IT 08 009

Examensarbete 30 hp

May 2008

# Voice mail system for IP Multimedia Subsystem

Henrik Back

Ming Zhao

Institutionen för informationsteknologi

*Department of Information Technology*

UPPSALA
UNIVERSITET

**Teknisk- naturvetenskaplig fakultet**
**UTH-enheten**

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

## Abstract

# Voice mail system for IP Multimedia Subsystem

*Henrik Back and Ming Zhao*

The IP Multimedia Subsystem (IMS) framework as an architectural framework to deliver multimedia services is under rapid development, to become the next generation telecommunication service framework.

The objective of this work is to design an application server for the IMS framework to provide voice mail service to subscribed users. This master thesis contributes to this objective by the development of a prototype of a voicemail application server that integrates into the IMS architecture. Several communication protocols, among them SIP and MEGACO, were implemented for the integration of the application server with the IMS core network. All implementation was done using Erlang, a language widely used in telecom. After the prototype of the voice mail server was built, it was tested in an IMS simulation environment. The tests confirmed that the basic functionality of the voicemail was successfully implemented and that the integration with the IMS network was satisfactory. In future work the issue of QoS is one that needs to be addressed.

Handledare: Martin Kjellin
Ämnesgranskare: Mats Daniels
Examinator: Anders Jansson
IT 08 009
Sponsor: Mobile Arts AB

Tryckt av: Reprocentralen ITC

# 1 Introduction

## 1.1 Motivation

It has long been standard for telephone operators to use Voice Mail in GSM systems, it provides a good service for customers, and earns more money for the operators, as it allows more calls to be completed. As the new IMS system is coming into the market, the authors of this thesis were asked by a company called Mobile Arts to find out what would be needed to build a Voice mail system for IMS, and to implement a prototype of such a system. Some of the questions that were posed were:

What protocols does the VMS have to support in order to work in an IMS environment?

Can the signaling and the media be separated in different logical entities, and more specifically, can the media handling part of the application be made more general so that it might be reused with other applications that require streaming media to be sent?

What possibilities, if any, are there in IMS that cannot be found in ordinary GSM?

What problems, if any, are there in IMS that cannot be found in ordinary GSM?

How can you make such a system scalable?

## 1.2 Aims and Objectives

The objective of the work is to gain understanding enough to find answers to the questions posed above, and to be able to draw some conclusions about do’s and don’ts when implementing applications for IMS in general, and a Voice mail System in particular. The aim will also be to implement a prototype of a VMS for the IMS system.

The goal of the implementation is to be able to:
-deposit a message
-retrieve a previously deposited message
-notify a user that he has a new message when he is coming online

## *1.3 Methodology*

When designing and implementing the system we will adhere to standard software engineering practices. This means that the work will be divided into several phases: Research, Design, implementation and validation. In the research phase, a lot of literature is studied in order to be able to understand the problem and its context. Some of the results of the research phase are presented in Chapter 2: Background. The Design phase consists of using the knowledge obtained in the research phase to build a detailed requirement specification, and to model a system that meets the requirements. It is described in detail in Chapter 3: Design. Likewise the implementation phase consists of implementing the system modeled in the Design phase. This phase is described in Chapter 4: Implementation. And finally the Validation phase consists of creating a test specification that covers all requirements set on the system in the design phase, and to test the system according to this. A more detailed description of the method, as well as the results, can be found in Chapter 5: Validation.

## *1.4 Related works*

Voice Mail Systems are common, and many such implementations and adaptations exist, for instance the company where this work was performed has one such implementation. The interesting part of that system for this work is the general architecture and mechanisms for handling large volumes of traffic rather than any implementation of specific protocols as we will be using others.
Johan Person and Roger Jacobsen have implemented a messaging system for IMS [7]. The work centers on the SIP protocol and as such implements some features that are of interest for our work, specifically notification.

## *1.5 Outline of the thesis*

**Chapter 1: Introduction**
It contains the motivations and the objectives of this thesis. Also contains a description of related works and this outline of the thesis.

**Chapter 2: Background**

It contains a detailed background description of the subject, including the IMS and Voice Mail Systems.

**Chapter 3: Design**
It describes the Designing of the Voice Mail application, both the methodology used and the final architecture of the system.

**Chapter 4: Implementation**
It describes the Implementation of the application, the methodology as well as the problems encountered.

**Chapter 5: Validation**
It describes the validation of the application: the method used and the results

**Chapter 6: Conclusion**
It contains our conclusions of the work, and thoughts about the current work and for the future.

# 2 Background

In this chapter a brief introduction to the IP Multimedia Subsystem (IMS) is given, as is an internationally standardized network architecture describing how telecom operators can provide multimedia services to users. The goal of the thesis is to produce voice mail service within IMS scope.

## *2.1 IP Multimedia Subsystem (IMS)*

IP Multimedia Subsystem is an architectural framework to deliver multimedia service to terminals over wireless network like GPRS or 3G. The mail goal of IMS is to standardize methods for deploying common service delivery architecture, and this architecture supports any service and media regardless of how the customer to be connected to the network.

Third Generation (3G) networks aim to merge two of the most successful paradigm in communications: cellular networks and the Internet. The IP multimedia subsystem is the key element in the 3G architecture that makes it possible to provide ubiquitous cellular access to all the services that the Internet provides. The future vision of IMS is to evolve mobile network beyond GSM. So IMS aims to combine the latest trends in technology, to make the mobile internet paradigm come true, to create a common platform to develop diverse multimedia services, and to create a mechanism to boost margins due to extra usage of mobile packet-switched network.

IMS network consists of 3 layers (See figure 1-1 from top down): Service/application layer, IMS control layer and transport Layer.

Transport Layer handles the incoming connections from all different kinds of the terminals (Mobile phone, PDA, computer) through different methods. The only requirement is that the client is running as a SIP User Agent [9] under IPv4/IPv6.

Once the device is connected to the IMS network, all the incoming data is processed in the IMS layer. IMS layer isolates different access networks from service layer. The advantage is for all service to have a common control layer. In this way, services do not have their own control functions. The IMS layer mainly consists of SIP servers and proxies called Call Session Control Function (CSCF).

The service and application layer contains various Application Server (AS) which hosts and executes the service. As mentioned earlier, the AS interfaces the CSCF using SIP. Since the IMS layer has all the control functions, the complexity of any AS is greatly reduced, thus the cost for providing and maintaining the AS is relatively low.

