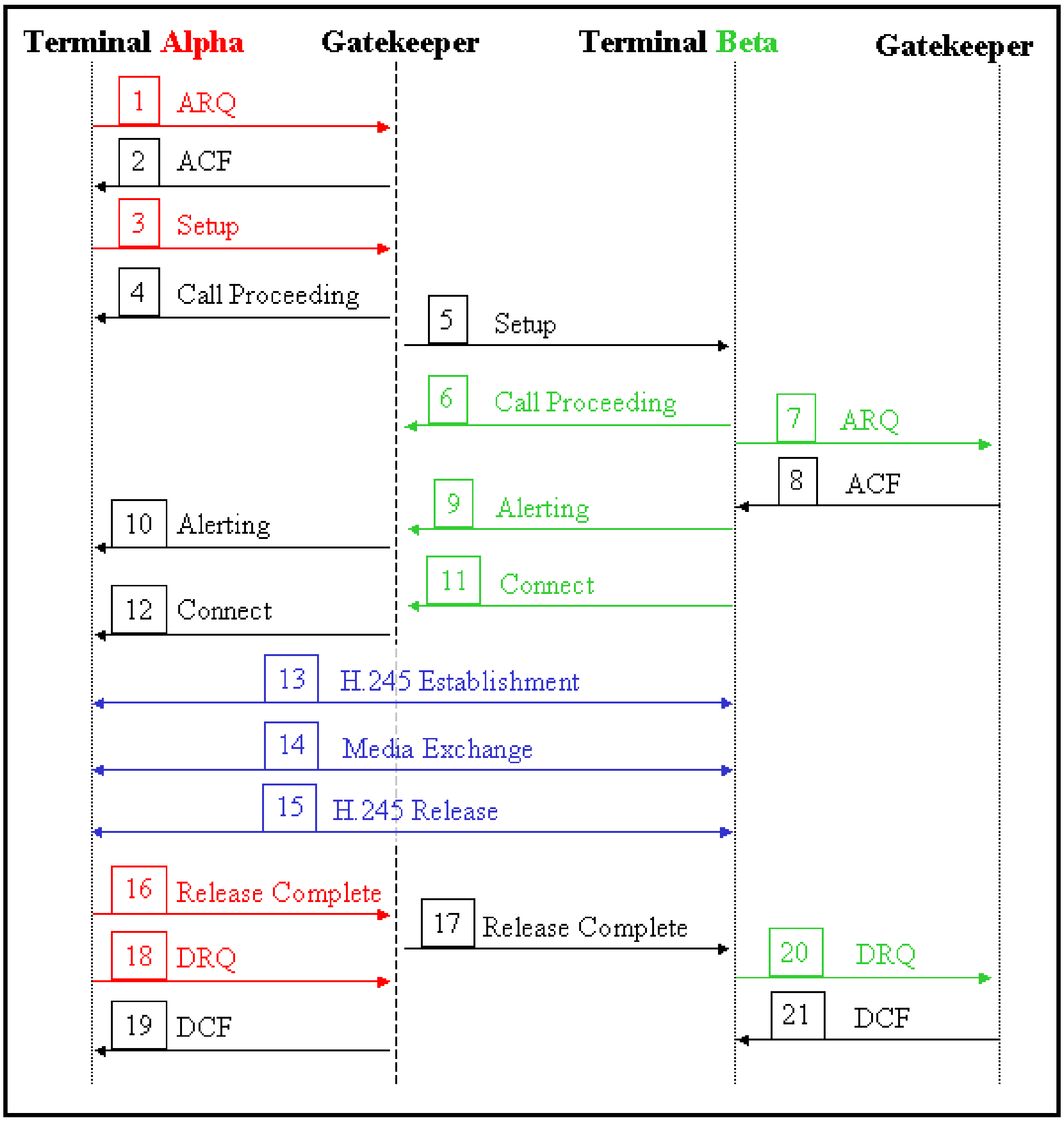


Figure 2.10 A basic call with gatekeeper routing

Fast-connect procedure

As can be seen from the above call scenario, setting up a call can require the exchange of quite a number of messages. To reduce the call signalling overhead, H.323 has a procedure known as the Fast-connect procedure. This procedure sets up media channels as quickly as possible, eliminating the need to set up a separate H.245 control channel. To achieve this, it includes a faststart element within the user-to-user element [Collins, 2001]. The faststart element is one or more *OpenLogicalChannels* message. It also includes the reverse logical channels parameters if the caller intends to receive media from the called endpoint. Essentially, the caller offers the callee a choice of media. The callee may send the faststart element in any of the reply messages up to and including the *Connect* message. If the caller does not receive the faststart element, the caller will assume that H.245 signalling should be used. Figure 2.11 shows the use of faststart.

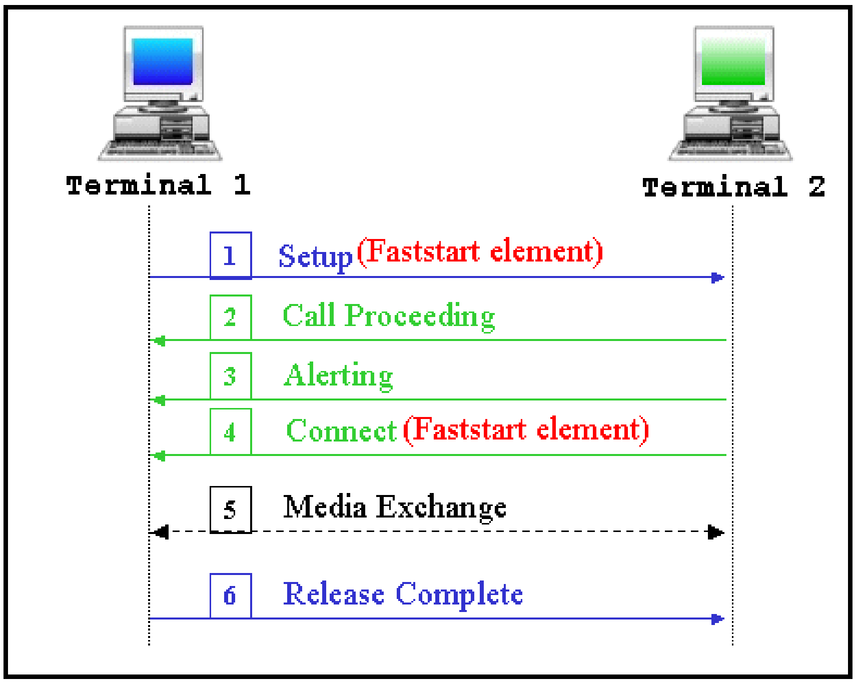


Figure 2.11 Fast-connect procedure

2.3.7 Chapter Summary

Chapter 2 introduced MGCP, SIP and H.323. This introduction went into detail on the call control aspects of these protocols to give the reader the required level of understanding for the following chapters.

Section 2.1 described the MGCP model, the commands and a call scenario. This was to familiarise the user with the call signalling and control capabilities of MGCP. Section 2.2 described the SIP architecture, terminology, addressing, network entities, commands and a call scenario. As with 2.1, this section introduced SIP from a call signalling point of view. Section 2.3 described H.323 in the same way that 2.2 described SIP.

In chapter 3 we look at services that were developed to explore the extent of interaction between the three protocols in call control situations.

Chapter 3

3.1 The H.323 SMS Service

This service allows an H.323 terminal to send a Short Message Service (SMS). The service was developed to investigate the call control used by H.323 and MGCP and how the two protocols communicate. The details of how the service works will be discusses later, but first a look at the technologies employed and the environment used to develop this service.

3.1.1 Technologies employed

SMS
Short message service (SMS) is a globally accepted wireless service that enables the transmission of alphanumeric messages between mobile subscribers and external systems such as electronic mail, paging, and voice-mail systems. An SMS can be sent to a mobile phone from another mobile phone or from an IP network with the aid of a GSM modem. These messages can be replied to and the replies sent to the originator of the SMS message.

GSM SMS Gateway

A GSM SMS Gateway/Modem is used to exchange messages between GSM and IP networks. It allows messages from an IP network to be sent to mobile telephones on a GSM network and vice versa. Physically a GSM modem is a piece of hardware that interfaces with a PC's serial port. To simplify its access, especially if the modem is attached to a remote machine on the network, additional protocols are often added. The modem used for this research was abstracted by using a SOAP layer.

SOAP

SOAP stands for Simple Object Access Protocol. It is a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized, distributed environment using XML, [Box *et al*, 2000]. It provides a schema that specifies how entities are to interact without actually defining specific application mechanisms. This makes it open to a large variety of systems. In the context of this research, these features make SOAP an ideal choice, since the GSM modem is being used for a number of concurrently running research projects at Rhodes University [Halse and Wells, 2002]. To view the XML schema for the GSM gateway used in this research as well as the SOAP messages, please see appendix 1.

3.1.2 The environment

The platform selected is Red Hat Linux 7.1. The MGCP stack used is included in the VOCAL package from Vovida[5]. This package was selected because previous and ongoing research at Rhodes University on related protocols is being conducted using this environment. The VOCAL stack's MGCP 1.2 package is MGCP version 1.0 compliant. Once the vocal package was installed, RFC 2705 (Media Gateway Control Protocol Version 1.0) [Arango *et al*, 1999] was used extensively as a reference.

On the Rhodes University Computer Science Department (RUCSD) network an H.323 version three compliant environment is deployed. It consists of a gatekeeper, a Multi Point Control Unit (MCU), two gateways, and a number of different terminals. Figure 3.1 shows the environment deployed on the RUCSD network.

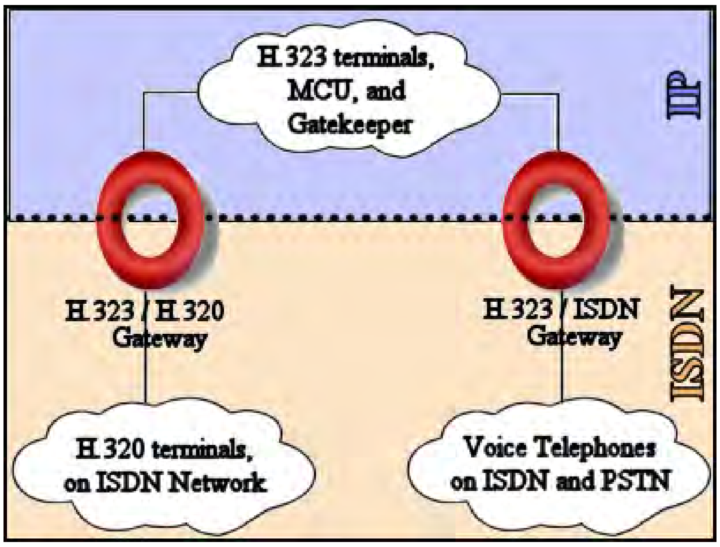

H.323 terminals, MCU, and Gatekeeper
IP
H.323 / H.320 Gateway
H.323 / ISDN Gateway
ISDN
H.320 terminals, on ISDN Network
Voice Telephones on ISDN and PSTN

Figure 3.1 H.323 environment deployed on the RUCSD network

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

3.1.3 Deploying the service

The aim of this service is to allow an H.323 terminal to send an SMS. H.323 provides a number of mechanisms to permit service creation. The challenge that this service presents is in terms of approach. There are two possible approaches to deploying this service.

The first approach is to use an H.323 terminal as a server. The server would function as a terminal with 'enhanced' capabilities in much the same way that a gateway does. This means that it would follow the H.323 standard for communications with H.323 terminals as well as being able to interface with a GSM gateway. Although this approach provides a possible solution, it is a specific non-extensible solution. What if another VoIP protocol wants to use the service? The service would have to be redeveloped from scratch to cater for the other protocol.

This leads us to the second approach, a more generic solution with the aid of MGCP. It is envisioned that deploying such a service with the aid of MGCP leads to a non-protocol specific solution. The benefit of this solution is a reduction in the time required to extend the service to other protocols, like SIP. This approach also facilitates an investigation into call control using MGCP, H.323 as well as the inter-working between the two.

3.1.4 Overview of the service

Figure 3.2 depicts the physical representation of the various components used in this service. It also shows a high level view of the communication that takes place when sending an SMS from an H.323 endpoint to a cellular telephone. This figure represents the GSM SMS gateway abstracted by the SOAP layer and an MGCP gateway as an MGCP gateway.

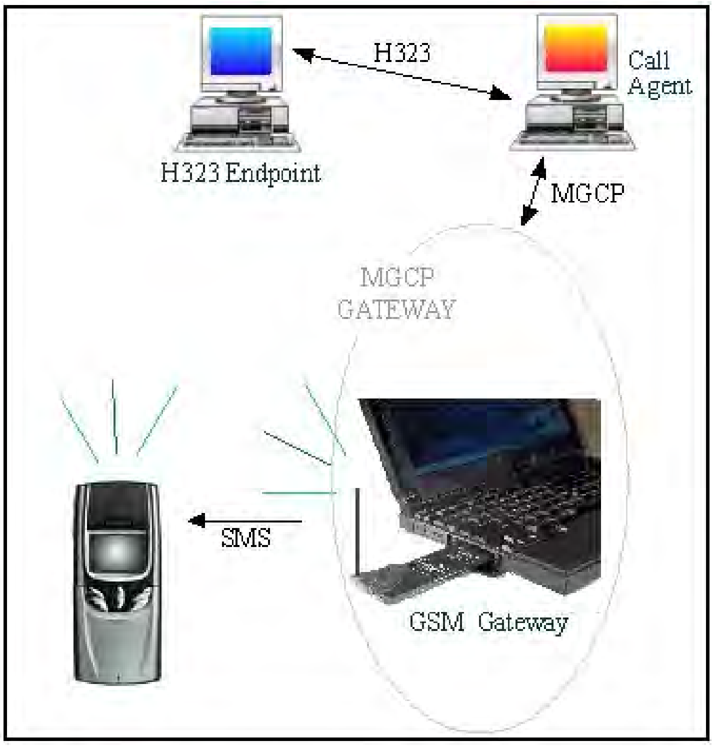


Figure 3.2 Overview of H.323 SMS Service

The SMS service works as follows: The H.323 terminal initiates a call with the MGCP Call Agent (CA), which acts as an H.323 terminal. The CA receives the request and signals the gateway that the H.323 terminal will be sending a text message. The CA then signals the H.323 terminal to send the message. The gateway receives the message and makes an HTTP post to the GSM gateway. The GSM gateway receives the mobile telephone's number, the message and the sender's ID and sends the SMS message. With this basic understanding of the control and message sequence, we now present the actual call setup and flow in detail, as well as the development issues that were encountered.

To begin with, the CA needed to be modified to allow it to communicate with the H.323 terminal. This was achieved by modifying a basic CA to incorporate the H.323 stack. The H.323 stack used was from [openH323, 2003]. This modification allowed the CA to communicate with H.323 terminals as well as with the gateway. The CA could be modified in a similar fashion to communicate with SIP endpoints.

Next the Media gateway (MG) was further decomposed into an MGCP compliant gateway and a GSM SMS Gateway. This was done so that the MG would not necessarily have to reside on the same machine as the GSM modem. This added to the distributed nature and flexibility of the MGCP model. The MG was then modified to allow it to make an HTTP post to the GSM gateway. As stated earlier, the GSM gateway was easily accessible, owing to the addition of a SOAP layer.

Having set up the various components for this service, let us take a closer look at the communication between the entities. Figure 3.3 shows the communication sequence. Note that that the media stream for this service was a TCP stream, but it has been depicted in figure 3.3 as an RTP stream. This is because the service was developed as a proof of concept that could be extended to voice or video later.