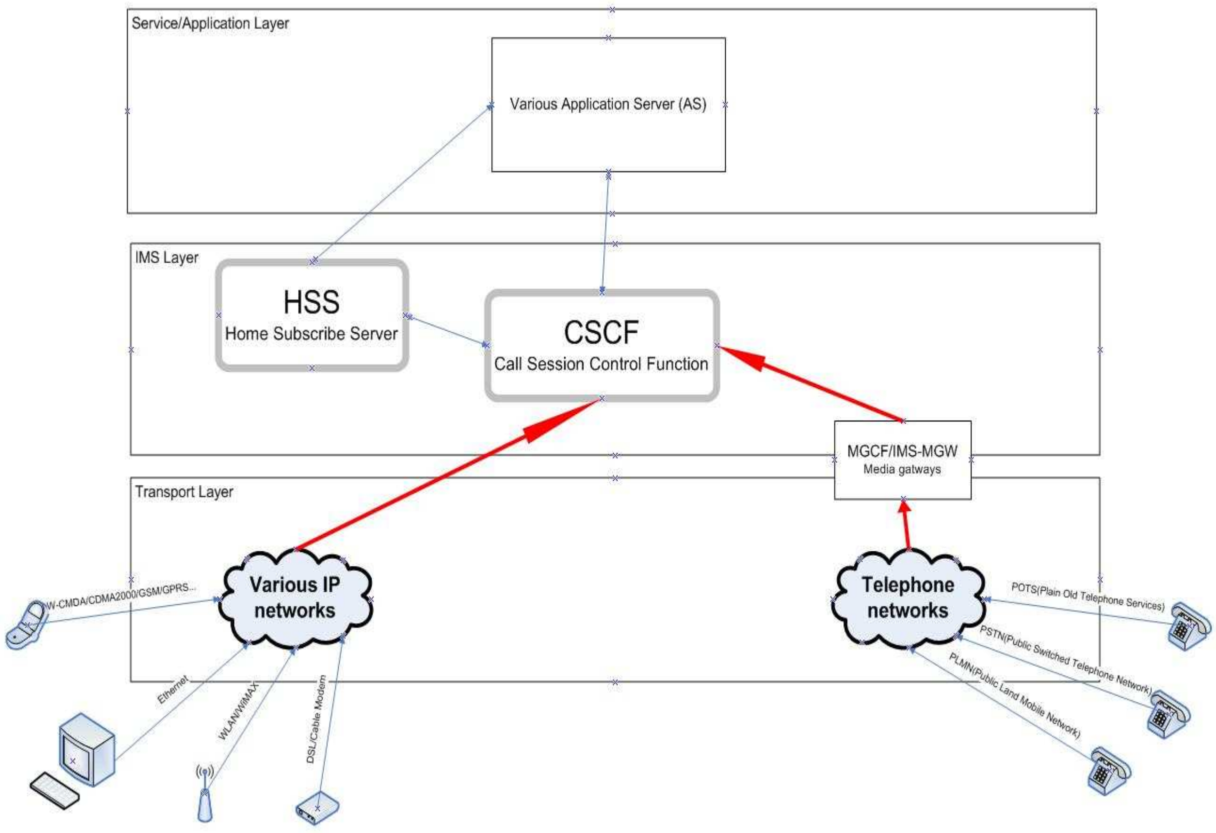

Service/Application Layer
Various Application Server (AS)
IMS Layer
HSS
Home Subscribe Server
CSCF
Call Session Control Function
MGCF/IMS-MGW
Media gatways
Transport Layer
Various IP networks
Telephone networks
W-CMDA/CDMA2000/GSM/GPRS...
Ethernet
WLAN/WIMAX
DSL/Cable Modem
POTS(Plain Old Telephone Services)
PSTN(Public Switched Telephone Network)
PLMN(Public Land Mobile Network)

Figure 1-1 IMS network layers

IP- and IMS-based solutions offer strong advantages in cost and functionality over traditional solutions.

In order to provide the same quality of multimedia service onto the IP network, one of the main issues about the packet-switched network is the lack of the QoS with a best-effort service. That means there are no guarantees for the real-time multimedia service to be successful without considering the bandwidth and packet delay like Internet.

Various protocols are being developed as solutions in IMS. Session Initiation Protocol (SIP) [9] plays the key role in IMS. The SIP is used to establish and manage the call session and ensure the QoS for the media transportation. For Authentication, Authorization, Accounting (AAA), the IMS uses the Diameter protocol. [11]. In addition to SIP and Diameter, H.248 [12] is used to signal the media node (Example: a media gateway controller controlling the media gateway). RTP (Real-Time Transport Protocol, defined in RFC 3550) and RTCP (RTP Control Protocol) are used to transport the real time media and provide the statistics to ensure the QoS.

As a whole, IMS is another packet switched network architecture that specializes on serving telecom services. The reasons to create IMS are listed below:

1. To provide Quality of Service with a best-effort service
2. To be able to charge multimedia sessions appropriately
3. To provide cost effectiveness service.
4. To provide integrated services (IMS defines the standard interfaces to be used by the service developers)

## 2.2 Voice Mail Service (VMS)

Voice mail service is a familiar and ubiquitous service that allows a call to be diverted to an automated system that records an audio message. It is intended to manage the telephone voice messages for a group of people who have registered the service. It uses the handset like mobile phone or telephone for the user interface, uses a centralized computerized server system to manage the voice mail box.

Recent enhancements to voicemail allow direct voice messaging, wherein the caller can leave a message for the subscriber without first trying to call their cell phone. In some offerings, the system will send an SMS to the subscriber notifying them of the waiting message and giving them a link to click on to retrieve the message. This direct voice messaging feature, which is conceptually similar to email or SMS, is becoming very popular and goes by a variety of names, such as audio SMS , voice SMS, SMS voice messaging, and instant voice messaging.

A voice mail service is a familiar service that should include but not limit to these functionalities.

- To deposit the voice message
- To retrieve the voice message
- To delete the voice message
- To notify the user of the newly stored message

The voice mail service enables the customer to have their incoming calling stored in a personalized mailbox associated with the user's identity (Telephone number, SIP URI), when the user appears to be offline or busy.

When the user change their state (being on-line or not busy), the voice mail system is able to send the notification to the terminal handset about the newly stored message. The updates will be forwarded to the user through a message.

The user can choose to retrieve the voice message stored in the mailbox with identification code (password). The voice mail is played with certain codec to the user. Meanwhile, the user can decide to delete the old messages that have been read. The interactions between the user and the mail system are conducted by playing announcements and using touch-tones.

Voicemail messages are stored on the hard disk drivers. Many users can retrieve or store messages at the same time on the same voicemail system. The concurrency and scalability requires certain reliable distributed system, thus Erlang OTP has been chosen to be the development foundation.

## 2.3 Voice Mail Service in IMS scope

The goal of this thesis is to implement voice mail service within the IMS scope. According to the IMS standard, all the service developed by the third parities could be adopted and provided to the users. Voice mail service is offered by integrating a voice server into IMS network.

Figure 1-2 depicts the components within the IMS network. On the left of the picture, there are the IMS terminals. As it shows in the picture, IMS terminal contains not only the mobile terminals that attach to a packet network though a radio link, such as mobile phone through GPRS, but also other devices that can be connected to IMS through alternative access, such as computer through WLAN.

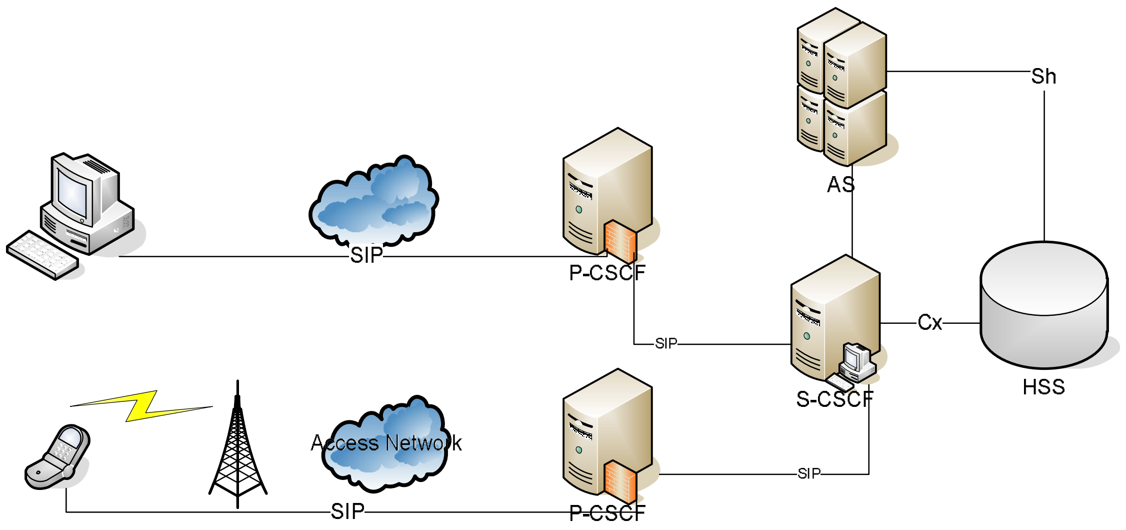


Figure 1-2 IMS architecture overview

Four central components in the IMS architecture are displayed. They are P-CSCF, S-CSCF, AS and HSS. The CSCF (Call/Session Control Function), which is a SIP server, is a critical node in IMS. P-CSCF performs here as a proxy server and it also authenticates and authorizes the user to IMS. S-CSCF is an essential SIP server, which performs the session control and it has implemented the Cx interface [18] to talk to the HSS database. The AS (Application Server) is a SIP entity that hosts and executes services. The AS interfaces the S-CSCF using SIP. The AS offers different kinds of services to the subscribers. The Home Subscriber Server (HSS) acts as a subscriber database, which contains all the user-related information.

Voice server acts as an application server as we described in figure 1-2 to provide the service to the customers. In order to merge Voice server to IMS network, the investigation has been made to the interfaces among the components in the IMS. The communication protocols between the entities of the IMS and voice mail server are SIP and Diameters etc. Voice server should fulfill the protocol stack to ensure the communication.

These protocols will be detailed in the later chapter on the implementation stage of the voice mail service.

# 3 Design

## *3.0 Requirements*

Before any modeling of the system can begin, it is important to have a proper requirement specification. In order to build such a document, it is necessary to have a good understanding of the problem, not only what the application should be able to do, but also what it must not do. As described above, the application will be interacting in a larger system, so it is also important to find out what requirements are laid on the application by the surrounding entities in order for it to be able to fit in to the larger system.
Of the latter, the required protocols to implement, whether fully or in part, are of special interest. Some protocols tend to set a lot of requirements on their users, but many of the smaller requirements are not that interesting in the design phase, they are more of an implementation problem, although they too need to be noted in the specification of course. One of the most important things of a protocol is whether the protocol is connection oriented (uses TCP or similar) or not (uses UDP or similar) as this can have a relatively large impact on how the system is designed. In this case, the signaling protocols were found to be connection oriented, whereas the media handling protocols were connectionless, and the H.248 protocol could be run as either.

## *3.1 The softswitch architecture*

One of the requirements is that if possible the media handling part of the application should be made generic, so that it might be reused in other applications with minimal changes. This is accomplished by using the so called softswitch architecture.

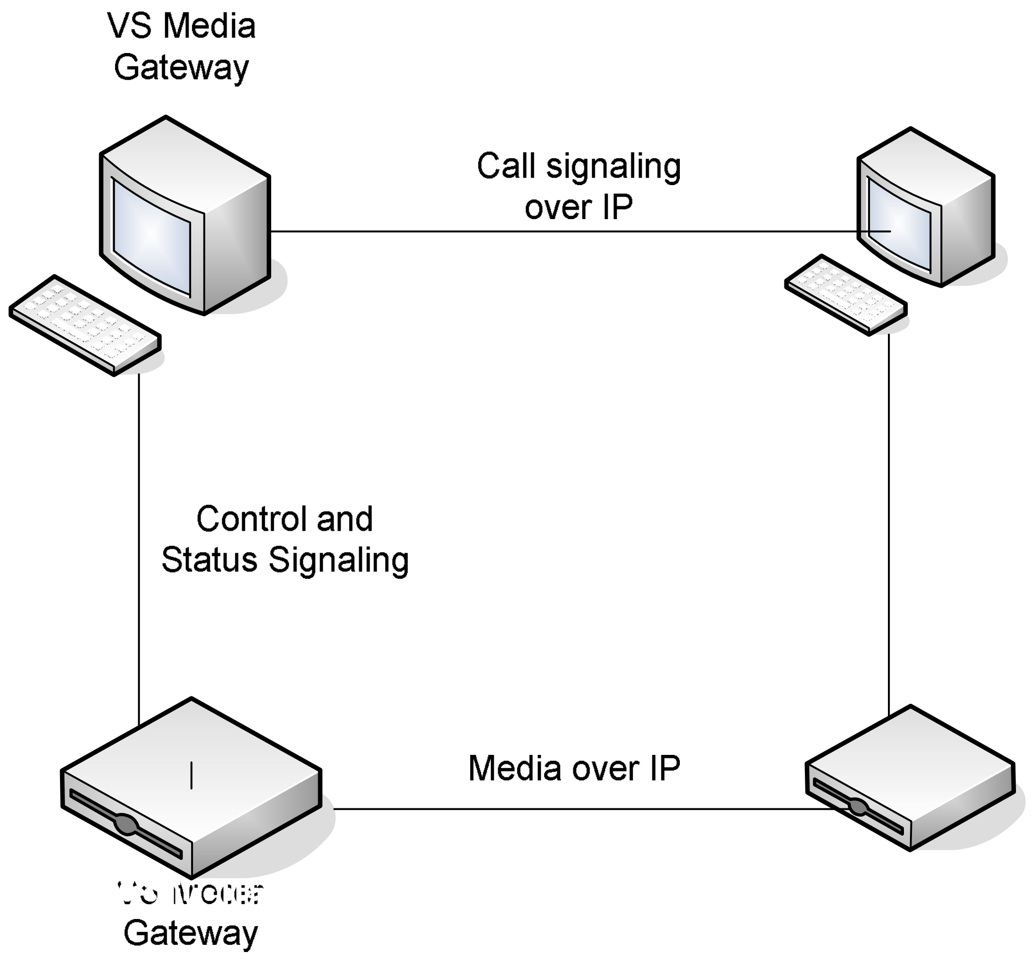


*Fig 3.1*

The softswitch architecture involves the separation of the media path and media-conversion functions from the call control and signaling functions. The voice server is deployed in a distributed way like the softswitch architecture in order to separate the functionalities as depicted in Figure 3-1.The entity that handles the call control is known as media gateway controller, MGC, while the entity that performs the media conversation is know as media gateway, MG.   This architecture also provides some scalability in that it is possible for the MGC to control more than one MG, thereby increasing the amount of media streams that can be handled.

## 3.2 Erlang and the Erlang design principles

When considering the requirement for concurrency, we found out that many telecom applications use the programming language Erlang/OTP (Open Telecom Platform), and something called the Erlang design principles Erlang [11] is a concurrent functional programming language which has many features like: concurrency, distribution, robustness. The main strength of Erlang is its ability of concurrency. It has small but powerful set of primitives to create processes and communicate between them. Erlang needs a run-time environment since it is by default an interpreted language. Open Telecom Platform is a library developed by Ericsson AB that defines a large portion of Erlang's behavior, since much of Erlang development is based on this library. This voice mail system is developed under the Open Telecom Platform.

The Erlang design principles are built on the concept of workers and supervisors. A worker is a process that does most of the actual computing, and a supervisor is a process that monitors the behavior of one or more workers. If a worker crashes or is malfunctioning, it can be restarted by its supervisor. The processes are arranged in a hierarchical three, called a supervision three (fig. 3-2)

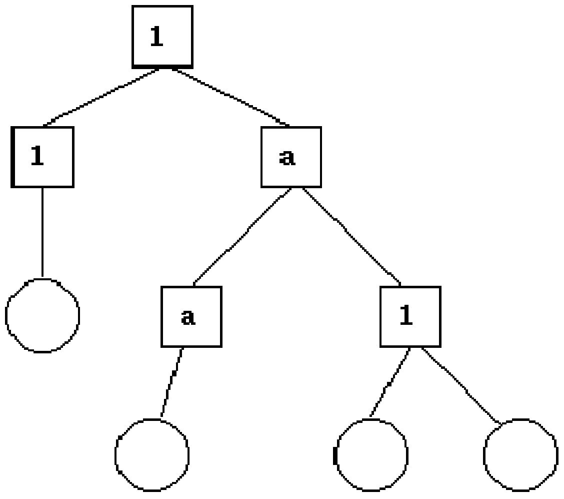


Fig 3.2, Supervision Tree

This concept provides good scalability and robustness to the application.