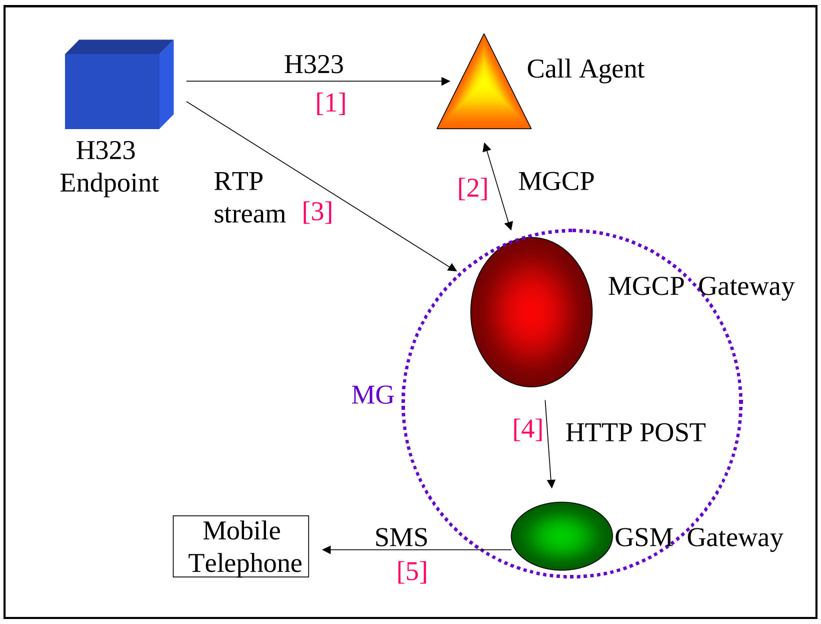


Figure 3.3 Diagram of communication sequence

1. H.323 messages between the H.323 endpoint to the CA to initiate and terminate the call.
2. MGCP messages between the CA and MGCP gateway to accept RTP stream from the H.323 endpoint.
3. RTP Stream (containing the mobile telephone number and the message) from the H.323 endpoint to the MGCP gateway.
4. HTTP post from the MGCP gateway to the GSM SMS Gateway.
5. SMS message to the designated mobile telephone.

Taking a closer look at the actual messages will give us an idea of how H.323 inter-works with MGCP. Figure 3.4 shows H.323 messages in blue and MGCP messages in red. This service makes use of H.323's Fast-connect procedure, which is used to set up media streams as quickly as possible. A terminal using this procedure assumes that the terminal it is calling has certain media capabilities. This allows it to greatly reduce the number of H.245 messages necessary to set up a media stream.

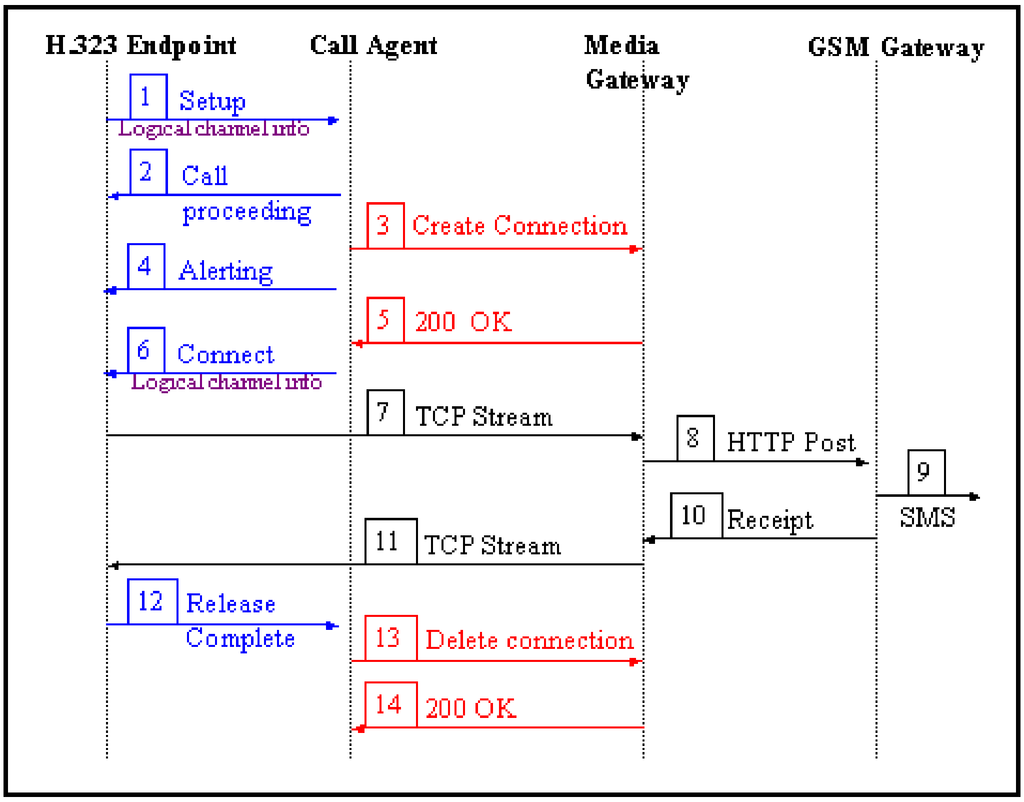


Figure 3.4 Message exchange for H.323 SMS Service

1. The H.323 endpoint sends a *Setup* message to the C.A. The *Setup* message contains a faststart element. The faststart element in this case is an *OpenLogicalChannel* request, which also contains the reverse logical channel parameters. In other words where to send the delivery report for the SMS.
2. The CA responds with a *Call proceeding* message. This informs the caller that the *Setup* message has been received and that call establishment procedures are underway.

3. The CA gets the caller’s IP address and sends a *Create connection* message to the MG with the caller’s IP address as part of the SDP.
4. The CA sends the H.323 endpoint an *Alerting* message. This informs the H.323 endpoint that the call setup is proceeding and is used as a measure to prevent a timeout.
5. The MG starts the TCP server and responds with a *200 OK* message. The IP address of the MG as well as the port of the TCP server are sent to the CA as part of the SDP
6. The CA responds to the H.323 endpoint’s *Setup* message with a *Connect* message. It sends the IP address of the MG as well as the port of the TCP server in response to the *OpenLogicalChannel* message.
7. The H.323 endpoint sets up a TCP stream with the MG and sends the mobile telephone number and the message.
8. The MG makes a HTTP post to the GSM gateway.
9. The GSM gateway sends the SMS.
10. The GSM gateway returns a status report of the SMS. It returns either an *OK* or *error* message. An explanation of this is given in appendix 1.
11. The MG returns a status message to the H.323 endpoint. The H.323 endpoint may send another SMS or end the call.
12. The H.323 endpoint ends the call by sending the CA a *Release Complete* message. When used with the Fast-connect procedure, this message has the same effect as closing the H.245 channels and terminating the call. It does not have to wait for a response since it is using reliable transport, in this case TCP.
13. The CA sends a *delete connection* message to the MG.
14. The MG stops the TCP server and responds with a *200 OK* message.

At this point the service is complete. The inter-working of H.323 and MGCP with regard to call control, and the outcome of this service will be discussed in Chapter 4. Next, the SMS service is extended to SIP endpoints.

3.2 The SIP SMS Service

SIP uses sessions for multimedia communications, which differ from H.323 calls, but will this approach add complexity or simplify the development of this service? The aim of this service is to offer SIP endpoints the ability to send SMS messages in the same way that the H.323 SMS service did for H.323 endpoints.

Having made the claim that deploying a single media gateway would be more practical than deploying protocol specific gateways, the implementation of this service is intended to test the claim.

3.2.1 The environment

Two SIP environments have been deployed on the RUCSD network. One is VOCAL, which uses the SIP stack from Vovida and the other uses the CINEMA architecture from Columbia University. The SIP environment chosen for the development of this service is CINEMA. The reasons for choosing this environment are that it is easier to use, better documented and requires less to allow the simplest call between two endpoints than the SIP stack offered by VOCAL. This decision was reached based on the recommendations made by M.C. Hsieh of the Computer Science Department of Rhodes University [Hsieh *et al*, 2002]. The architecture for the SIP service is very similar to the H.323 service shown in Figure 3.2. The only difference is that a SIP UA replaced the H.323 endpoint in the figure.

3.2.2 Deploying the service

To begin with, the CA needed to be modified to allow it to communicate with a SIP UA. This is achieved by modifying a basic CA to incorporate the SIP stack, the same way the CA had been modified for the H.323 service. The modification allows the CA to communicate with SIP terminals as well as with the media gateway. The media gateway used here is the same one used in the H.323 SMS service without any modifications.

Figure 3.5 shows the structure of the service. Comparing this to Figure 3.3, the architecture is identical. The only difference between the two figures is the protocol used to communicate with the CA.

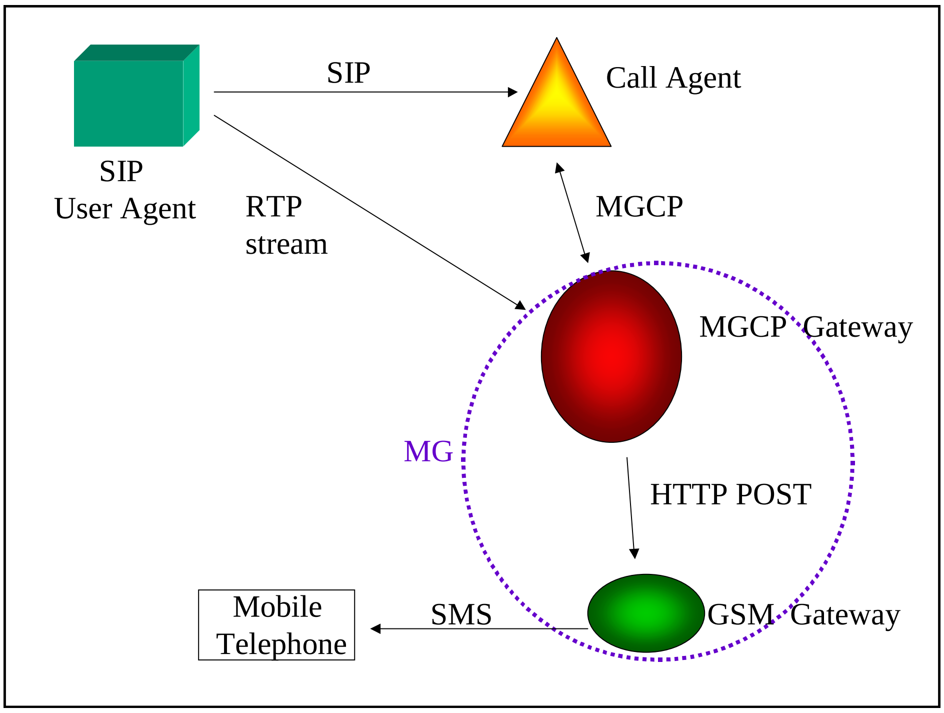


Figure 3.5 SIP SMS Service

The service works as follows:

1. The SIP UA initiates a call by sending an *Invite* message to the CA.
2. The CA then sends a *Create connection* message to the MG to inform it that it will be receiving a message from the endpoint.
3. The MG accepts the connection from the CA. The MG then starts a TCP server and sends an *OK* message back to the CA. As part of the *OK* message, the MG includes the session description containing the IP address and port of the TCP server.
4. The CA then accepts the call using a *200 OK* message which signals the endpoint to send its message to the MG, giving the endpoint the IP address and port to send the message to.
5. The endpoint replies with an *Ack* message to the CA. A session has now been established between the SIP UA and the CA.

6. The endpoint then communicates with the TCP server to send the MG a message containing the mobile phone’s number and a message. The TCP stream between the SIP UA and the MG mimics the RTP stream of a voice call.
7. Once the MG receives this message, it makes an HTTP post to the GSM gateway.
8. The GSM gateway receives the mobile telephone’s number, the message and sends the SMS message.
9. After sending the message, the GSM gateway sends the MG a receipt with the status of the message.
10. The MG then passes this back to the endpoint. It returns either an *OK* or *error* message. An explanation of this is given in appendix 1.The endpoint may send as many SMS messages as it wants to.
11. Once the SIP UA has finished sending SMS messages and wishes to terminate the call, it sends the CA a *Bye* message to end the session.
12. The CA sends the MG a *delete connection* message.
13. The MG stops the TCP server and ends its call with the CA. It responds with a *200 OK* message.
14. Lastly the CA replies to the SIP UA with a *200 OK* message to end their session

The message flow is shown in Figure 3.6. Messages 1 to 8 are used to setup the call and send the SMS. Messages 9 to 14 are used to tear down the call. SIP messages are in green and MGCP messages are depicted in red.

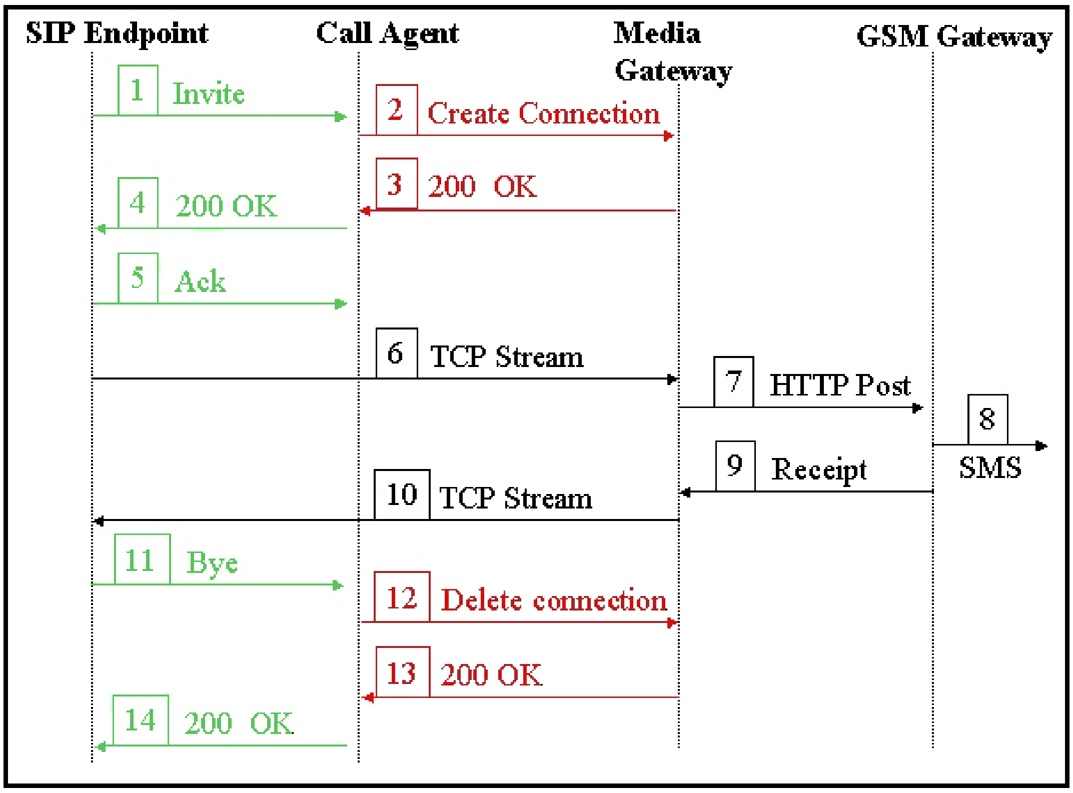


Figure 3.6 Diagram of Message sequence

3.3 Chapter Summary

This chapter described two SMS services that were developed to investigate the call control used by H.323, SIP and MGCP. Having described how the services work, chapter 4 looks in detail at the interworking issues between H.323 and MGCP, as well as those between SIP and MGCP.