## 3.3 Service logic (application architecture)

As stated earlier, it was decided to divide the application into a Media Gateway, MG, and a Media Gateway Controller, MGC. The MGC contains all the service logic of the Voice Mail System. The main functions of the MGC are call setup and control, and notification when a message has been deposited. Since the notification and the call management is two distinctly separate tasks, it was decided to also divide the MGC into two parts: The call logic module, and the notification module. When a supervisor is added to these modules, and each module is broken down to a main process and a number of workers, the result is what you see in fig. 3.3 and fig 3.4

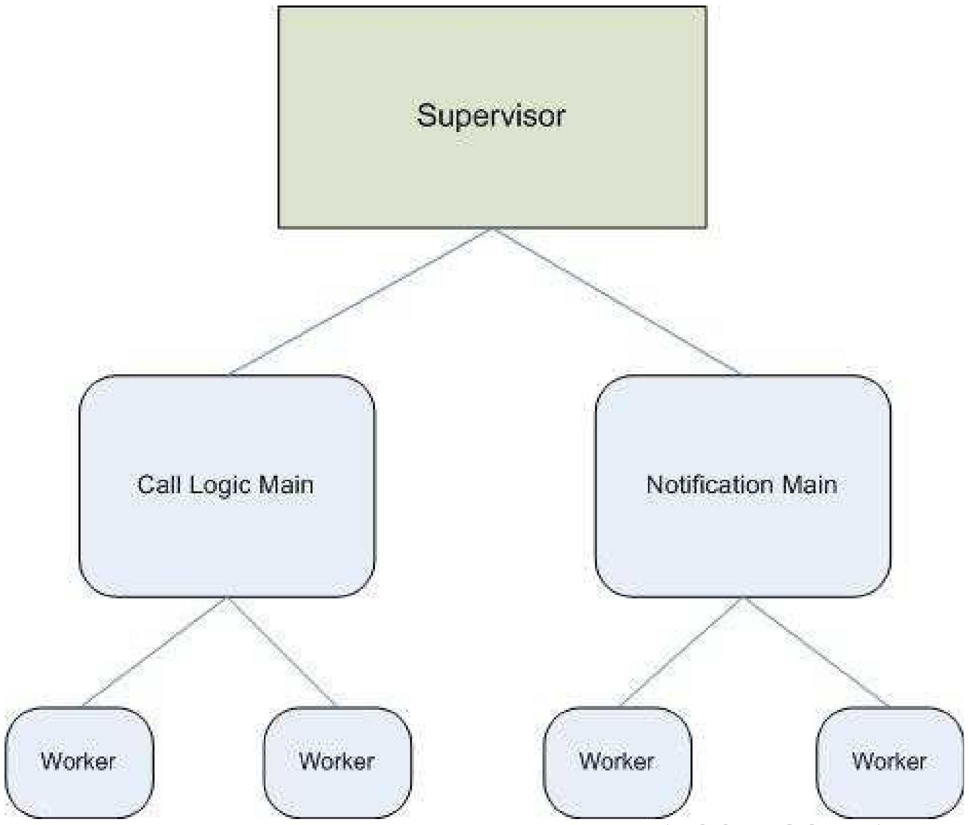


*Fig 3.3, MGC Basic*

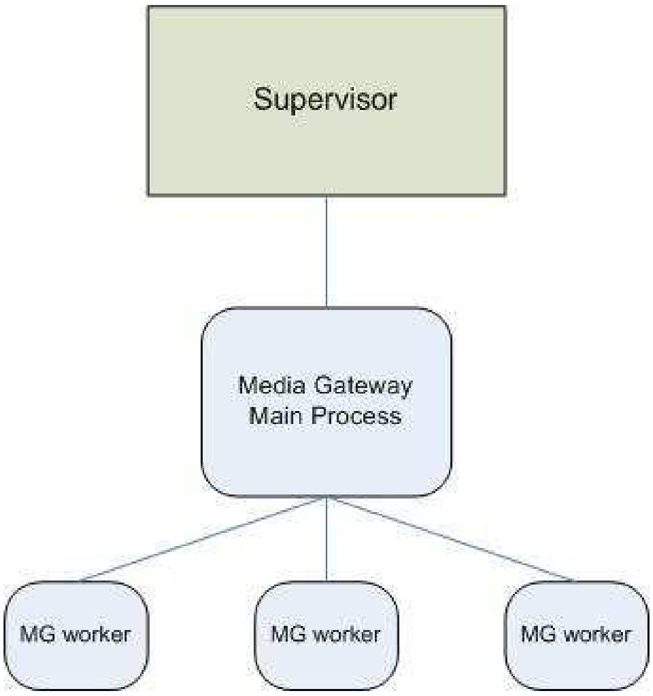


*Fig 3.4, MG Basic design*

The main call logic process needs to listen for SIP and H.248/Megaco messages, the main notification process needs to listen to DIAMETER messages, and the main MG process needs to listen to H.248/Megaco messages. The idea is that each main module will listen for connections, and once a connection has been established, it will start a worker to take care of it, and continue to listen for more connections. After this, only one major issue remains: where and how to store deposited voice mails. The main issue was whether to make the storage an integrated part of the MGC, a part of the MG or a standalone module. As we had decided to make the MG as generalized as possible, making the storage a part of the MG was ruled out first. When considering the remaining options we discovered an already existing module that could do the job with some modifications, so it was decided to use that.

## 3.4 Database design

Since the voice mail server provides the voicemail service to the subscribed user, a database must be maintained. The voice mails can be sent to the mailbox of the subscribed user. The subscribed user should be able to retrieve the voicemails in owned mailbox. After analyzing the basic functionalities of the mailbox, a simple database design is proposed, see figure 3-1. The voice mail box and message are connected by the combination of the mailbox address and message id. A mail box could have many messages. A message is labeled as either read or unread. In the Mailbox design, the pin code is essential to verify the user's identity. The mailbox address, that its form is a SIP URI, is like: <sips:Jimmy@vms.com> .The user can reach the mailbox by dialing the SIP URI. The link in the message table is the clip id from the clip manager where the real media is stored .About the clip manager:

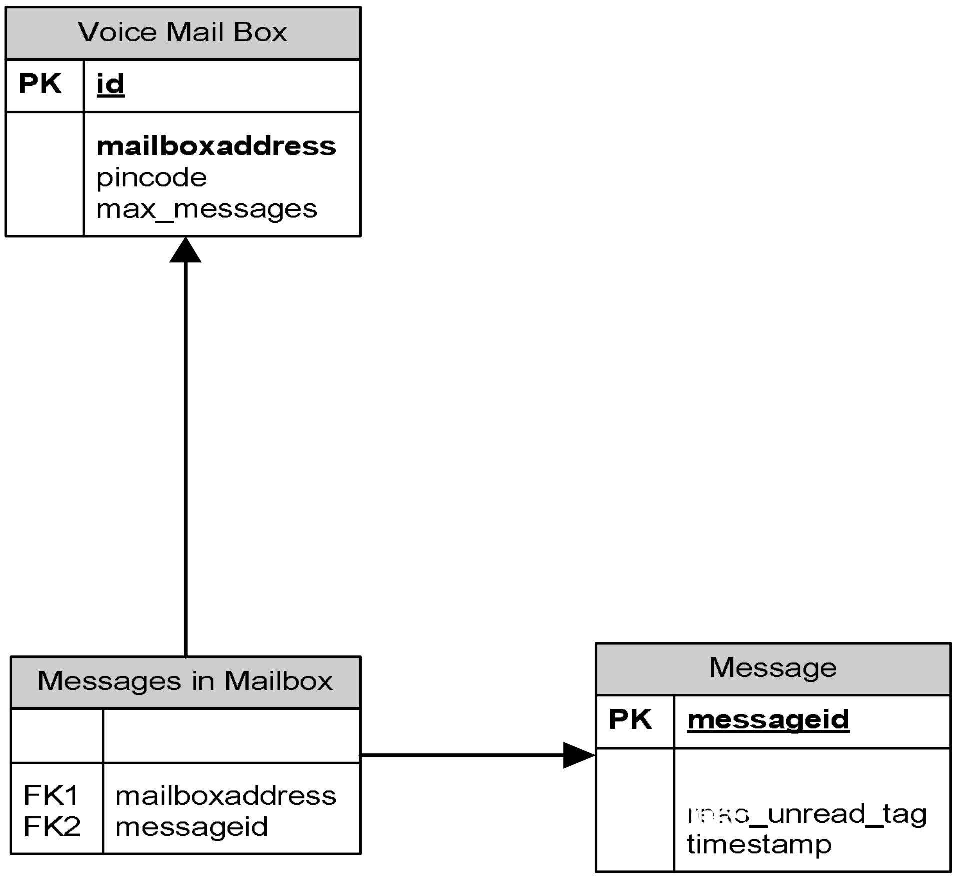


Figure 3-1 Voice mail system database design

# 4 Implementation

The IMS architecture and the role, functions, and interfaces of voice mail server have already been described. This chapter will apply the information and expand on it to describe the implementation of the voicemail application server. According to the system engineering process, the implementation of this thesis can be divided into several stages: specification, design, coding, unit testing, integration and verification.

## *4.1 Development Overview*

*A system is a purposeful collection of interrelated components that work together to achieve some objective.*

The voice mail system development accords with the principle of system engineering. The phase of the voicemail system engineering is shown in Figure 4-1

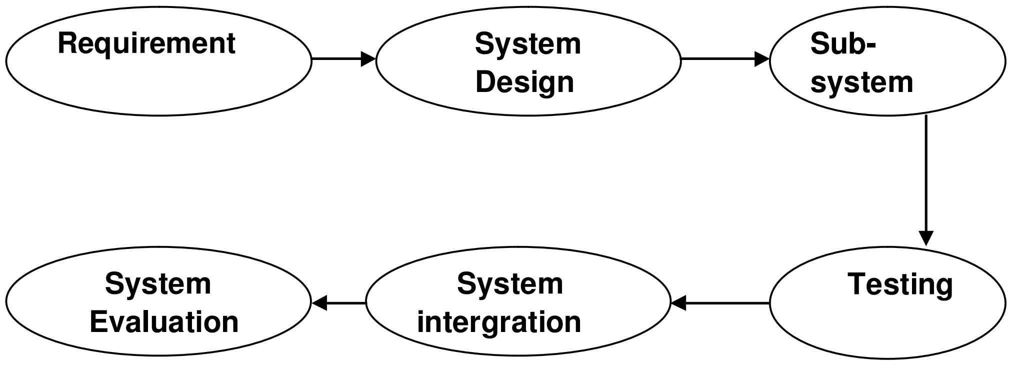


Figure 4-1 Voice mail system engineering process

The software process mode is an abstract representation of software process. This project has adopted both the waterfall mode and the evolutionary mode. For the voice mail system, the requirements are well understood. Since it is a complex large system, the waterfall mode is used during the period of development (see 4-1).

Meanwhile, according to the system architecture design in Chapter 3, the voice mail system is divided into 2 major parts: one contains the signaling method for communication as voicemail controller, and the other one as media gateway contains the media handling for the multimedia stream. Since the interfaces of these stand-alone two parts are well defined, the evolutionary approach is used when developing individually (see 4-2).

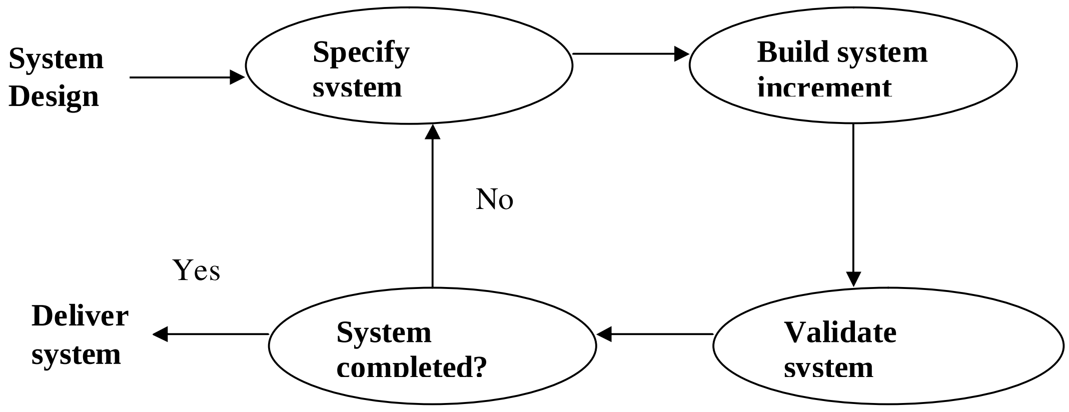


Figure 4-2 Evolutionary software process mode

The specification of the voicemail system is well defined and understood. At the beginning of the project, the basic required functionalities are set, and then the added-on features are developed incrementally according to the actual mount of the work. The following stage is the process of converting the system specification to the executable system which involves the system design and system implementation. Few steps to achieve the goal as milestones are as follows:

Step 1: Investigations are carried out into the IMS architecture and related technologies at the first month. The concept of the IMS architecture and the design pattern of Erlang are studies. Several testing applications are implemented to experiment the various features. For example, the Erlang programming, SIP communication, and setting up the working environment like svn etc. The small applications adds lots of help to understand these technologies used in the final prototype.

Step 2: The primitive design is produced, after acquiring extensive knowledge about the IMS architecture and Erlang programming due to the investigations. The softswitch architecture is applied in the voicemail system in order to separate the call control and media path. Both the internal interfaces among the system components and external interfaces with the world outside are selected by using the different protocols, for instance: Megaco/H.248 between MGC and MG, SIP between IMS CSCF and Voicemail system. All the details of this implementation will be presented later this chapter.

Step 3: Build up the code skeleton of voicemail system using Erlang. The fault-tolerant feature is tested through forcing the process to die and purposely inputting incorrect data. The tests result proves to build up the reliable server by Erlang is a right choice.

Step 4: Through the user-case in the specification, the abstract specification of its services and constraints for the sub-system both MGC and MG is produced. For each of them the interfaces are well designed, and the services are allocated, including the flowing chart of the message between two entities for the interfaces, and the traffic flow chart for the service logic.

Step 5: The development of the program to implement the system follows naturally from the previous step. During this step, data structure and algorithm design are most critical, especially when writing the parser for the protocol stacks. Normally, the testing code is generated every time when a new module is finished.

Step 6: The sub-systems are integrated in this phase. Both component testing and system testing are carried out to meet the requirements of the system. Lots of the time is spent on testing in this phase, because the system integration brings lots of unexpected problems. The interfaces between MGC and MG are well defined that leads to the efficiency on message flows testing.

Step 7: System evaluation took place here. The voice mail system is put into the IMS simulation environment created by the testing team. Meanwhile, the sip phone client has been used to test the system. The evaluation report will be mentioned later on the Chapter 5 evaluation.

From all above, the system development process is given, and each step takes certain mount of time. However, the step 6 (system integration) and step 7 (evaluation testing) took much more than we expected. The whole project is finished within 30 weeks or so.