Chapter 4

4.1 H.323 and MGCP

Having described how the H.323 and SIP SMS services work, this chapter looks in detail at the inter-working issues between H.323 and MGCP as well as those between SIP and MGCP.

Both H.323 and MGCP handle call signalling and control, but since they were developed by two different standards bodies and with different architectures, certain procedures or functions from one protocol do not map to those in the other protocol. In the case of H.323 and MGCP, H.323 has a huge set of functions compared to the meagre set possessed by MGCP. However since they both setup and tear down calls, a mapping of these functions is investigated in the development of the SMS service.

It has been suggested that procedure calls from one protocol need to be translated to another protocol through the use of some kind of gateway [Black, 2001]. In this case, since MGCP employs a decomposed gateway architecture, the translation is also decomposed. The CA handles the call control and the MG the media translation. As explained in 3.1.4, the CA uses MGCP to communicate with the MG. If a CA has to communicate with another protocol, in this case H.323, the CA incorporates the H.323 stack. It thus appears as another H.323 endpoint. This allows the CA to communicate seamlessly with all H.323 compliant endpoints.

4.1.1 The H.323 SMS Service

The prototype H.323 SMS service was developed to investigate call signalling and control between a CA and an MG as well as between H.323 and MGCP. The service was kept simple by using text as the media to be exchanged instead of voice or video. This allowed us to develop the service quickly and use the results to determine the outcome if voice or video was used. For the following section, please refer to Figure 3.4, which shows the messages exchanged in the SMS service.

To begin with, let us consider the MGCP messages between the CA and MG. Three MGCP messages were exchanged, *Create Connection*, *Delete Connection* and *200 OK*. With these three messages, the CA communicated effectively with the MG. All three messages carried SDP information that setup and took down the TCP stream.

The MG was modified to allow it make an HTTP post to the GSM Gateway. Even with this non standard behaviour, the gateway was controlled effectively with MGCP messages and remained compliant with the protocol. The MGCP messages are shown in Table 4.1.

	Command	**Explanation**

1	`CRCX 1000`
	`146.231.121.60 MGCP 1.0`
	`C: 007`
	`I: H323One`
	`M: sendrecv`
	`m=application 6000 UDP`
		The CA sends the MG a Create Connection (CRCX) message to create a connection on endpoint 146.231.121.60. The TransactionId is 1000. The CallId is 007. The ConnectionId is H323One. The mode is send and receive since the H.323 terminal has indicated it wants to receive the SMS receipt on port 6000.

2	`200 1000 OK`
	`I: H323One`
	`v=0`
	`c=IN IP4 146.231.121.141`
	`m=application 40000 UDP`
		The MG responds to the CRCX message with a 200 OK message that uses the same TransactionId (1000). The ConnectionId is H323One. The TCP Server has an IP address of 146.231.121.141 and supports data on port 40000.

3	`DLCX 2000 146.231.121.60`
	`MGCP 1.0`
	`C: 007`
	`I: H323One`
		The CA sends the MG a Delete Connection (DLCX) message to delete a connection on endpoint 146.231.121.60. The TransactionId is 2000. The same CallId (007) is used.

Table 4.1 MGCP messages for H.323 SMS service

The CA setup and tore down connections for endpoints and used SDP effectively to communicate the media information required for the sessions. One point that should be noted however is that SDP assumes that media will be transmitted using either RTP or UDP [Handley and Jacobson, 1998]. This is the reason the media type in row 2 of Table 4.1 is specified as UDP, although the messages transferred are TCP.

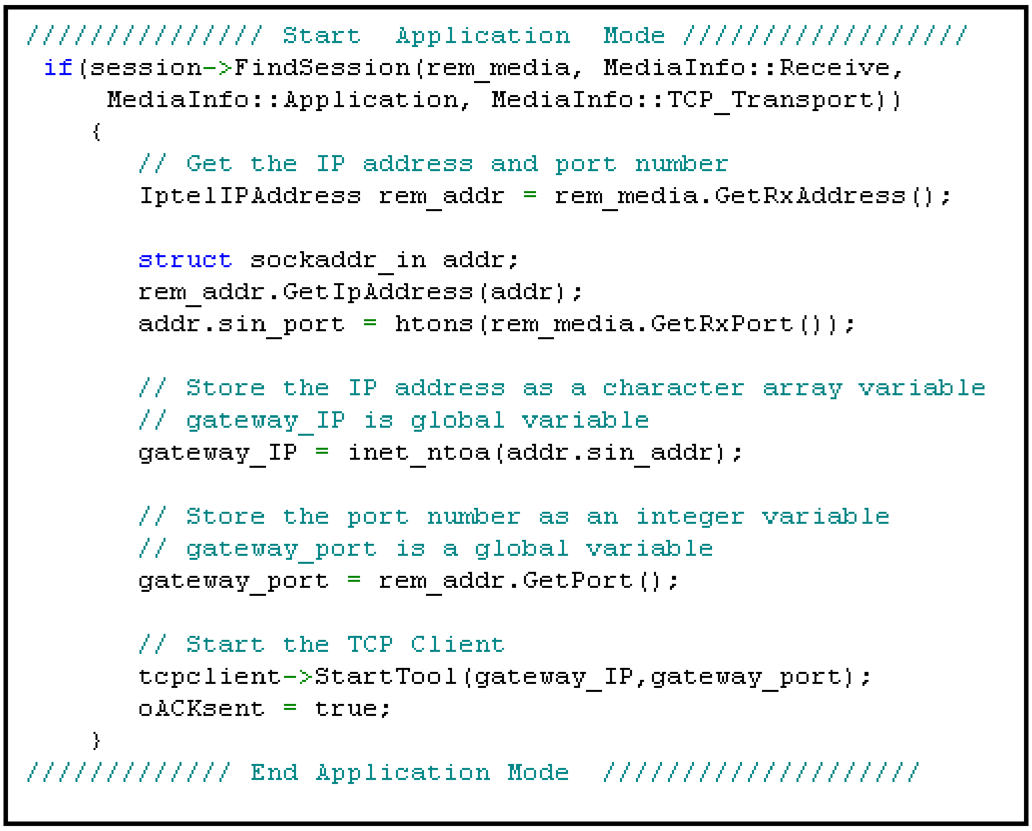
Let us now consider the H.323 messages. The H.323 messages transferred between the terminal and the CA are the standard messages that would be exchanged in an H.323 call using the Fast-connect procedure. The difficulty that arose was in the translation of the media information from H.323 to MGCP. This problem is twofold. Firstly, H.323 uses ASN.1 encoding for its messages increasing the complexity of retrieving the required parameters. Secondly, H.323 uses H.245 to describe its media capabilities while MGCP uses SDP.

The CA acted as a protocol translator between H.323 and MGCP to make the interworking between the two protocols appear seamless. Next, we consider what was required to extend the SMS service to SIP endpoints and how well SIP messages map to MGCP messages.

4.2. SIP and MGCP

4.2.1 Extending the SMS Service to SIP

As has been described in 3.2.2, modification to the CA was required. For the CA to communicate with SIP endpoints, the CA incorporated a SIP stack from CINEMA. As with the H.323 SMS Service, this made the CA appear to SIP endpoints as a SIP endpoint. In addition the SIP UA used with this service was modified to support text mode. Most of the code that was added is shown in Figure 4.1.



```
//////////////// Start  Application  Mode //////////////////
 if(session->FindSession(rem_media, MediaInfo::Receive,
     MediaInfo::Application, MediaInfo::TCP_Transport))
    {
       // Get the IP address and port number
       IptelIPAddress rem_addr = rem_media.GetRxAddress();

       struct sockaddr_in addr;
       rem_addr.GetIpAddress(addr);
       addr.sin_port = htons(rem_media.GetRxPort());

       // Store the IP address as a character array variable
       // gateway_IP is global variable
       gateway_IP = inet_ntoa(addr.sin_addr);

       // Store the port number as an integer variable
       // gateway_port is a global variable
       gateway_port = rem_addr.GetPort();

       // Start the TCP Client
       tcpclient->StartTool(gateway_IP,gateway_port);
       oACKsent = true;
    }
///////////// End Application Mode  ////////////////////
```

Figure 4.1 Code segment from SIP UA

When the SIP UA initiates a call, the method `OnCallEstablished()`is called. This method determines the type of media that is to be exchanged and starts the appropriate tool. The code segment shows the part of the method that checks if the media is of type ‘Application’. If this is the case, it starts the TCP Client. This modification of the CINEMA SIP UA is similar to the research done by [Hsieh, 2004] in the development of a SIP Alarm Service using CINEMA.

With the modifications done, the service was now complete. Figure 3.6 shows the messages that were exchanged between the SIP endpoint and the CA [Jacobs and Clayton, 2003].

4.2.2. Evaluating the SMS Service

Examining the call signalling, let us first consider the messages that were exchanged between the CA and the MG. As stated in Chapter 3, no modifications to the media gateway were required. This means that the messages exchanged between the CA and MG remained identical to those in the H.323 SMS Service.

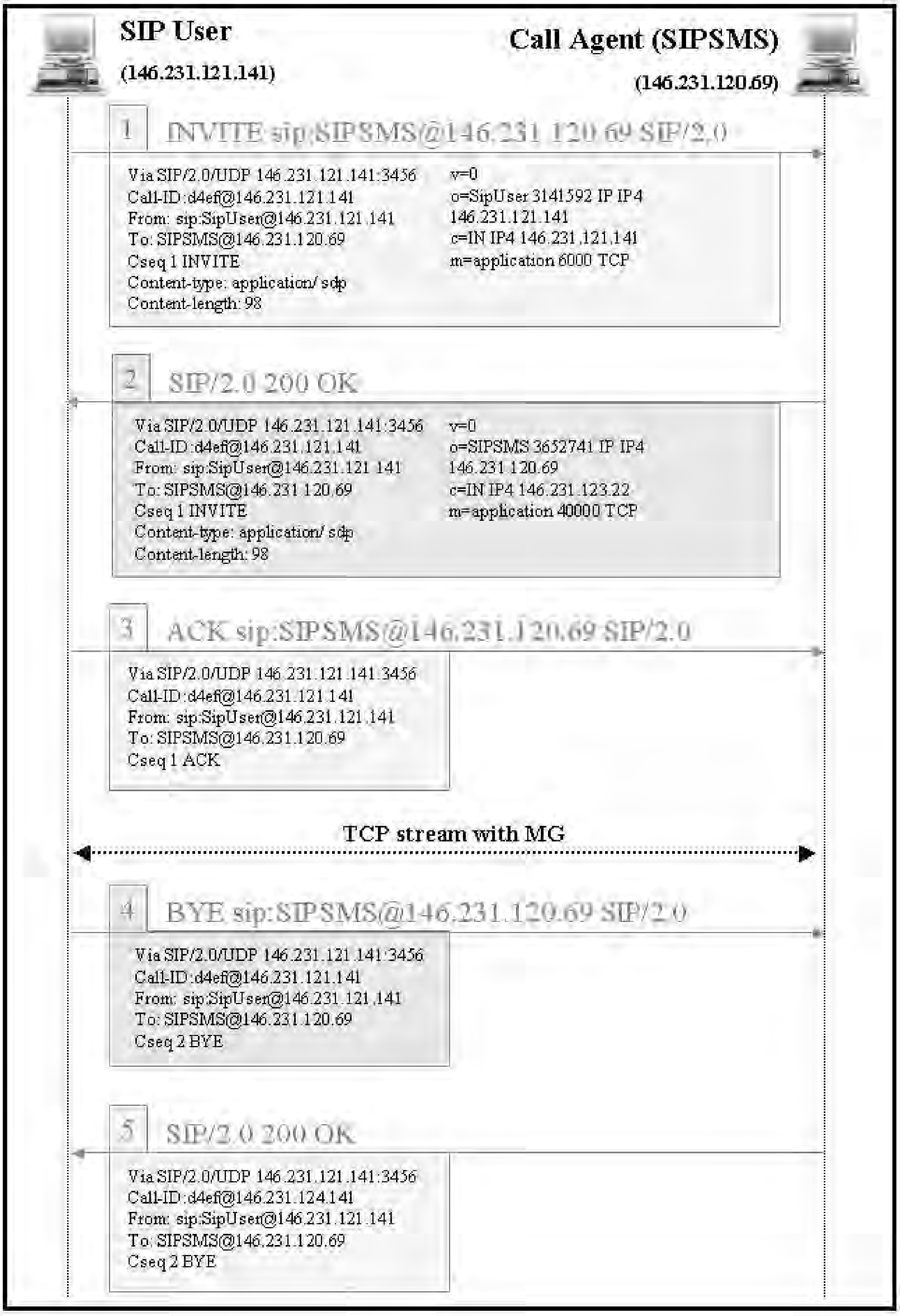
Now let us consider the messages between the CA and the SIP UA. Using standard SIP call signalling, the SIP UA initiates a call with the CA via an *Invite* message. It sends information about the type of session it wishes to have using SDP. The SDP specifies the media exchanges to be of type 'Application'. It also indicates the transport protocol to be used as TCP and the port on which it is listening. Since the CA also uses SDP to describe parameters in MGCP messages, no translation of the media information is required. The CA simply takes the SDP information from the invite message and passes it onto the MG in the *Create Connection* message.

The MG responds with a *200 OK* message that contains the IP address and port number of its TCP Server. The CA accepts the call from the SIP UA using a *200 OK* message that contains the media information from the MG. The SMS can now be sent. The SIP UA ends the call with a *Bye* message. The message exchange to tear down the call is as unproblematic as setting up a call.

SIP's and MGCP's use of SDP to describe media information made the development of this service easier than the H.323 SMS service. A detailed comparison of the two services will be carried out in 4.3 but first let us examine the SIP messages. Figure 4.2 shows the details of the SIP messages exchanged.

The following should be noted from Figure 4.2:

- ❖ The *Invite* message from the SIP UA specifies the session as type 'Application', and that it can receive TCP media on port 6000.
- ❖ The *200 OK* message from the CA to the SIP UA specifies the IP address of the MG as 143.231.123.22, the port as 40000 and the transport as TCP.
- ❖ The SIP UA acknowledges the *200 OK* message with an *Ack* message.
- ❖ When the SIP UA has finished sending SMS messages, it ends the session with a *bye* message to the CA.
- ❖ The CA responds with a *200 OK* message.



1 INVITE sip:SIPSMS@146.231.120.69 SIP/2.0

```
Via SIP/2.0/UDP 146.231.121.141:3456      v=0
Call-ID:d4ef@146.231.121.141              o=SipUser 3141592 IP IP4
From: sip:SipUser@146.231.121.141         146.231.121.141
To: SIPSMS@146.231.120.69                 c=IN IP4 146.231.121.141
Cseq 1 INVITE                             m=application 6000 TCP
Content-type: application/ sdp
Content-length: 98
```

2 SIP/2.0 200 OK

```
Via SIP/2.0/UDP 146.231.121.141:3456      v=0
Call-ID:d4ef@146.231.121.141              o=SIPSMS 3652741 IP IP4
From: sip:SipUser@146.231.121.141         146.231.120.69
To: SIPSMS@146.231.120.69                 c=IN IP4 146.231.123.22
Cseq 1 INVITE                             m=application 40000 TCP
Content-type: application/ sdp
Content-length: 98
```

3 ACK sip:SIPSMS@146.231.120.69 SIP/2.0

```
Via SIP/2.0/UDP 146.231.121.141:3456
Call-ID:d4ef@146.231.121.141
From: sip:SipUser@146.231.121.141
To: SIPSMS@146.231.120.69
Cseq 1 ACK
```

4 BYE sip:SIPSMS@146.231.120.69 SIP/2.0

```
Via SIP/2.0/UDP 146.231.121.141:3456
Call-ID:d4ef@146.231.121.141
From: sip:SipUser@146.231.121.141
To: SIPSMS@146.231.120.69
Cseq 2 BYE
```

5 SIP/2.0 200 OK

```
Via SIP/2.0/UDP 146.231.121.141:3456
Call-ID:d4ef@146.231.124.141
From: sip:SipUser@146.231.121.141
To: SIPSMS@146.231.120.69
Cseq 2 BYE
```

Figure 4.2 SIP messages between SIP UA and CA

4.3 Comparison of H.323 and SIP SMS Services

The two SMS services worked equally well with their respective endpoints. Therefore, in this comparison the emphasis is on the development of the two services. We look at the interworking issues between SIP and MGCP as well as H.323 and MGCP.