## *4.2 System Modules*

The voice mail system contains two components. They are media gateway controller (MGC) and Media gateway (MG). They interface by Megaco/H.248 protocol. (See figure 4-3)

### 4.2.1 MGC

MGC (Media Gateway Controller) is the most critical component of the voice mail system. It handles all the service logic and connections from both CSCF via SIP and HSS via Diameter Sh. The MGC controls the flow of the application and describe the application service logic.

It involves how to handle the incoming SIP connections and set up the session, how to communicate with HSS to get the user information data, how to power the media gateway to send RTP packet, how to send the notification message to the end users when they come online and how to properly store the message in the server. In order to answer those questions, the detailed implementation of the MGC is explained here

To integrate the voice mail server to the IMS architecture and set up the conversation session with the client, the SIP protocol stack is implemented. According to the RFC

3264, the SIP interface of the MGC is carried out. More details will be narrated in following sub section about SIP.

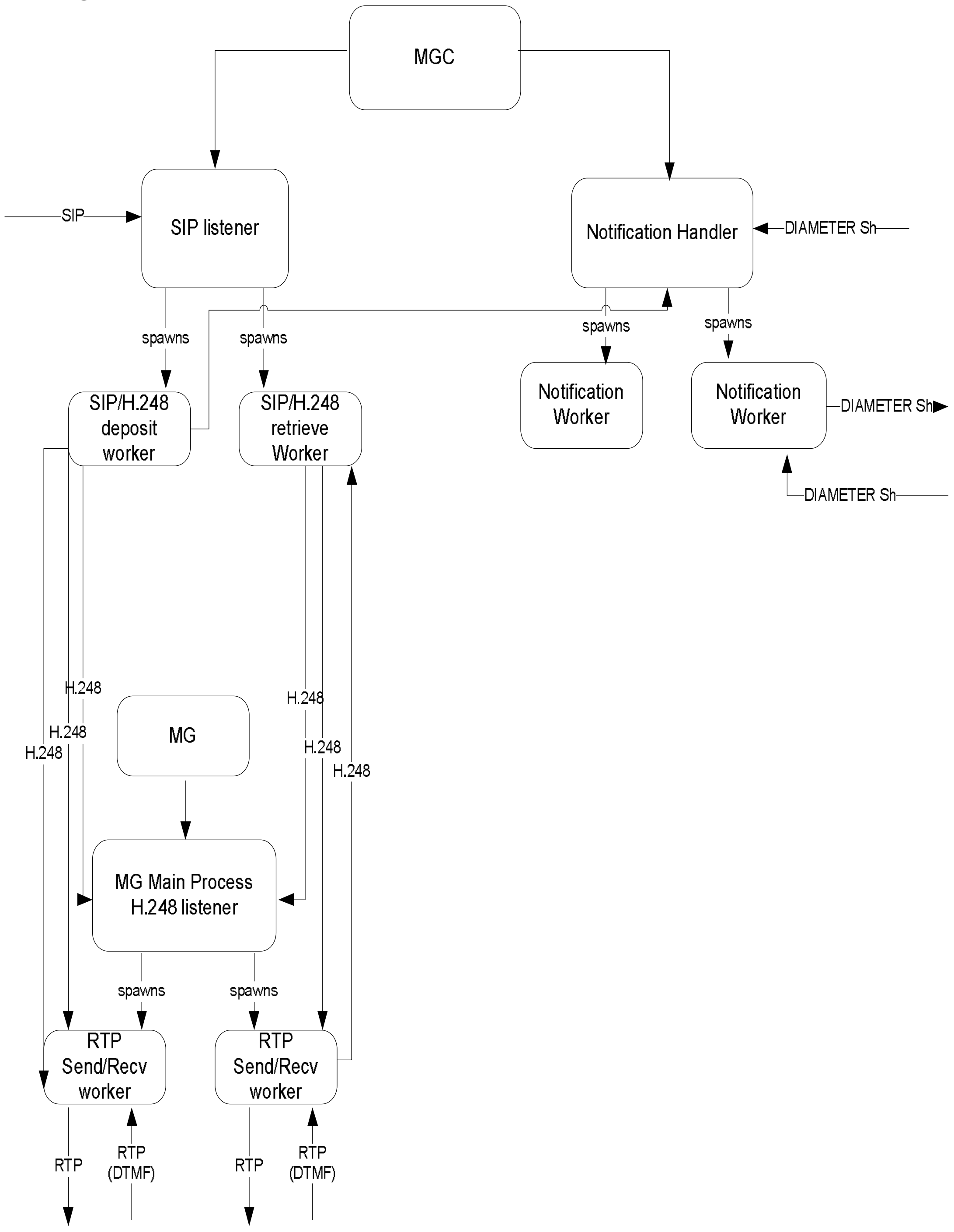


Figure 4 -3 voice mail system overview (MGC, MG and Interfaces)

The next issue is to let the MGC power the MG (media gateway) to complete the media conversation after the session has been set up. MGC uses Megaco/H 2.48 protocol to power on the media gateway. More detailed implementation of Megaco will be discussed in the later sub section.

In order to get the user information from the HSS database, the MGC need have the Diameter Sh interface with HSS. Diameter Sh also provide the possibility to let MGC subscribe the update of the user state. When MGC get the notification from HSS database, a SIP message will be sent to the user who comes online.

Beside all the problems above, one major problem about how to store the message in the voice mail server is left. Mnesia database has been chosen to overcome the obstacle. Mnesia is a build-in database in the Erlang OTP platform. It can be easily accessed and manipulated by Erlang code. According to the database design at the previous chapter, the implementation is carried out. However, in the database, only the link to the message is stored but not the media message itself. The media stream is stored on a Clip manager that is a real-time module created by Mobile Arts.

After finishing all the interfaces and modules of the MGC, the service logic module is generated to combine all the interfaces and other stuff like database module altogether. The functionalities of MGC to fulfill the requirement are presented through the control of service logic.

The problem we encountered during the development of the MGC is that the protocol stack is too generic to implement. To fulfill the functionalities from the requirement, only part of the protocol stack need to be done. Not all the messages have to be taken care by the MGC. For the time being, some message sent to the MGC will be ignored if it has nothing to do with the service logic. Meanwhile, it leaves us the potential possibilities to enhance the capabilities of the MGC.

### 4.2.2 MG

MG (Media Gateway) is the entity of voice mail server that enables the media conversation functions despite of the signaling and call control functions offered by the MGC. As you can see from the figure 4-3, MG is powered by the MGC by Megaco/H.248. Meanwhile, the MG handles the RTP session with the clients like sip phones. The media over IP is received by the MG after the session is set up.

MG involves how to send and receive the media stream, how to enable creation, modification and deletion of media streams including the capability to negotiate the media formats to be used, and how to fetch/store the media stream from/to the disk array.

The first requirement is to enable the MG to send/receive media, and the RTP protocol stack is implemented to fulfill this requirement. RTP is the real-time transportation protocol. More detailed implementation will be mentioned in the following sub section about RTP.

Secondly, the functionalities like creation, modification and deletion of media streams are conducted by using the control protocol Megaco/H.248 between MGC and MG. The requirements also include the capability to negotiate the media format, to report the occurrence of specified events within the media stream (e.g. DTMF tones or call associated signaling). MG should automatically take the action that MGC specified when such events are detected. MG should be able to apply tones or announcements. Those basic functionalities could be met by using Megaco/H.248. The implementation of Megaco/H.248 will be presented later in the interface section.

The final requirement is to let the MG to fetch/store the media stream from/to the disk array. The solution is to integrate the clip manager module created by Mobile Arts to MG. The clip manager is a real-time media storing module that handles the incoming media stream and store them to the disk array and return with a clip id. The media stream can be fetched with the same clip id reversely. Basically, when the MG gets the incoming media from the RTP session with the client, it opens a TCP connection and sends the stream seamlessly to the clip manager. The MG passes the clip id up to the MGC. The service logic inside the MGC will examine the clip id and then store it in the Mnesia database. The service offered by the clip manager is easy to integrate into the MG. Because the clip manager module is written in Erlang and the exported function is handy to be called from other module. This integrated component saved lot of time from developing the MG. It overcomes the difficulty of storing the media stream by using the clip manager.

MG, as a whole, handles the media traffic from the client and it is powered by the MGC that request the media gateway to apply different actions and report the QoS if needed. The RTP stack is implemented as an interface to receive/send media. The Megaco/H.248 is implemented as an interface for the system integration and control protocol.

## *4.3 Protocol Interfaces*

The interfaces exist in the voice mail system are essential to keep the system up and running. Some of them like Megaco/H.248, link the system components together. Some interfaces are used to communicate with other external entities. The thorough explanation about the interfaces is given here.

### 4.3.1 SIP/SDP

"Session Initiation Protocol (SIP) is an application-layer control protocol that can establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls. SIP can also invite participants to already existing sessions, such as multicast conferences. Media can be added to (and removed from) an existing session." ([1] RFC 3261)

SIP, in combination with other protocols like SDP, describes the session characteristics to potential session participants. Although strictly speaking, SIP is written such that the media to be used in a given session could use any transport protocol, the media will normally be exchanged by using RTP as the transport protocol.

SDP simply provides a format for describing session information to potential session participants. Because it only provides the session descriptions and does not provide a means for transporting the session to potential participants, we must use SDP in conjunction with SIP that makes a perfect match. SIP carries the SDP information in the message body. However, we are not going to focus on the structure and syntax of SIP and SDP. Please refer to [1] RFC 3261 for SIP and [2] RFC 2327 for SDP.

SIP is promising because of its natural integration with the IP-world and its flexibility. Before going any deeper into SIP, three important elements in SIP are introduced here .They are UAC, UAS, and SIP Proxy:

1. UAC: User Agent Client, the caller application that initiates the requests, e.g. SIP phone or the logic entity that simulates the UAC behavior.
2. UAS: User Agent Server, accepts, redirect, rejects requests and send response to the incoming requests e.g. voice mail server
3. SIP Proxy: An entity that handles the incoming requests. Mostly it redirects the messages to their destinations. There are several types of proxy servers defined. A stateful Proxy is a proxy that stores server or client transaction state machines. A stateless Proxy only forwards request/response, and do not store any transaction state. E.g. the CSCF in the IMS architecture.

From Figure 4-4, it is about the SIP signaling. Through SIP protocol, the user 1 and use 2 set up the session point to point. Then the media is exchanged by the media path through RTP.

We have specified the entities in SIP transaction. In the implementation, the voice mail server need have a SIP stack and act as a UAS to handle the incoming calls from the client. The voice mail server acts as a User Agent Server. More likely, the voice mail server is like the user 2 from figure 4-4. It handles the requests from the CSCF which is like a SIP proxy server. The requests are generated from the client User 1 that is the UAC.

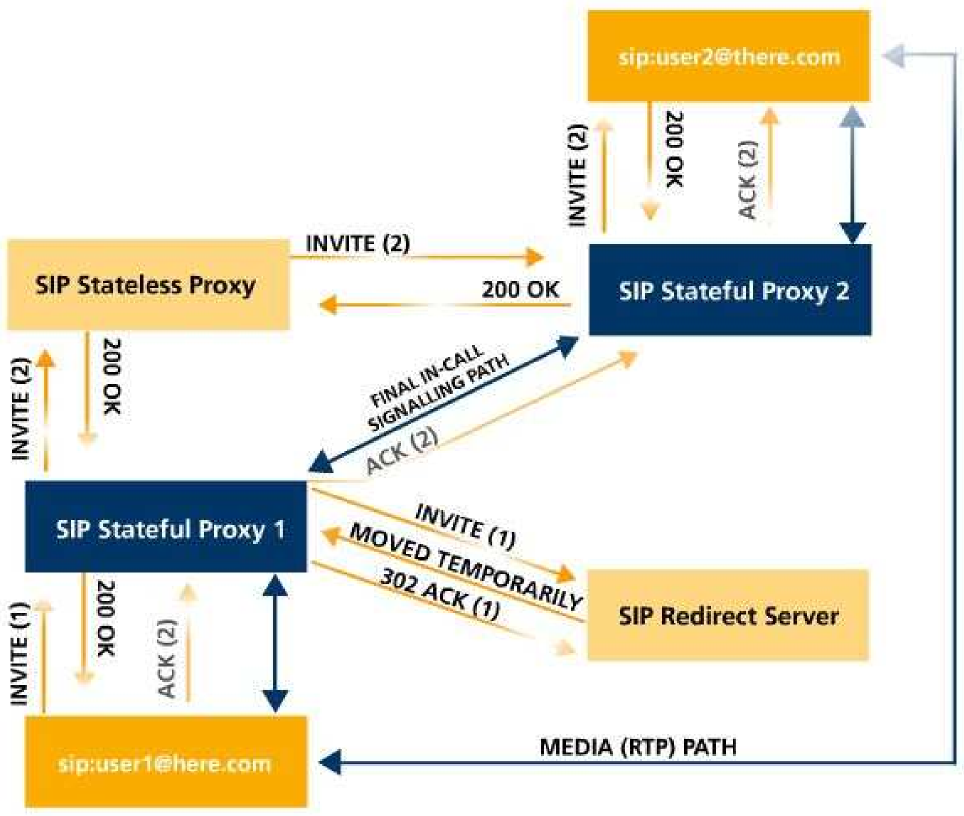


Figure 4-4 SIP Signaling

UAC with address “user1@here.here” initializes a transaction, in this case an “INVITE” request. The request is first sent to the domain SIP Stateful Proxy 1. The Proxy 1 sends the request to an already known server. The redirect server return a response to proxy 1, states that the destination is moved, and provides a new address to the SIP Stateless Proxy. The Proxy 1 responses that, and sends the request to the new address. The SIP Stateless Proxy processes the request and sends it to the Proxy 2, which directs the request to the end user “user2@there.there”. The end user sends a 200 OK message for accepting the request, and the 200 OK response is send all the way back upstream from the path it comes from. When user 1 received the 200 OK Response, it generates a final ACK response and sends it to the Proxy 1. The Proxy 1 now knows the address of Proxy 2, so it sends the final response directly to it. Then the Proxy 2 sends the response to user 2. Now the communication session is up, and through SDP exchanging in the SIP message body, both users can now communicate directly using other protocols.

SIP is essential to set up the session between client and voice mail server. Few messages are exchanged as described above, and the messages contain the SDP in the body. The message flow is like Figure 4-5.The content of the messages being exchanged will be presented in the Appendix at the end of the report.