To begin with, looking at protocol complexity, the H.323 suite of protocols is much larger and more complex than SIP. In terms of development this meant a much steeper learning curve. As a result the development time required to develop the H.323 SMS service was longer than that required to develop the SIP SMS Service.

Another factor that affected development time was the mapping of session information from SIP and H.323 to MGCP. Since MGCP and SIP both use SDP to describe session information, the mapping between SIP and MGCP was seamless. Parameters from a SIP message could easily be passed to an MGCP message and vice versa. In addition, given that MGCP and SIP messages use text encoding, they can be easily debugged using a network-sniffing tool. In this deployment, Ethereal from GNU was used.

H.323 on the other hand uses H.245 to describe session information. Therefore, session information and parameters were not as easily transferred to SDP. Additionally, H.323 messages use binary encoding. Messages are encoded using two formats Q.931 or ASN.1 Packet Encoding Rules (PER). Therefore, to debug H.323 messages requires a Q.931 and ASN.1 PER decoding abilities. On the other hand, binary encoding produces smaller messages than text encoding, and depending on the implementation, can result in faster operating times.

Lastly, the mapping of H.323 and SIP messages to MGCP messages was considered. The question that arose was how well messages from SIP and H.323 would map onto MGCP messages in view of MGCP’s small message set? For MGCP to act as the common denominator for SIP and H.323, it would need a good mapping of SIP and H.323 messages to its own. In the development of the SMS Services, only a few of MGCP’s messages were required. Sessions were competently setup and torn down by the CA. Furthermore, MGCP's message set is equipped to handle any media changes or anomalies that may occur mid-call. Therefore, in terms of message mapping, the mappings between H.323 and MGCP messages and SIP and MGCP messages were adequate. MGCP's message set may be small, but it is adequate for its intended purpose. Table 4.2 shows a comparison between H.323 and SIP with regard to the development of the respective SMS Services.

Factors	H.323	SIP
Protocol complexity	Complex	Simpler
Development time	Longer	Short
Session Description	H.245	SDP
Protocol mapping to MGCP	Adequate	Adequate

Table 4.2 Comparison of H.323 and SIP with regard to development

4.4 Observations

Having developed the SMS services, several observations were made. From these observations certain conclusions can be drawn.

1. Choice of MGCP gateway over protocol specific gateway

In the development of these services an MGCP gateway was used instead of a protocol specific gateway. This choice proved to be a major saving in development time; as once the gateway had been developed for the H.323 SMS Service it did not require any modification to serve the SIP SMS Service. The only modification that was required was to the CA. Therefore this approach of using a generic gateway is valuable in heterogeneous networks.

2. MGCP gateway could replace H.323 gateway and SIP gateway

Taking it one step further, an MGCP gateway's intended use is to interface between different networks. Bearing this in mind, a major benefit of adding MGCP to a network would be to allow it to take over the inter-network interfacing. In a network with SIP and H.323, an MGCP gateway could provide interworking with the PSTN for these endpoints. This approach would make the H.323 gateway redundant. Furthermore as explained by [Jacobs and Clayton, 2002], MGCP's Master/Slave approach to gateways scales better than H.323's monolithic approach especially in larger networks. In other words, as the media translation demands on a gateway increase, it is much simpler to add an additional MGCP media gateway under the control of an existing CA, than to add an additional H.323 gateway. The added MGCP media gateway would provide the required media translation without the overhead required for an additional H.323 gateway.

3. SIP SMS service mirrors H.323 SMS Service

In developing these services, it was found that the SIP SMS Service mirrors the H.323 SMS service. This means that the only thing that was changed to allow SIP endpoints to send SMS messages was the CA as had been predicted. The MG for both services remained unchanged. The SIP and H.323 SMS services are shown running concurrently in Figure 4.3. Since data is being transferred here instead of voice, the RTP streams depicted in Figure 4.3 are representative of the TCP streams used for the service.

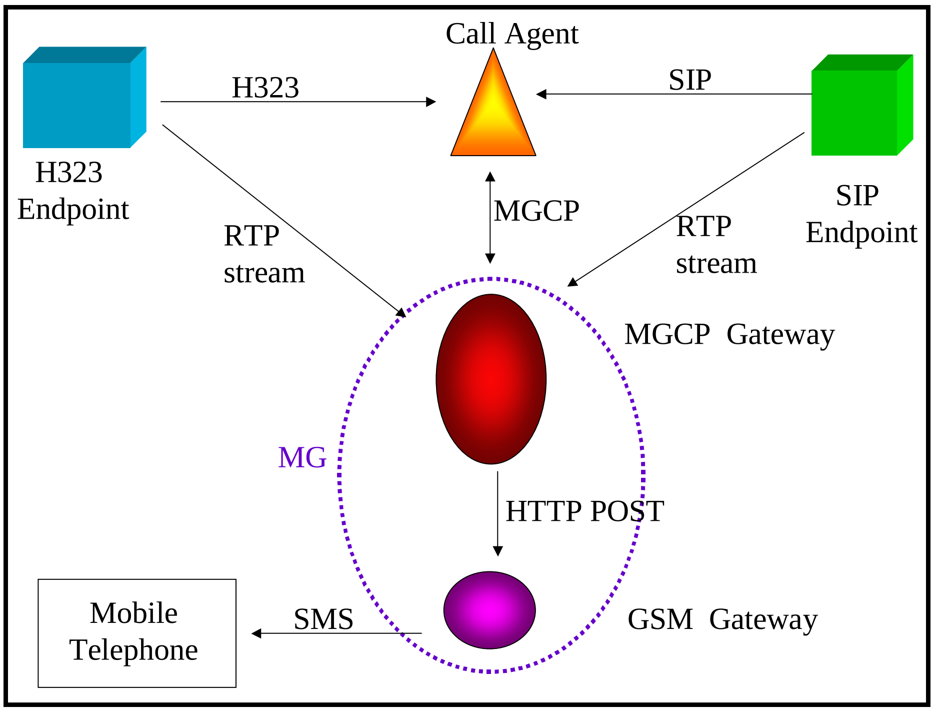


Figure 4.3 Mirrored SMS Service

The significance of the service being mirrored is that no additional modifications or enhancements were required for either service. This in turn means that SIP and H.323 interface with MGCP in a standard or predictable way. Thus to add services or media types would not be difficult. Therefore the RTP streams in Figure 4.3 could be representative of data, audio or video.

4. MGCP Gateway used as an interworking gateway between SIP and H.323

Figure 4.3 depicts the CA as a protocol translator. The CA translates SIP and H.323 messages to MGCP messages. However, if the CA has both the SIP and H.323 protocol stacks could it not work as an interworking function between the two protocols? This will be the study of Chapter 5.

4.5 Chapter Summary

This chapter looked at the issues encountered in the development of the H.323 SMS service in 4.1 and the extension of the service to SIP endpoints in 4.2. The development issues of the two services were then compared in 4.3. This was done for two reasons. Firstly, it evaluated the difficulty of service creation using SIP and H.323. Secondly, it explored the interworking issues that SIP and H.323 have with MGCP. The chapter ends with a number of positive remarks on the use of MGCP for service creation. However, the use of MGCP is not appropriate for every situation or service for that matter. To investigate this, chapter 5 pushes the limits of MGCP even further, by investigating its use as a protocol translator between SIP and H.323.

Chapter 5

5.1 SIP & H.323 interworking field trial

The previous chapter examined call control between H.323 and MGCP, and between SIP and MGCP. It showed MGCP's use as a unifying protocol that facilitates the addition of services with a minimum amount of effort. A good understanding of basic call control between SIP, H.323 and MGCP was attained. Furthermore, according to [Varshney *et al*, 2002], MGCP can be used as part of H.323 to simplify interworking. Using this understanding, it was felt that an investigation into the use of MGCP to aid the interworking of SIP and H.323 would be in order.

5.2 Background

H.323 is currently the most widely used call processing protocol in the industry, while carrier networks using IP telephones seem to be built based on SIP [Black, 2001]. Therefore an interworking between the two protocols is desirable in order to achieve universal connectivity. Interworking will include two types of endpoints: H.323 terminals and SIP user agents and a SIP-H.323 Interworking Function (IWF). Other entities may include H.323 gatekeeper (GK), and SIP servers.

An IWF translates messages from one protocol to another and vice versa, allowing the two protocols to communicate. Therefore, if a service is developed for one protocol, by using an IWF, the same service can be made available to another protocol without the need for any modification. In the SMS services deployed in the previous chapter, the provision of an IWF would make it possible to develop the service for either SIP or H.323 and it would be available to both protocols without the need for modification. Obviously, the interworking between the two protocols depends on the effectiveness of the IWF.

Since there are inherent differences between H.323 and SIP, allowances must be made to facilitate the interworking between the two protocols. Instead of concentrating on one standard versus another, the voice/video over IP community is working on better ways of ensuring interoperability between standards to provide end-to-end connectivity throughout the network and to offer the value-added IP-centric services that will demonstrate the power of IP-based communications [Wang, 2002]. Programmers and companies who used to work with H.323 software are contributing to this project to ensure that there is interoperability between the H.323 network and the SIP network [Hsieh, 2004].

The specifications for interworking are given in the SIP-H.323 Interworking Internet Draft [Agrawal *et al*, 2001]. Based on this draft, research and development has gone into this area at Rhodes University. Ming Hsieh [Hsieh, 2004], has looked at the IWF provided by VOCAL and CINEMA. He concluded that, as the complexity of the services increases, so too do the number of messages the IWF has to translate. Firstly, this complex translation may introduce delays in the call as the IWF has many more messages to translate. Secondly, it requires an accurate and exact mapping between the protocols and where this is not possible, translation becomes difficult if not impossible. An example of this would be a service that has been implemented for H.323 and is not translatable to SIP because SIP has left out a required function.

5.3 MGCP's use as an IWF

In view of the fact that the SIP-H.323 Interworking Internet Draft [Agrawal *et al*, 2001] specifies the requirements for the IWF and not a specific protocol, many implementations into this area are being researched. In addition, since both protocols operate over IP and use RTP for transferring audio and video media, the task of interworking is reduced to the translation of signalling and session description [Singh and Shultzrinne, 2000]. The IWF is thus best performed by a signalling gateway. It is with this in mind, that MGCP's decomposed architecture of a signalling CA and media translation MG was thought to be worth investigating to be used as an IWF.

The next section, 5.4, introduces some issues faced by IWFs. This is followed in 5.5 by the field trial of MGCP as an IWF and how it addresses the issues raised in 5.4.
5.6 evaluates the MGCP IWF. Lastly, some observations and recommendations are made in 5.7.

5.4 IWF issues

Before developing the MGCP IWF, the author felt that it was appropriate to consider the general issues faced by IWFs. This way, the deployment of the IWF could attempt to address these issues in addition to performing the function of interworking between SIP and H.323. While, attempting to address all the issues is beyond the scope of this thesis, two issues that stood out are call setup and user registration. These issues are considered next.

5.4.1 Call Setup

Chapter 2.2 and 2.3 introduce call setup for SIP and H.323 respectively. What should be noted is the way in which they address the three essential elements for a call, namely the signalling destination address, the local and remote media transport addresses and the local and remote media capabilities. SIP determines this information using an *Invite* message and the response it receives. H.323 on the other hand collects this information over a number of messages. Therefore in terms of translation, a SIP call can easily be translated to an H.323 call, since a SIP *Invite* message contains the three essential elements which can then be spread across the various H.323 messages. On the other hand, translating a call from H.323 to SIP is not as easy, since the various elements have to be merged into a single SIP *Invite* message [Singh and Shultzrinne, 2000]. Figures 5.1 and 5.2 illustrate this point.

Figure 5.1 shows a call between a SIP UA and an H.323 terminal. The following should be noted from figure 5.1. Once the SIP UA sends the *Invite* message, it simply waits for the IWF to setup the call with the H.323 terminal. Once the IWF receives message 5 the *Connect* message from the H.323 terminal, the call has been accepted by the H.323 endpoint. Messages 6 to 16 are H.245 messages to determine the capabilities of the endpoints, master and slave and the channels for the media.

The master-slave determination procedure uses two pieces of information. The first is a terminal type value, based on the terminals capabilities, and the second is a random number. The terminal with the higher terminal type value becomes the master. For example, a basic terminal would have the lowest terminal type value and a MCU that is managing a conference would have the highest [Collins, 2001]. If the terminals had the same terminal type value, then they would select a random number. The higher random number would determine the master. In Figure 5.1, the IWF would have a higher terminal type value than the H.323 endpoint and therefore would be master.

Once the IWF has this information it accepts the call from the SIP UA with a *200 OK* message containing the SDP information. The SIP UA acknowledges this with an *Ack* message and this concludes the signalling required to establish a call.

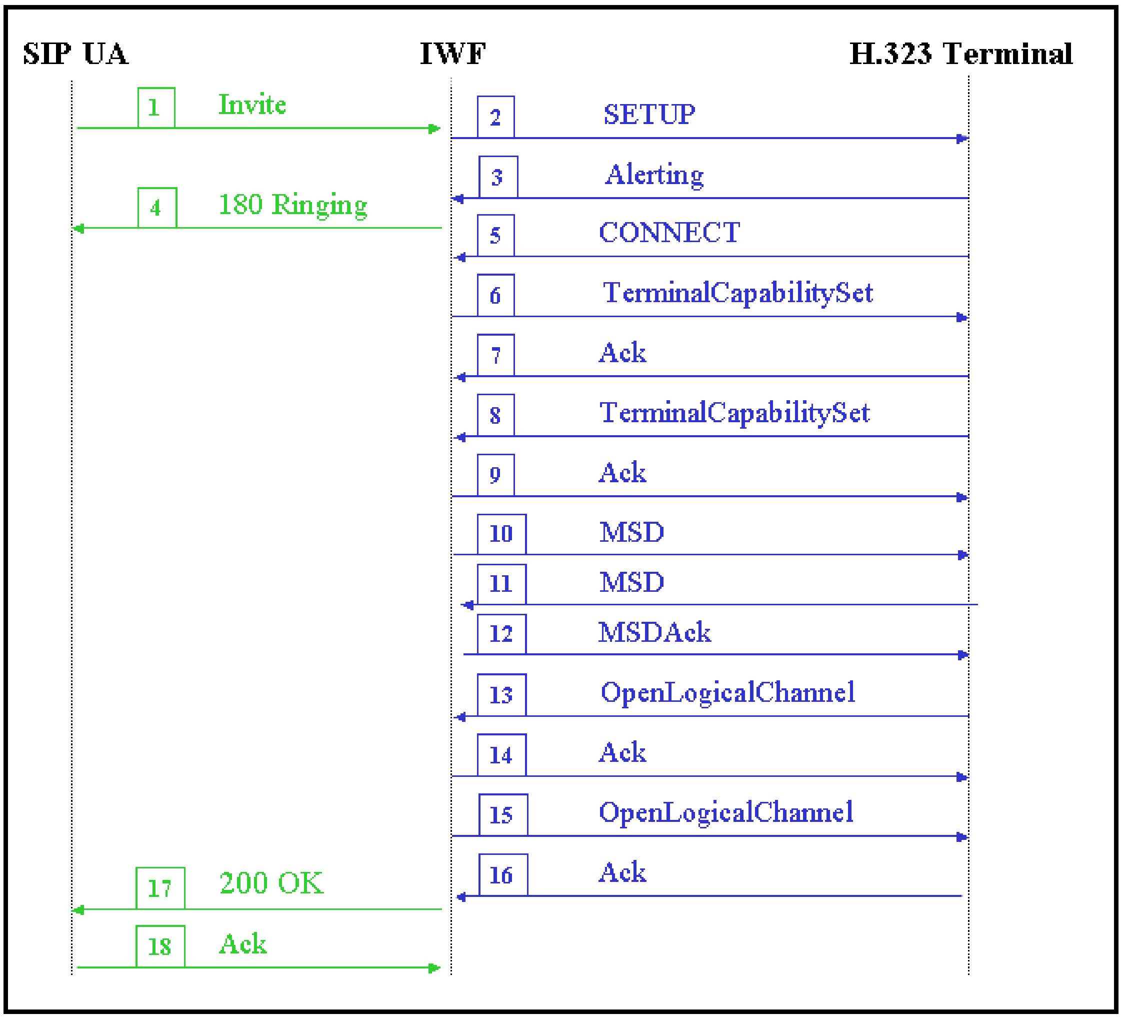


Figure 5.1 SIP to H.323 call

Figure 5.2 illustrates a call between an H.323 terminal and a SIP UA. The H.323 terminal uses Q.931 messages 1 to 8 to initiate the call. The SIP UA accepts the call, but does not have the necessary session information for the media. This is because

H.323 exchanges the media information using H.245 messages. Messages 10 to 20 are H.245 messages that are exchanged between the H.323 terminal and the IWF. Once the IWF has the media information it reinitiates a call with the SIP UA with an *invite* message containing this information. The SIP UA now has all the necessary information to setup a call.

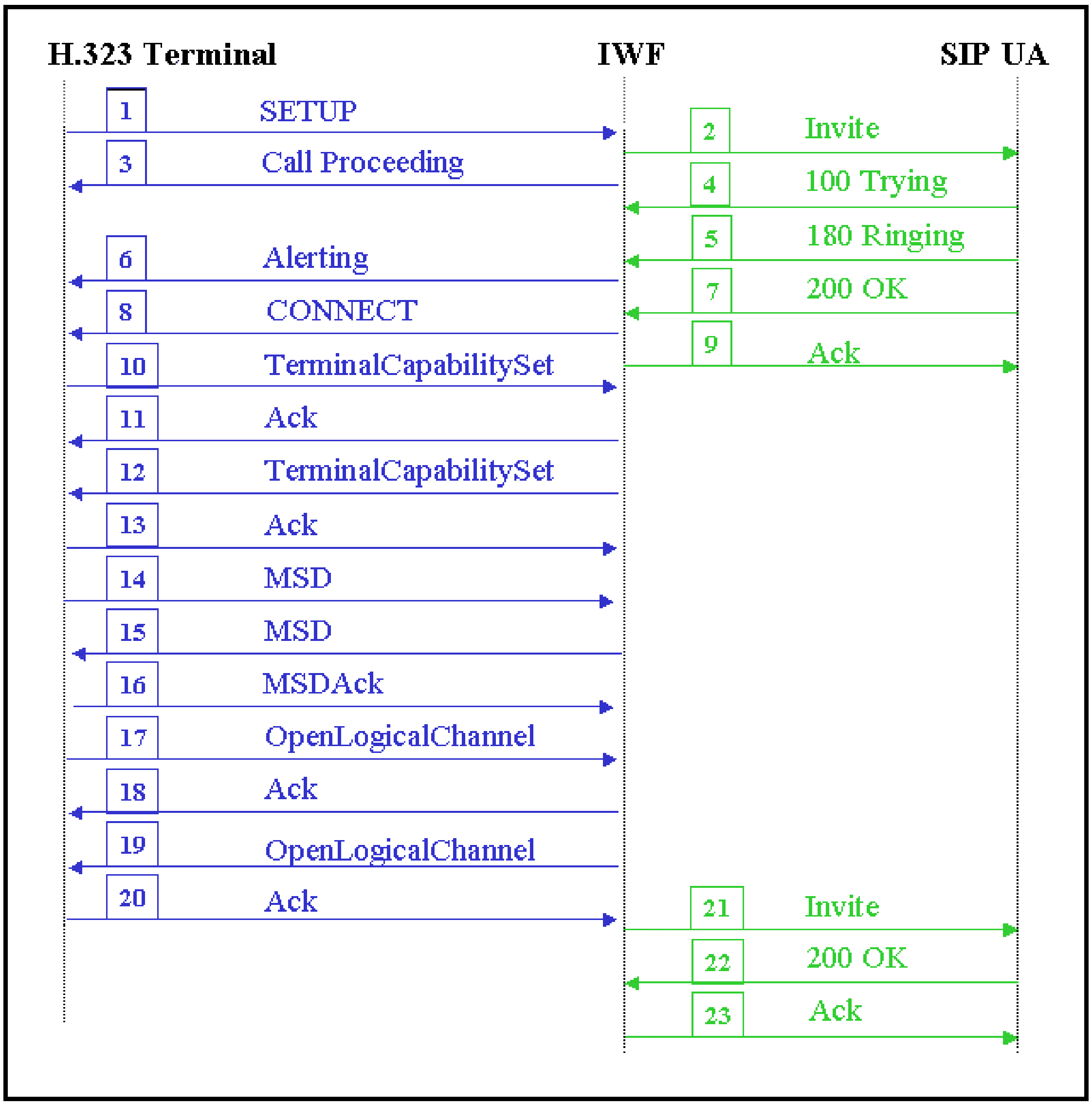


Figure 5.2 H.323 to SIP call

The number of messages required to setup a call can be greatly reduced by using H.323's Fast-connect procedure. We explore this in the deployment of an IWF.

5.4.2 User registration

User registration is the mapping of user names, telephone numbers or email addresses to network addresses. This mapping is stored in a registrar server in the case of SIP or in a Gatekeeper in the case of H.323. A user Susan may register using a location independent identifier for example susan@ru.ac.za. If another user calls Susan using SIP for example, the call will be addressed as sip : susan@ru.ac.za. A proxy server will then access a registrar server to find out where Susan is registered and direct the call to the appropriate IP address. In order to facilitate calls from a SIP endpoint to an H.323 endpoint or vice versa, an IWF needs to access the registration information from both networks. We suggest ways to address this issue in the next section.

5.5 Implementation of the IWF

In this section we describe our implementation and deployment of an IWF using MGCP. We then explain how the issues raised in 5.3 were addressed by our implementation. Finally, we assess the effectiveness of this solution and make some recommendations on MGCP's use in the development of an IWF between SIP and H.323.

5.5.1 The environment

The operating systems used are Windows 2000 and Red Hat Linux 7.1. Windows 2000 is the platform used for the SIP environment CINEMA. Linux is the platform used for the MGCP stack from Vovida as well as the open source H.323 stack. The SIP UA used is from the CINEMA package and the H.323 terminal is an open source H.323 terminal. A CA was modified to incorporate both the SIP and H.323 stacks. From here on, this modified CA will be referred to as either MGCP CA or IWF.

Figure 5.3 shows an overview of the IWF. Since it incorporates an H.323 stack as well as a SIP stack, it appears as a SIP UA to SIP endpoints and an H.323 terminal to

H.323 endpoints. Next, various call scenarios are explored to determine the effectiveness of using an MGCP CA as an IWF.

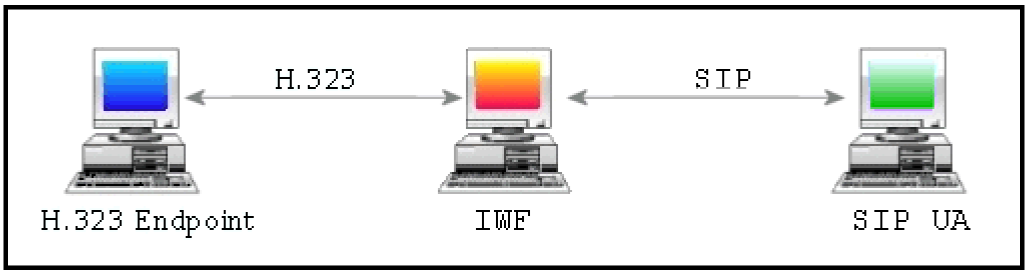


Figure 5.3 MGCP CA used as an IWF between H.323 and SIP

5.5.2 Call Scenarios

A call from a SIP UA to an H.323 terminal was made successfully using the message sequence shown in figure 5.1. A call was then made from an H.323 terminal to a SIP UA using the message sequence shown in figure 5.2. As expected, this call was more complicated than the SIP to H.323 call. Additionally, the call was lengthier to setup because the mapping is not one-to-one. In an attempt to simplify the H.323 to SIP call, the Fast-connect procedure was considered.

From Chapter 3.1.4 the Fast-connect procedure had been used to speed up the deployment of the H.323 SMS service, by reducing the number of messages required to use the service. Here, it was thought that using the Fast-connect procedure would improve the mapping between SIP and H.323. This procedure is not always used since it is an optional H.323 version 2 feature and H.323 version 1 equipment does not support it. However, support of the Fast-connect procedure is specified as a requirement of an IWF [Schulzrinne & Agboh, 2004]. Thus, a call from an H.323 Terminal to a SIP UA using H.323's Fast-connect procedure was established as is shown in Figure 5.4.

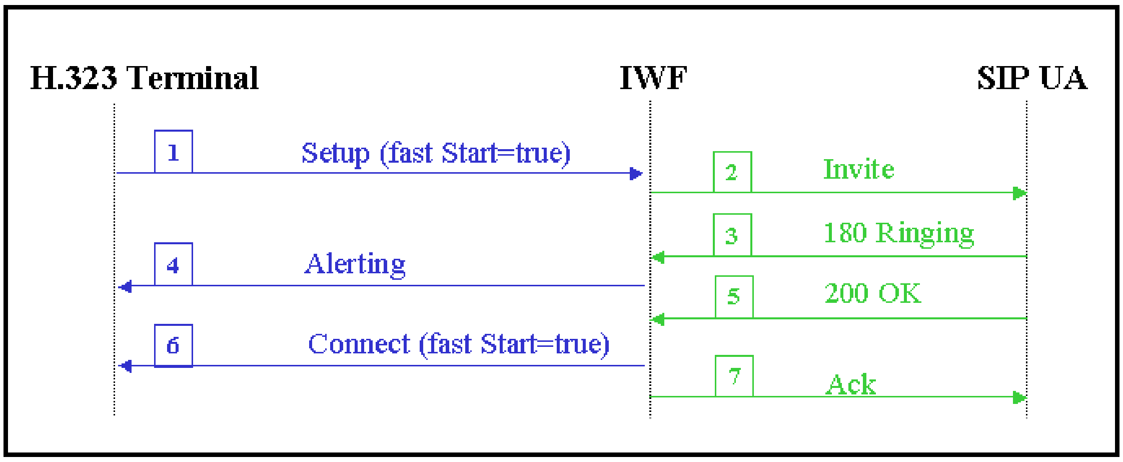


Figure 5.4 H.323-SIP call using H.323 V2 Fast-connect

The call flow shown above offers the best mapping between H.323 and SIP. This is because the *Setup* message contains the local address, media capabilities and the ports that the H.323 endpoint is listening on. The IWF passes this information on to the SIP UA using an *Invite* message that contains the media and port information as part of the SDP. Once the SIP UA receives the *Invite* message, it responds with a *180 Ringing* message. The IWF translates this into an *Alerting* message and sends it to the H.323 endpoint. The SIP UA then replies to the *Invite* message with a *200 OK* message if it wishes to accept the call. The SDP of the *200 OK* message contains information about the media and ports the SIP UA is listening on. The IWF translates this information into a *Connect* message and passes it on to the H.323 terminal. The *Connect* message contains the SIP UA's media and ports information. Both endpoints now have the three elements, mentioned in 5.4.1 that are necessary to setup the call. The IWF sends the SIP UA an *Ack* message and the call signalling necessary to setup a call is now complete. An RTP stream is initiated from the H.323 terminal to the SIP UA and the call proceeds. The messages exchanged are shown in Appendix 2.

Using H.323's Fast-connect procedure affords a one-to-one mapping between H.323 and SIP using the least number of messages. Although this is one solution, it is not the ideal one, as it does not cater for H.323 terminals that do not support Fast-connect. Hence, a more generic solution like the one shown in figure 5.2 is more appropriate. The use of the Fast-connect procedure here is useful in testing the MGCP CA's ability to handle basic call setup. Next, we explore how the IWF handles endpoints that are registered on different networks.

If a call is initiated from an endpoint in the SIP network to an endpoint in the H.323 network, the call has to be routed and translated by the IWF. This scenario examines how calls are routed from one network to another, as was raised in 5.4.2, and offers a solution.

5.5.3 Registration Issues

A user that registers with a network and wants to call a user on another network needs to have the call routed and translated in order to contact the other user. For example if a SIP user wants to contact an H.323 user, how does the SIP proxy / registrar contact the H.323 gatekeeper to look for the user? If the IWF is to perform the translation between the two servers, it needs to be able to access user registration information from both networks. Three approaches have been suggested [Singh and Shultzrinne, 2000]. One approach is to have the IWF contain an H.323 gatekeeper. The second approach is for the IWF to contain a SIP proxy / registrar. The third approach is for the IWF to contain neither. These three approaches are considered and the best one is deployed. For the remainder of this thesis, a SIP registrar server that is co-located with a SIP proxy server shall be collectively referred to as the SIP proxy.

5.5.3.1 IWF contains an H.323 gatekeeper

This approach combines the IWF with an H.323 gatekeeper and shall be referred to as the IWF-Gatekeeper. In this approach the SIP proxy / registrar server maintains the registration information for both the SIP endpoints as well as the H.323 endpoints. A SIP UA registers with the SIP proxy / registrar using a *Register* message. When an H.323 Terminal registers with the IWF-Gatekeeper, it sends an *RRQ* message. The IWF-Gatekeeper translates the H.323 address into a SIP URL and sends a *Register* message to the SIP proxy after having translated the H.323 Alias Address into a SIP URL as is shown in Figure 5.5.

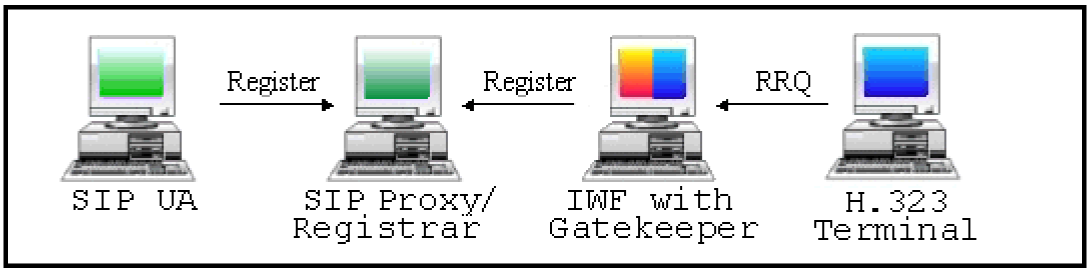


Figure 5.5 Registration of SIP and H.323 endpoints

If an H.323 terminal wants to talk to an entity on the SIP network, it sends an *ARQ* message to its gatekeeper. If its gatekeeper is not the IWF-Gatekeeper, it broadcasts an *LRQ* message to all the other gatekeepers. When the IWF-Gatekeeper receives this request, it sends an *OPTIONS* message to the SIP proxy. If the user is available, the IWF-Gatekeeper responds with an *LCF* message.

The drawback to this approach is that the SIP proxy has to maintain all the user registration information for both the SIP and H.323 networks. The advantage to this approach is that even if the H.323 network is large, and there are a number of gatekeepers, as long as one of them has an IWF, endpoints in the SIP network can be reached by using the *LRQ* messages. The next approach offers an alternative solution, but also potentially has more drawbacks.

5.5.3.2 IWF contains a SIP proxy and Registrar

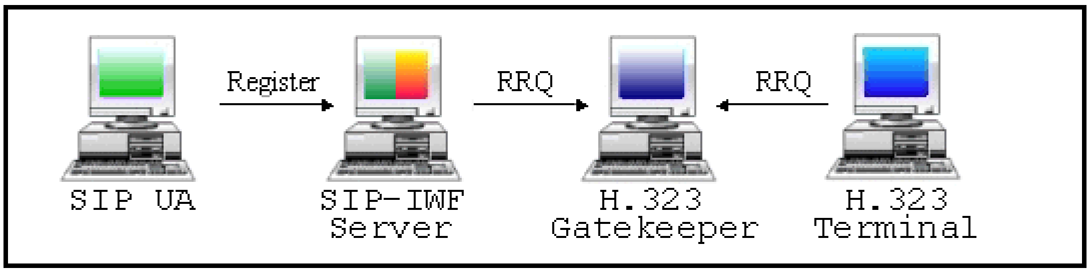


Figure 5.6 Registration of SIP and H.323 endpoints

This approach combines the IWF with a SIP proxy and registrar server and will be referred to as the SIP-IWF Server. In this approach the H.323 gatekeeper maintains the registration information for both the H.323 and SIP networks. H.323 endpoints register with the Gatekeeper in the normal manner by sending an *RRQ* message. When the SIP-IWF Server receives a SIP *Register* message, it sends an *RRQ* message to the H.323 gatekeeper after having translated the SIP URL into an H.323 Alias Address as is shown in Figure 5.6. Thus H.323 endpoints can contact SIP endpoints, since the SIP registration information is also held by the H.323 gatekeeper. For a call from the SIP network to the H.323 network, the SIP endpoint sends an *Invite* message to the SIP-IWF Server. The SIP-IWF Server sends an *LRQ* message to the H.323 gatekeeper. The H.323 gatekeeper responds with the IP address of the H.323 endpoint. The SIP-IWF server can then route the call.

The main drawback to this approach is that the H.323 gatekeeper has to keep the registration information of the H.323 endpoints as well as the SIP endpoints. Furthermore, in larger networks with more than one SIP Registrar, SIP UAs that are not registered with the SIP-IWF Server will find it difficult to communicate with H.323 terminals. This is because a call from a SIP UA not registered with the SIP-IWF Server will have to be relayed from proxy to proxy using the VIA operation [Black, 2001] until it reaches the SIP-IWF Server. A location request can then be relayed to the H.323 gatekeeper. The next approach aims to avoid saddling either the H.323 gatekeeper or the SIP proxy with user registration.

5.5.3.3 IWF is independent of H.323 gatekeeper and SIP proxy

This approach seeks to keep the IWF separate from the SIP proxies and H.323 gatekeepers. This means that endpoints on the SIP network register with a SIP proxy/registrar server and endpoints on the H.323 network register with the H.323 gatekeeper as they would normally. The IWF is capable of receiving and sending *LRQ* messages from the H.323 network. It is also capable of sending and receiving *OPTIONS* messages. This gives the IWF some of the abilities of an H.323 gatekeeper and a SIP proxy.

Having considered the two previous scenarios, the author is of the opinion that this approach best addresses the issue of user registration. This is because the approaches in 5.5.3.1 and 5.5.3.2 put additional strain on the SIP proxy and H.323 gatekeeper respectively, by requiring each of them to maintain the user registration information for both networks. Therefore, by keeping the IWF independent of the SIP proxy and H.323 gatekeeper, there is no additional strain put on these servers. Consequently, the prototype MGCP CA was modified to include this functionality. Having setup the IWF to listen to both networks, calls are now attempted from both sides. The merits and drawbacks of this implementation will then be discussed.

H.323 to SIP call

If an H.323 endpoint wishes to contact a SIP endpoint, it sends an *ARQ* message to its Gatekeeper. Since the SIP endpoint is not registered with the H.323 gatekeeper, the H.323 gatekeeper broadcasts an *LRQ* message to find the address of the SIP endpoint. This message is picked up by the IWF which proxies it to the SIP proxy as an *Options* message. If the user is registered with the SIP proxy, it sends the *Options* message to the user. The user responds with its capabilities. This message is relayed by the SIP proxy to the IWF. The IWF then sends an *LCF* message to the H.323 gatekeeper with the IP address of the SIP endpoint. The H.323 gatekeeper then sends the H.323 endpoint the IP address of the SIP endpoint. The H.323 endpoint then initiates a call with the SIP endpoint via the IWF. It may initiate a standard call as shown in Figure 5.2 or one using the H.323 Fast-connect procedure as shown in Figure 5.4 since it now knows the media capabilities of the SIP endpoint.

SIP to H.323 call

A call from the SIP network to the H.323 network is pretty much the H.323 to SIP call in reverse. The SIP endpoint is registered with the SIP proxy and the H.323 Terminal is registered with the H.323 gatekeeper. The SIP endpoint initiates a call using an *Invite* message, which it sends to the SIP proxy. If the callee is not registered with the SIP proxy, the SIP proxy is setup to forward the message to the IWF. When the IWF receives the *Invite* message it translates the address and broadcasts an *LRQ* message. This message is picked up by the H.323 gatekeeper, which responds with an *LCF* message containing the H.323 Terminal’s IP address. The IWF then sends the H.323 Terminal a *Setup* message containing the relevant information based on the SIP *Invite* message. The rest of the call setup is the same as the call scenario shown by Figure 5.1. Figure 5.7 shows the IWF bridging the SIP network with the H.323 network by listening to both networks.

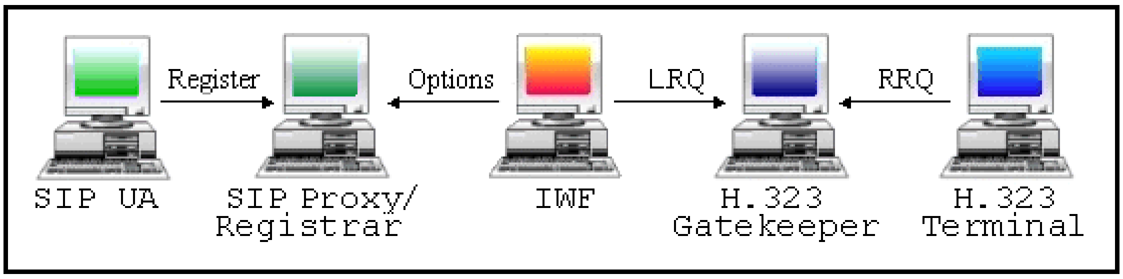


Figure 5.7 IWF is independent of SIP and H.323 networks

Evaluation of this deployment

In our implementation, the IWF works well in translating calls from the H.323 network to the SIP network and vice versa. This approach is more effective than the approaches described in 5.5.3.1 and 5.5.3.2 in the following ways:

- ❖ The H.323 gatekeeper and the SIP proxy are not burdened with additional registration information.
- ❖ Only the necessary location request information is sent from one network to the other.

This deployment of an IWF meets some of the basic requirements specified in the SIP-H.323 Interworking Requirements Draft [Schulzrinne & Agboh, 2004] that a signalling gateway must meet to qualify as an IWF. At this point, the experimentation has provided sufficient results for us to draw conclusions about MGCP applicability to interworking between SIP and H.323. This will be discussed in the next section.

5.6 MGCP's applicability to interworking between SIP and H.323

H.323 and SIP use similar mechanisms to transport their media. The main difference between the two protocols is their call signalling. Thus, MGCP's decomposed architecture seemed to be ideally suited to the task of interworking between SIP and H.323. Since the MGCP CA handles call signalling while the MG handles the media, it was thought that the CA could be used to perform the call signalling between SIP and H.323 endpoints and communicate with the MG for calls with the PSTN.

Call Setup

In the development of the IWF, the MGCP-CA was modified to incorporate a SIP and H.323 stacks. This allowed the IWF to process calls from SIP to H.323 and vice versa. From 5.1, we can see that calls are easily translated from SIP to H.323. The difficulty is in translating calls in the other direction. After attempting the call as shown in Figure 5.2 and realising the numerous messages that have to be exchanged, the call was re-attempted using the Fast-connect procedure. This procedure produces the best mapping between H.323 and SIP.

Using the Fast-connect procedure simplified the call setup from H.323 to SIP. However, as long as the IWF has to cater for H.323v1 and H.323v2 equipment, the message sequence shown in 5.2 will have to be used.

User Registration

Having proved the IWF's ability to handle basic call setup in both directions, the next issue that was explored was a sequence of calls from the SIP network to the H.323 network and vice versa. Having explored three possible solutions, our deployment showed that the IWF could be used effectively in translating calls from SIP to H.323 networks and vice versa. Our deployment proved to be the best of all worlds. It required the least amount of modification, yet produced an elegant solution that handled translation, scaled well, and did not put undue pressure on the SIP proxy or H.323 gatekeeper.

At this point, the MGCP-CA could handle calls between SIP and H.323 endpoints as well as between their respective networks. However, the MGCP-CA did not use any MGCP to provide this functionality. By incorporating the SIP and H.323 stacks, the MGCP-CA acted as an IWF. Thus, while the MGCP-CA could in effect perform as an IWF, MGCP had nothing to do with it. Furthermore, the media exchanged by the endpoints did not require any manipulation, nor did it pass through the MG.

It is true that MGCP can be the Lingua Franca for call signalling protocols, but this functionality is only appropriate or relevant when used in conjunction with the MG as was done with the SMS services in Chapter 3. Otherwise, MGCP is not providing anything more than a signalling gateway would. It was with these reasons in mind, that the development of the IWF was halted at this point.

5.7 Remarks and Observations

There are a variety of VoIP networks in existence. They contain MGCP, SIP, H.323 or a combination of the three. Depending on the VoIP needs of the users these networks continually change and advance. However, the co-existence of legacy equipment cannot be overlooked and must be kept in mind when developing these networks. As a result heterogeneous telephony networks will exist for a while as will the need for interworking between them.

MGCP was developed to allow communication between the PSTN and IP networks. H.323 networks and SIP networks can communicate with the PSTN, without MGCP. However a network that contains both SIP and H.323 would need two gateways to communicate with the PSTN and a signalling gateway for the interworking between SIP and H.323.

The development of the SIP and H.323 SMS Services in Chapter 3 and 4 show the seamless communication between MGCP and the two protocols. Therefore it would be able to replace the two PSTN gateways with one MGCP gateway. Furthermore as the network grows, MGCP's decomposed gateway architecture is thought to scale better than the monolithic gateway approach [Jacobs and Clayton, 2002].

This chapter looked at the possibility of using an MGCP CA to function as an IWF between SIP and H.323. In the development of the IWF to meet the SIP-H.323 Interworking Requirements Draft [Schulzrinne & Agboh, 2004], it was found that MGCP has very little if anything to contribute to the IWF. An MGCP CA could be used to perform the IWF, but apart from reducing the number of network servers needed to provide interworking, it does not offer anything significantly more than a signalling gateway would offer. It is the author's opinion that using an MGCP CA to perform the SIP-H.323 IWF does not offer any significant advantages over using a signalling gateway, and does not recommend MGCP for this purpose. MGCP can be used as the lingua franca for the creation of services for SIP and H.323, but is not recommended for use as an interworking function.

MGCP’s small message set has made interworking with the PSTN simple. However this small message set also limits its ability to interwork with other types of networks. MGCP’s ability to interwork with SIP and H.323 to aid service creation as has been described in Chapter 4; and is the extent to which MGCP’s limits can be pushed.

To date, the SIP-H.323 Interworking Requirement Draft [Schulzrinne & Agboh, 2004] is awaiting RFC status. As a result, several implementations of an IWF exist. In recent times, softswitches and soft PBXes have offered solutions to the interworking of SIP and H.323. One of the most popular open source platforms for converged telecommunications worth mentioning is Asterisk. Asterisk began as a soft PBX and has since grown to allow different types of IP telephony hardware, middleware and software to interface with each other consistently. Asterisk uses host processing for TDM and DSP, a lightweight protocol (IAX) for packet voice, and a flexible application-centric architecture for PBX services. It also provides interoperability with other VoIP protocols such as SIP, MGCP, and H.323 [Manesh, 2004]. As such, Asterisk can perform the protocol translation functions of an IWF.

Softswitches and soft PBX are gaining popularity based on their flexibility and associated cost saving. With additional features like protocol translation to boot, it is no wonder that they are receiving attention from all sectors of the market. As different architectures and solutions attempt to solve the SIP-H.323 interworking problem, only time will tell which solution suits which situation best. Chapter 6 sums up the research and makes some overall observations and conclusions.

Chapter 6

This research set out to investigate the use of MGCP within the VoIP environment, through the use of field trials. The H.323 and SIP SMS Services investigated the inter-working of H.323 and SIP with MGCP. To begin with, this chapter will compare the factors that affected the development of the H.323 and SIP SMS Services. The interworking of SIP and H.323 with MGCP will then be described.

The experimental use of MGCP as an interworking function between H.323 and SIP was investigated in Chapter 5. The observations and recommendations from this investigation will be summarised. In conclusion MGCP's applicability to service creation, its place in the network and the future of GCPs will be discussed.

6.1 Observations

The SMS Services

The main focus of this research was investigating MGCP’s call control. However, since this was done through the development of services, some factors that affected the development of these services are worth noting. Chapter 4 describes the development of the two services; therefore in this chapter just the factors that affected the development will be compared. Table 6.1 makes a comparison of SIP and H.323 in terms of the factors that affected the development of the respective SMS Services.

Criteria	SIP	H.323
Protocol complexity	Simpler	Complex
Extensibility	Very extensible	Extensible
Call setup messages	Few messages	Many Messages
Data Call	Text mode	T.120
Media management	SDP	H.245
Message encoding	Simple – text based	Complex – ASN.1 encoding
Development time	Short	Longer
Protocol mapping to MGCP	Adequate	Adequate

Table 6.1 Comparison of development of Sip and H.323 SMS services

Table 6.1 does not attempt to list all the differences between SIP and H.323, just those that affected the development of the SMS services. There are many factors that affected the development of the services. These factors are a result of the protocols themselves. Since comparisons have been made between the features of SIP and H.323, this comparison looks at the protocols from a service creation point of view.

To begin with, H.323 is a well-specified and backward compatible suite of protocols. However, the need to maintain this compatibility results in it being large and complex. As a result, the development of even a simple service like this one, requires knowledge of several protocols within the H.323 stack.

SIP on the other hand is a much simpler protocol. It does not maintain backward compatibility and is specified in a much smaller document. It has been developed more specifically for the VoIP environment and therefore it does not require much to get going. SIP has a small message set, but is very extensible in terms of features. The approach that SIP uses is to ignore unknown headers and values. If a client requires a specific feature, it puts this feature under the require header. If the server does not support this feature it can return an error code with a list of features it does not understand. The client can then determine which feature is not supported by the error code and attempt a simpler operation. It is worth noting that the error codes are organised in a hierarchical structure in which only the class of the code is required to understand the error. For example, a 4XX error is a Client Error: The request contains bad syntax or cannot be fulfilled by the server and the request is rejected. The additional digits in the error code can be used to describe additional features while achieving compatibility with the semantics [Ohrtman, 2003]. SIP also phases out older or unused features resulting in a smaller, more concise stack.

H.323's need to maintain backward compatibility affects its extensibility too and makes it less extensible than SIP. As new features come and go, the size of the H.323 stack continually grows. In the H.323 SMS Service, the Fast-connect procedure was used. However, this procedure is an optional H.323 V2 feature and therefore not all H.323 terminals support it. This means that services developed for all H.323 endpoints need to support the standard lengthy call setup.

Next, let us consider what is required to setup a call. SIP requires the exchange of a minimum of three messages to setup a call. H.323 can also setup a call using three messages, if it is using the Fast-connect procedure. However if it is using standard H.323 signalling the number of messages increases significantly.

In the development of the SMS Service, the type of media exchanged was of type data. SIP's approach to this, is to establish a data session. H.323 on the other hand requires that a call be established first, before a data session can be started using the T.120 standard. A data session using the T.120 standard is appropriate when the session involves many exchanges, like in a chat session; however it was not necessary for this service. Thus, SIP's simpler approach to data communications aids the development of data services.

Let us now consider how the two protocols manage media. SIP does not define a specific protocol for media management, however in practice it uses SDP exclusively [Singh and Shultzrinne, 2000]. H.323 on the other hand specifies the H.245 protocol for media management. H.245 uses several messages to negotiate media capabilities and set up media streams. To perform the same task, SDP uses fewer messages, but does not have all the functionality that H.245 logical channels have. Thus, for example, if an endpoint advertises that it can receive certain media types on certain ports, it has to listen immediately on those ports. When the called endpoint sends media, it may send media on only certain of the advertised ports leaving the remainder idle.

Another factor that affected the development of the services in the way messages are encoded by SIP and H.323. SIP uses text-based encoding while H.323 uses binary encoding Q.931 or ASN.1 PER. Therefore, for debugging purposes, SIP messages are easily debugged using a network-sniffing tool like Ethereal. The down side to text encoding is the larger message sizes. H.323's messages are not as easily debugged since they are in binary format. Therefore, additional ASN.1 and Q.931 decoders are necessary. On the other hand, binary encoding produces smaller messages than text encoding.

Of all the factors that affected the development of the SMS Services, the different approaches to media management produced the greatest difference. In the SIP SMS service for example, since the CA included a SIP stack, it appeared as another SIP UA to SIP endpoints. Therefore the CA could communicate with SIP endpoints using model SIP communication. The only challenge that arose was in translating SIP session information to MGCP session information and vice versa. Since both, SIP and MGCP use SDP to describe session information, this was not much of a challenge.

Similarly in the H.323 SMS Service, the CA's communication with H.323 terminals was seamless. The challenge to map session information however was not as easy. Since H.323 uses H.245 and MGCP uses SDP, session information between the two had to be translated.

Thus, considering all the factors listed above, we can conclude that with regard to development, it was easier to develop the SIP SMS Service than the H.323 one. Although this is true for this service because it involved the use of MGCP, this remark does not necessarily hold true for the development of all VoIP services. Consequently, the question that arises is what else can be deduced from this service?

Firstly, the use of an MGCP gateway, instead of a protocol specific gateway, resulted in a major saving in development time. By developing a generic gateway that could be used by both SIP and H.323 meant that effort did not have to be duplicated as would be the case in creating protocol specific gateways. From this we can deduce that in a heterogeneous VoIP network, the use of MGCP can facilitate services that were otherwise not easily implemented. These services can be developed using the least amount of effort for one protocol and easily extended for any other protocols. Although the services were developed using data as the media type, they could be extended to include voice or video.

Let us now consider the impact of adding MGCP to the network. MGCP's core function is the control of gateways between heterogeneous networks. Bearing this in mind, a major benefit of adding MGCP to a network would be to allow it to take over the inter-network interfacing. In a network with SIP and H.323 an MGCP gateway could provide interworking with the PSTN for these endpoints, reducing the number of inter-networking gateways. Additionally, MGCP gateways scale better than monolithic gateways and are easier to manage and maintain.

The SIP-H.323 IWF

Lastly, based on the success of the SMS Services, we looked at the extension of MGCP to interwork between SIP and H.323. The CA was modified to include both the H.323 and SIP stacks. The CA allowed calls from SIP to H.323 and vice versa. It even allowed calls from an endpoint in the SIP network to a terminal in the H.323 network. However, experimentation was stopped at this point as sufficient insight into what MGCP adds in an IWF situation had been gained. The following conclusions were drawn.

An MGCP CA can perform as a basic IWF, but it does not offer anything significantly more than a signalling gateway would offer. In addition, modifying the MGCP CA further to meet more of the requirements of an IWF, results in an IWF that has even less to do with MGCP. Therefore, it is the author's opinion that using an MGCP CA to perform the SIP-H.323 IWF does not offer any significant advances as compared to using a signalling gateway, and would not recommend MGCP for this purpose.

6.2 Conclusion

There are a variety of VoIP networks in existence. They contain mostly SIP, H.323 or a combination of the two. Depending on the VoIP needs of the users these networks continually change and advance. Furthermore the need to communicate with telephones on the PSTN has required the use of gateways and gateway control protocols.

MGCP was designed specifically for the edge of the network to interface with the PSTN. As a result, it has a small message set that is developed for this precise use. This research set out to investigate and test the limits of MGCP’s call control within the network.

This research found that MGCP is valuable as a protocol that not only interfaces with the PSTN, but also facilitates the creation of services. The SMS services that were developed showed the use of MGCP in the development of a service that was not previously available to SIP or H.323. It showed that MGCP facilitated the creation of services and was particularly applicable to heterogeneous VoIP networks.

MGCP's proven ability at the edge of the network will ensure its continued existence. Our findings give substance to Orhtman's claims [Orhtman, 2003] that, as long as media gateways are interfacing with analogue or PSTN connections to IP networks, MGCP will be the controlling protocol. They also support Arango's *et al*'s prediction [Arango *et al*, 1999] that MGCP will continue to be an integral element in any softswitch architecture.

6.3 Future of GCPs

The GCP that was intended to supersede MGCP is Megaco/ H.248. Megaco currently co-exists with MGCP as hardware developed for both protocols continues to be manufactured. In terms of interworking with other networks, Megaco builds upon the success of MGCP, shares many of its features, and interworks with even more networks, including ATM. However, Megaco departs from the objectives of MGCP in that it is not seen purely as MGCP's successor, especially in terms of its relationship with other IP protocols. While MGCP was intended as a complimentary protocol to SIP, Megaco is seen as a competing protocol. Megaco's centralized VoIP architecture is perceived as an alternative to SIP's distributed architecture. Both architectures will co-exist for a while, according to [Gaynor, 2003], however by comparing the two architectures in terms service provisioning, it is likely that SIP will prevail, as it possesses more functionality for providing centralised as well as end-to-end services. SIP's architecture meets the needs of the big centralized service providers, the smaller service providers, and the individual users who want to experiment and innovate.

In terms of this research, our findings are applicable to Megaco, since the call control features of MGCP are closely replicated in Megaco. Megaco can be used to aid service creation for SIP and H.323 endpoints, but, like MGCP, is not recommended for the development of an interworking function between SIP and H.323. It is also interesting to note that although Megaco is the successor to MGCP, recent software developments in the open source community, like Asterisk [Asterisk, 2004] for example, cater for interworking with SIP, H.323 and MGCP, but not for Megaco. This could be an indication of the degree of deployment of MGCP compared to Megaco.

Future work in this area should include a comparison between MGCP and Megaco, and an analysis of each protocol's market adoption. Another area that is worth exploring with regard to MGCP's deployment is Next Generation Networks (NGN). Currently, there are many ideas on what the NGN will look like. Research to investigate the role that MGCP will play in the next generation network (NGN) would be an obvious extension to this research.

At present, the commercial use of GCPs appears to be primarily limited to their original purpose of interfacing the IP network with the PSTN or other analogue networks. However, with a range of factors influencing the future of these protocols, the final chapter on GCPs has not yet been written.

Appendix 1

XML Schema for the GSM gateway

```
<?xml version="1.0" encoding="iso-8859-1" ?>
  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:enc="http://www.w3.org/2001/12/soap-encoding">
    <xsd:import namespace="http://www.w3.org/2001/12/soap-
       encoding" />
    <xsd:annotation>
       <xsd:documentation xml:lang="en">This schema is intended
          to allow applications to interface with the GSM modem,
          located in the Department of Computer Science at
          Rhodes University, in order to send SMS
          messages.</xsd:documentation>
    </xsd:annotation>
    <xsd:element name="sms" type="smsenvelope" />
    <xsd:complexType name="smsenvelope">
       <xsd:choice>
          <xsd:choice minOccurs="1" maxOccurs="unbounded">
             <xsd:annotation>
                <xsd:documentation xml:lang="en">Client
                   Request</xsd:documentation>
             </xsd:annotation>
             <xsd:element name="sendsms"
                type="sendsmsenvelope" />
             <xsd:element name="readsms" type="recvlist" />
          </xsd:choice>
          <xsd:sequence>
             <xsd:annotation>
                <xsd:documentation xml:lang="en">Service
                   Response</xsd:documentation>
             </xsd:annotation>
             <xsd:choice minOccurs="1"
                maxOccurs="unbounded">
                <xsd:element name="sentsms"
                   type="sentsmsenvelope" />
                <xsd:element name="recvsms"
                   type="recvsmsenvelope" />
             </xsd:choice>
             <xsd:element name="newsms" type="idlist"
                minOccurs="0" maxOccurs="unbounded" />
          </xsd:sequence>
       </xsd:choice>
    </xsd:complexType>
    <xsd:complexType name="sendsmsenvelope">
       <xsd:annotation>
          <xsd:documentation xml:lang="en">container for a
             client request for one or more SMS messages to be
             sent.</xsd:documentation>
       </xsd:annotation>
       <xsd:all>
```

```
        <xsd:element name="phone" type="phonetype"
           minOccurs="1" maxOccurs="unbounded" />
        <xsd:element name="message" type="inmessagetype"
           minOccurs="1" maxOccurs="unbounded" />
     </xsd:all>
  </xsd:complexType>
 <xsd:complexType name="sentsmsenvelope">
     <xsd:annotation>
        <xsd:documentation xml:lang="en">container for
           service response after attempting to send SMS for
           client.</xsd:documentation>
     </xsd:annotation>
    <xsd:sequence>
        <xsd:element name="phone" type="phonetype" />
        <xsd:element name="message" type="outmessagetype" />
        <xsd:element name="status" type="statustype" />
        <xsd:element name="id" type="idtype" />
     </xsd:sequence>
  </xsd:complexType>
 <xsd:complexType name="recvsmsenvelope">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">container for
           service response to client request to read an
           SMS</xsd:documentation>
     </xsd:annotation>
    <xsd:sequence>
        <xsd:element name="phone" type="phonetype" />
        <xsd:element name="message"
           type="outmessagetype" >
        <xsd:element name="status" type="statustype" />
        <xsd:element name="id" type="idtype" />
        <xsd:element name="date" type="dateTime" />
     </xsd:sequence>
  </xsd:complexType>
 <!--
restrictions on field contents
 -->
<xsd:simpleType name="inmessagetype">
   <xsd:restriction base="xsd:string">
      <!--
      space for tag on outgoing SMS
       -->
        <xsd:length value="154" />
     </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="outmessagetype">
   <xsd:restriction base="xsd:string">
        <xsd:length value="160" />
     </xsd:restriction>
  </xsd:simpleType>
 <xsd:simpleType name="statustype">
   <xsd:restriction base="xsd:string">
        <xsd:enumeration value="OK" />
        <xsd:enumeration value="ERROR" />
        <xsd:enumeration value="UNKNOWN" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="phonetype">
     <xsd:restriction base="xsd:string">
        <xsd:pattern value="\+[0-9]{0,40}" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="idtype">
     <xsd:restriction base="xsd:string">
        <xsd:pattern value="(S|R)\d{1,4}" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="idlist">
      <xsd:list itemType="idtype" />
    </xsd:simpleType>
    <xsd:simpleType name="recvlist">
      <xsd:list itemType="recvtype" />
    </xsd:simpleType>
   <xsd:simpleType name="recvtype">
     <xsd:restriction base="xsd:string">
        <xsd:pattern
          value="(ALL|UNREAD|BCAST|SENT|RECEIVED|((S|
          R)\d{1,4}))" />
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:schema>
```

Typical SOAP request to send SMS

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<soap:Envelope xmlns:soap="...">
  <soap:Body>
    <sms:sms xmlns:sms="...">
      <sms:sendsms>
        <sms:phone>0821234567</sms:phone>
        <sms:message>Hello World!</sms:message>
      </sms:sendsms>
    </sms:sms>
  </soap:Body>
</soap:Envelope>
```

This request shows a client asking for the message "Hello World!" to be sent to the cell phone whose number is 0821234567.The web service would process this request, and would send the client a SOAP response indicating the status of this message. A typical SOAP response (and in fact, the response to the previous example) is shown below.

Typical SOAP response

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<soap:Envelope xmlns:soap="...">
  <soap:Body>
    <sms:sms xmlns:sms="hellip;">
      <sms:sentsms>
        <sms:phone>0821234567</sms:phone>
        <sms:message>Hello World!</sms:message>
        <sms:status>OK</sms:status>
        <sms:id>S0015</sms:id>
      </sms:sentsms>
      <sms:newsms>R0006</sms:newsms>
    </sms:sms>
  </soap:Body>
</soap:Envelope>
```

The response contains the full information of the request, as well as various other fields containing information on the status of the message.

The most important of these is the status field. This contains information on whether or not the service was able to send the message. In the example above, the message “Hello World!” was successfully sent to cell number 0821234567 as indicated by the “OK” in the status field. If the message cannot be sent, this is indicated by an “ERROR” status.

Appendix 2

H.323-SIP Call using IWF

A call from H.323 terminal to SIP terminal using Fast-connect procedure based on [Agrawal *et al*, 2001]. The messages shown here correspond to those from Figure 5.4.

```
Message Details
---------------

1 Setup (fastStart=true,OLC) H323 -> IWF

H323-UserInformation
{
   h323-uu-pdu                                       :
     h323-message-body                               :
       setup                                         :
         protocolIdentifier                          : itu-t
                                                     : recommendation
                                                     : h
                                                     : 2250
                                                     : version
                                                     : 2
         h245Address                                 :
           ipAddress                                 :
             ip                                      : 164.164.28.101
             port                                    : 2000
         sourceAddress                               :
           e164 address                             : 01000220013101720
           h323-ID address                          : UserB@there.com
           e164 address                             : 7199557429
         sourceInfo                                  :
           vendor                                    :
             vendor                                  :
               t32CountryCode                        : 11
               t32Extension                          : 11
               manufacturerCode                      : 11
               productId                             : IWF
               versionId                             : SIP-H323
           terminal                                  :
           mc                                        : false
           undefinedNode                             : false
         destCallSignalAddress                       :
           ipAddress                                 :
             ip                                      : 164.164.28.121
             port                                    : 1720
         activeMC                                    : false
         conferenceID                                : Hex( 56 34 34 34
                                                       34 EF 0B 00 21
                                                       21 E4 A5 35 A3
                                                       9A 82)
         conferenceGoal                              : create
         callType                                    : pointToPoint
         sourceCallSignalAddress                     :
           ipAddress                                 :
             ip                                     : 164.164.28.101
             port                                   : 1700
         callIdentifier                             :
           guid                                     :      Hex( 56 34 34 34 34 EF
                                                            0A 00 21 21 E4 A5 35
                                                            A3 9A 82 )
         fastStart                                  :
           fastStart - Sequence[ 0 ]                :
           forwardLogicalChannelNumber              : 1
           forwardLogicalChannelParameters          :
             dataType                               :
               audioData                             :
                 g711Ulaw-64k                        : 60
```

```
  multiplexParameters                      :
  h2250LogicalChannelParameters            :
    sessionID                              : 1
    mediaControlChannel                    :
      unicastAddress                       :
        iPAddress                          :
          network                          : 164.164.28.101
          tsapIdentifier                   : 2327
                                           :
fastStart - Sequence[ 1 ]                  :
forwardLogicalChannelNumber                : 2
forwardLogicalChannelParameters            :
  dataType                                 :
    audioData                              :
      g7231                                :
        maxA1-sduAudioFrames               : 8
        silenceSuppression                 : false
  multiplexParameters                      :
  h2250LogicalChannelParameters            :
    sessionID                              : 1
    mediaControlChannel                    :
      unicastAddress                       :
        iPAddress                          :
          network                          : 164.164.28.101
          tsapIdentifier                   : 2327
                                           :
fastStart - Sequence[ 2 ]                  :
forwardLogicalChannelNumber                : 4762
forwardLogicalChannelParameters            :
  dataType                                 :
    nullData                               :
  multiplexParameters                      :
    none                                   :
reverseLogicalChannelParameters            :
  dataType                                 :
    audioData                              :
      g711Ulaw-64k                         : 60
  multiplexParameters                      :
  h2250LogicalChannelParameters            :
    sessionID                              : 1
    mediaChannel                           :
      unicastAddress                       :
        iPAddress                          :
          network                          : 164.164.28.101
          tsapIdentifier                   : 2326
    mediaControlChannel                    :
      unicastAddress                       :
        iPAddress                          :
          network                          : 164.164.28.101
          tsapIdentifier                   : 2327
                                           :
fastStart - Sequence[ 3 ]                  :
forwardLogicalChannelNumber                : 4762
forwardLogicalChannelParameters            :
  dataType                                 :
    nullData                               :
  multiplexParameters                      :
    none                                   :
reverseLogicalChannelParameters            :
  dataType                                 :
    audioData                              :
      g7231                                :
        maxA1-sduAudioFrames               : 8
        silenceSuppression                 : false
  multiplexParameters                      :
  h2250LogicalChannelParameters            :
    sessionID                              : 1
    mediaChannel                           :
      unicastAddress                       :
        iPAddress                          :
          network                          : 164.164.28.101
          tsapIdentifier                   : 2326
    mediaControlChannel                    :
      unicastAddress                       :
        iPAddress                          :
          network                          : 164.164.28.101
          tsapIdentifier                   : 2327
```

```
                                                  :
           fastStart - Sequence[ 4 ]              :
           forwardLogicalChannelNumber            : 3
           forwardLogicalChannelParameters        :
             dataType                             :
               videoData                          :
                 h261VideoCapability              :
                   qcifMPI                        : 1 [1/29.97 Hz]
                   cifMPI                         : 1 [1/29.97 Hz]
                   temporalSpatialTradeOffCapability : false
                   maxBitRate                     : 600 [100
                                                    bit/sec]
                   stillImageTransmission         : false
             multiplexParameters                  :
             h2250LogicalChannelParameters        :
               sessionID                          : 2
               mediaControlChannel                :
                 unicastAddress                   :
                   iPAddress                      :
                     network                      : 164.164.28.101
                     tsapIdentifier               : 2329
                                                  :
           fastStart - Sequence[ 5 ]              :
           forwardLogicalChannelNumber            : 4762
           forwardLogicalChannelParameters        :
             dataType                             :
               nullData                           :
             multiplexParameters                  :
               none                               :
           reverseLogicalChannelParameters        :
             dataType                             :
               videoData                          :
                 h261VideoCapability              :
                   qcifMPI                        : 1 [1/29.97 Hz]
                   cifMPI                         : 1 [1/29.97 Hz]
                   temporalSpatialTradeOffCapability : false
                   maxBitRate                     : 600 [100
                                                    bit/sec]
                   stillImageTransmission         : false
             multiplexParameters                  :
             h2250LogicalChannelParameters        :
               sessionID                          : 2
               mediaChannel                       :
                 unicastAddress                   :
                   iPAddress                      :
                     network                      : 164.164.28.101
                     tsapIdentifier               : 2328
               mediaControlChannel                :
                 unicastAddress                   :
                   iPAddress                      :
                     network                      : 164.164.28.101
                     tsapIdentifier               : 2329
                                                  :
         mediaWaitForConnect                      : false
         canOverlapSend                           : false
     h245Tunneling                                : false
}
```

2. INVITE IWF -> SIP

```
INVITE sip:UserB@there.com SIP/2.0
Via: SIP/2.0/UDP 164.164.28.121:5060
From: <sip:UserA19284@164.164.28.121>
To: <sip:UserB@there.com>;tag=9876
Call-ID: 4423493498581@164.164.28.121
CSeq: 1024 INVITE
Contact: <sip:UserA19284@164.164.28.121>
Content-Type: application/sdp
Content-Length: ...

v=0
o=UserA 2890844526 2890844526 IN IP4 164.164.28.101
s=-
c=IN IP4 164.164.28.101
t=3034423619 0
m=audio 2326 RTP/AVP 0
```

```
a=rtpmap:0 PCMU/8000
```

3. 180 RINGING SIP -> IWF

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 164.164.28.121:5060
From: <sip:UserA19284@164.164.28.121>
To: <sip:UserB@there.com>
Call-ID: 4423493498581@164.164.28.121
CSeq: 1024 INVITE
Content-Length: 0
```

4. Alerting IWF -> H323

```
H323-UserInformation
{
   h323-uu-pdu                                              :
     h323-message-body                                      :
       alerting                                             :
         protocolIdentifier                                 : itu-t
                                                            :
recommendation
                                                            : h
                                                            : 2250
                                                            : version
                                                            : 2
         destinationInfo                                    : gateway
         callIdentifier                                     : Hex( 56 34 34 34 34
                                                                   EF 0A 00 21 21 E4
                                                                   A5 35 A3 9A 82 )
}
```

5 200 OK SIP -> IWF

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 164.164.28.121:5060
From: <sip:UserA19284@164.164.28.121>
To: <sip:UserB@there.com>;tag=9876
Call-ID: 4423493498581@164.164.28.121
CSeq: 1024 INVITE
Contact: <sip:UserB@there.com>
Content-Type: application/sdp
Content-Length: ...

v=0
o=UserB 2890844526 2890844526 IN IP4 164.164.28.141
s=-
c=IN IP4 164.164.28.141
t=3034423619 0
m=audio 4346 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

6 Connect (fastStart=true,OLC) IWF -> H323

```
H323-UserInformation
{
   h323-uu-pdu                                      :
     h323-message-body                              :
       connect                                      :
         protocolIdentifier                         : itu-t
                                                    : recommendation
                                                    : h
                                                    : 2250
                                                    : version
                                                    : 2
         destinationInfo                            :
           vendor                                   :
             vendor                                 :
               t32CountryCode                       : 11
               t32Extension                         : 11
```

```
              manufacturerCode                    : 11
              productId                           : IWF
              versionId                           : SIP-H323
          gateway                                 :
          mc                                      : false
          undefinedNode                           : false
        conferenceID                              : Hex( 56 34 34 34 34
                                                         EF 0B 00 21 21 E4 A5
                                                         35 A3 9A 82 )

        callIdentifier                            :
          guid                                    : Hex( 56 34 34 34 34
                                                         EF 0A 00 21 21 E4 A5
                                                         35 A3 9A 82 )

        fastStart                                 :
          fastStart - Sequence[ 0 ]               :
          forwardLogicalChannelNumber             : 1
          forwardLogicalChannelParameters         :
            dataType                              :
              audioData                           :
                g711Ulaw-64k                      : 60
            multiplexParameters                   :
            h2250LogicalChannelParameters         :
              sessionID                           : 1
              mediaChannel                        :
                unicastAddress                    :
                  iPAddress                       :
                    network                       : 164.164.28.141
                    tsapIdentifier                : 4326
              mediaControlChannel                 :
                unicastAddress                    :
                  iPAddress                       :
                    network                       : 164.164.28.141
                    tsapIdentifier                : 4327
                                                  :
          fastStart - Sequence[ 1 ]               :
          forwardLogicalChannelNumber             : 1
          forwardLogicalChannelParameters         :
            dataType                              :
              nullData                            :
            multiplexParameters                   :
              none                                :
          reverseLogicalChannelParameters         :
            dataType                              :
              audioData                           :
                g711Ulaw-64k                      : 60
            multiplexParameters                   :
            h2250LogicalChannelParameters         :
              sessionID                           : 1
              mediaControlChannel                 :
                unicastAddress                    :
                  iPAddress                       :
                    network                       : 164.164.28.141
                    tsapIdentifier                : 4327
                                                  :

                                                  :
     h245Tunneling                                : false
}
```

7 Ack IWF -> SIP

```
ACK sip:UserB@there.com SIP/2.0
Via: SIP/2.0/UDP 164.164.28.121:5060
From: <sip:UserA19284@164.164.28.121>
To: <sip:UserB@there.com>;tag=9876
Call-ID: 4423493498581@164.164.28.121
CSeq: 1024 ACK
Content-Length: 0
```

Appendix 3

Acronyms and Abbreviations

ASN.1 – Abstract Syntax Notation 1

CA – Call Agent alias MGC

DSP – Digital Signal Processor

GCP – Gateway Control Protocol

GK – Gatekeeper

GSM – Global System for Mobile communications

H.323 – ITU-T standard for peer-to-peer call control systems

IETF – Internet Engineering Task Force

ISUP – Integrated Services Digital Network User Part (SS7)

ITU – International Telecommunication Union

IWF – Interworking Function

MCU – Multipoint Control Unit

Megaco/H.248 protocol –The single gateway control open standard jointly developed by both IETF and ITU-T international standards bodies (IETF Megaco Protocol and ITU-T H.248),

MCU – Multipoint Control Unit

MGC – Media Gateway Controller

MG – Media Gateway alias Call Agent

MGCP – Media Gateway Control Protcol

PC – Personal Computer

PSTN – Public Switched Telephone Network

RFC – Request For Comment (the format used for IETF approved standards and informational documents)

RUCSD – Rhodes University Computer Science Department

SDP – Session Description Protocol

SIP – Session Initiation Protocol (IETF standard for peer-to-peer call and session control systems)

SOAP – Simple Object Access Protocol

SMS – Short Message Service

SS7 – Signalling System 7

VoIP – Voice over Internet Protocol

VOCAL – Vovida Open Communication Application Library

TDM – Time Division Multiplexing

XML – eXtensible Markup Language

References

Agrawal H, Roy R, Palawat V, Johnston A, Agboh C, Wang D, Schulzrinne H, Singh K, and Maeng J, 2001. SIP-H.323 Interworking, Internet Draft, Internet Engineering Task Force

Arango M, Dugan A, Elliott I, Huitema C, and Pickett S, 1999. Media Gateway Control Protocol (MGCP) Version 1.0, IETF RFC 2705

Asterisk, 2004. Open Source Soft PBX website, http://www.asterisk.org

Bayer M, 2001. Computer Telephony Demystified, McGraw-Hill

Black U, 2001. Internet Telephony – Call Processing Protocols, Prentice Hall PTR

Box D, Ehnebuske D, Kakivaya G, Layman A, Mendelsohn N, Nielsen H F, Thatte S, Winer D, 2000. Simple Object Access Protocol (SOAP) 1.1 World Wide Web Consortium. http://www.w3.org/TR/SOAP

Collins D, 2001. Carrier Grade Voice over IP, McGraw-Hill

Cisco, 2002. Understanding Packet Voice Protocols, The International Engineering Consortium Web Forum Tutorials, http://www.iec.org

Dang L, Jennings C and Kelly D, 2002. Practical VoIP Using VOCAL, O' Reilly

Douskalis B, 2000. IP Telephony-The Integration of Robust VoIP Services, Prentice Hall PTR

Gaynor M, 2003. Linking Market Uncertainty to VoIP Service Architectures, IEEE Internet Computing, July 2003, Pages16-22, http://computer.org/internet/

Greene N, Ramalho M and Rosen B, 2000. Media Gateway Control Protocol Architecture and Requirements, IETF RFC2805

Halse G, Wells G, 2002. A bi-directional SOAP / SMS gateway service, SATNAC Conference Proceedings, Drakensberg

Handley M, Jacobson V, 1998. SDP: Session Description Protocol, IETF RFC2327

Handley M, Schulzrinne H, Schooler E and Rosenberg J, 1999. SIP: Session Initiation Protocol, IETF RFC2543

Hersent O, Gurle D and Petit J, 2000. IP Telephony – Packet based multimedia communication systems, Addison Wesley

Hsieh M, Okuthe J, Terzoli A and Wentworth P, 2002. An investigation into multimedia service creation using SIP, SATNAC Conference Proceedings, Drakensberg

Hsieh M, 2004. Service provisioning in two open source SIP implementations, CINEMA and VOCAL, MSc Thesis, Rhodes University, Grahamstown

ITU-T, 1997. ITU-T Recommendation Q.762: Signalling System No. 7 - User part and general functions of messages and signals, Telecommunications Standardization sector, International Telecommunication Union

ITU-T, 1998a. ITU-T Recommendation H.323: Packet- based multimedia communications systems, Telecommunications Standardization sector, International Telecommunication Union

ITU-T, 1998b. ITU-T Recommendation H.245: Control protocol for multimedia communications, Telecommunications Standardization sector, International Telecommunication Union

ITU-T, 1998c. ITU-T Recommendation H.225.0: Call signalling protocols and media stream packetization for packet-based multimedia communication systems, Telecommunications Standardization sector, International Telecommunication Union

ITU-T, 2000. ITU-T Recommendation H.248: Gateway control protocol, Telecommunications Standardization sector, International Telecommunication Union

Jacobs A, Clayton P, 2002. Utilizing MGCP to design an H.323 Endpoint SMS Service, SATNAC Conference Proceedings, Drakensberg

Jacobs A, Clayton P, 2003. Investigating Call Control using MGCP, SATNAC Conference Proceedings, George

Levin D, 2001. IP Call Control: The standards that matter, http://www.radvision.com/articles/121801_callcontrol.html

Manesh N, 2004. Asterisk: A non-technical Overview, http://www.millenigence.com /articles/asterisk-non-technical-review.pdf

Ohrtman F, 2003. Softswitch Architecture for VoIP, McGraw Hill

OpenH323, 2003. The OpenH323 Project coordinated by Equivalence Pty Ltd, http://www.openh323.org

Penton J, Terzoli A and Wentworth P, 2001. Deploying a feature-rich H.323 environment in which to practice the creation of services, SATNAC Conference Proceedings, Wild Coast Sun

Russel T, 2000. Signalling System #7, McGraw-Hill

RADVision, 2002. Implementing Media Gateway Control Protocols- A RADVision White Paper

Rosen B, 2001. VoIP gateways and the Megaco architecture, BT Technology Journal, Volume 19 No. 2, Pages 66-76

Schulzrinne H, Agboh C, 2004. Session Initiation Protocol (SIP) - H.323 Interworking Requirements, http://www.ietf.org/internet-drafts/ draft-agrawal-sip-h323-interworking-reqs-07.txt

Singh K, Shultzrinne H, 2000. Interworking Between SIP/SDP and H.323, IPTEL 2000 Conference proceedings, Berlin

Varshney U, Snow A, McGivern A and Howard C, 2002. Voice over IP, Communications of the ACM, Volume 45 (1), Pages 89-96

Wang L, 2002. Modelling and verification of interworking between SIP and H.323, MSc thesis, Concordia University, Quebec