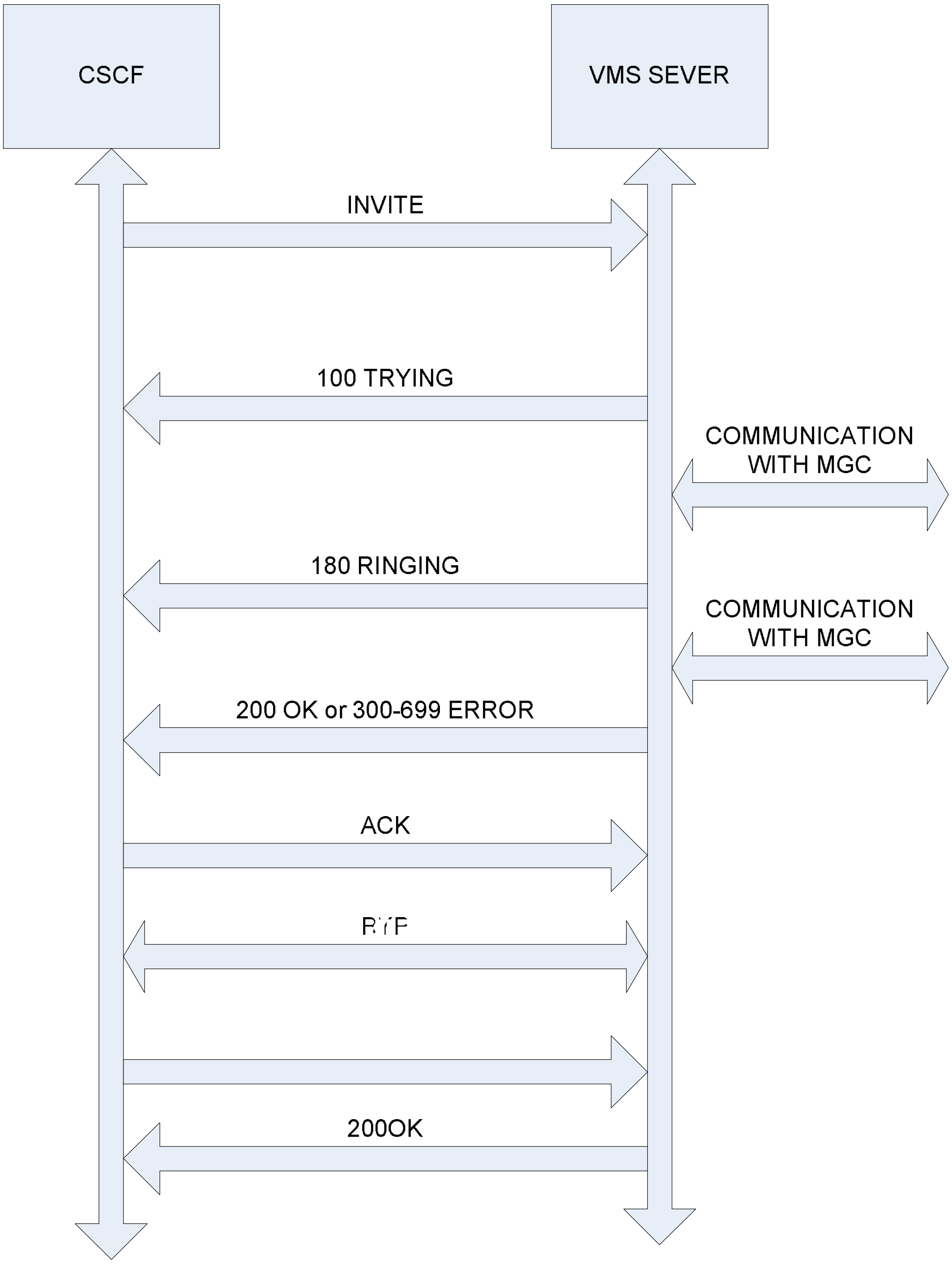


Figure 4-5 SIP message flow of Voice Mail System

The implementation of the SIP protocol has been a major part of the whole project. See the figure 4-6 about the SIP stack.

Since the Erlang does not have the standard library to parse the SIP message. The whole SIP stack is implemented from the scratch. Firstly, the parser for the SIP syntax and encoding is built. Different headers are stored into a key list structure. Secondly, according to the RFC 3261, at the transport layer, the UAS should be able to handle at least TCP and UDP. We fulfill both the requirements but not SCTP which is for secure transportation. The transaction layer for the UAS is maintained before the session is set up. The TU layer handles the message exchange after the session dialog is set up. The transaction layer and Transaction User layer are presented in the service logic when UAS handling the incoming messages.

The SIP stack implemented is also a stand-lone module. It can be reused in other parts of the development where sip message handling is needed. However, the SIP stack only supports methods like: INVITE, ACK, BYE, CANCEL, and MESSAGE, because it is time-consuming and unnecessary to implement a complete stack for this project. However the stack is extensible to add more methods. It will be the future work if the new requirement appears.

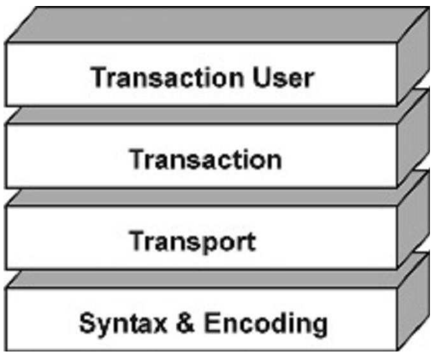


Figure 4-6 SIP stack layers

### 4.3.2 RTP/RTCP

Real-time Transport Protocol (RTP) is a protocol for end-to-end delivery for real-time data. It also contains end-to-end delivery services for real-time data: payload-type (codec) identification, sequence numbering, time-stamping and delivery monitoring. RTP does not provide Quality of Service; it does, however provide QoS monitoring using the RTP Control Protocol (RTCP). [RFC 3550]

RTP runs on the top of UDP. Because UDP does nothing in terms of avoiding packet loss or even ensuring ordered delivery. RTP must ensure that the received packets are presented to the user in the correct order. It also contains the Timestamp to indicate at which time the packet was sampled from its source media stream. The destination application can make use this time-stamp to ensure synchronized play-out to the destination user and to calculate delay and jitter. See the RTP packet format in details below

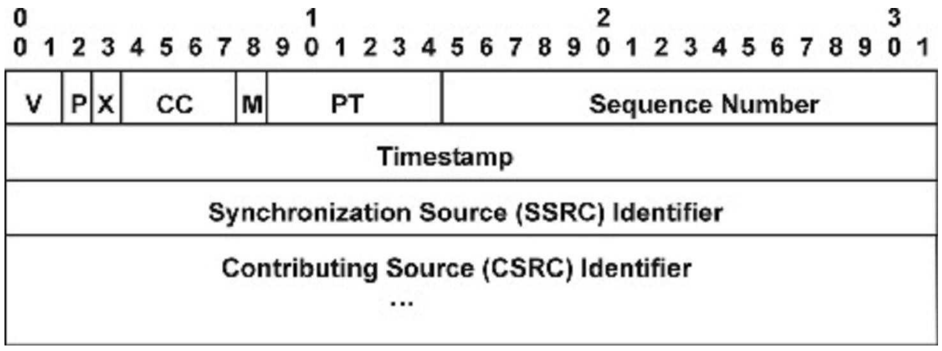


Figure 4-7 RTP Packet Format

The sequence number is to let the destination application be able to detect the loss of the packet and ensure the packets in the correct order.

RTP is a relatively easy protocol to be implemented. It has no other different layers besides the transport and application layer. We build a module to handle the UDP packets and simply put the extra information on the top the UDP packet. RTP is the key element to ensure the media transportation in the voice mail system. As mentioned, RTP has a companion protocol, RTCP, which provides a number of messages that are exchanged between session users and that provide feedback regarding the quality of the session. RTCP is a useful protocol to ensure the QoS, but it is not implemented in the prototype of the voice mail system. It could be an add-on feature once required. Another feature that RTP supports is to specify the payload type of the encoded voice. It handles different encoded voice. When the user fetches the media from the server, the voice mail system must be able to play the required codec through the RTP packet.

### 4.3.3 Diameter Sh

Diameter is an Authentication, Authorization and Accounting (AAA) protocol. Diameter is used to provide AAA services for a range of access technologies. Diameter is loosely based on the Remote Authentication Dial In User Service (RADIUS).

Diameter is a peer to peer protocol and runs on top of TCP or SCTP. Diameter has three different types of the network node: clients, servers and agents. In IMS architecture, for the purpose of performing authentication and authorization functions, there are three interfaces :( Cx, Dx, and Sh interfaces), over which the authentication and authorization are performed, See Figure 4-8 architecture for authentication and authorization in IMS

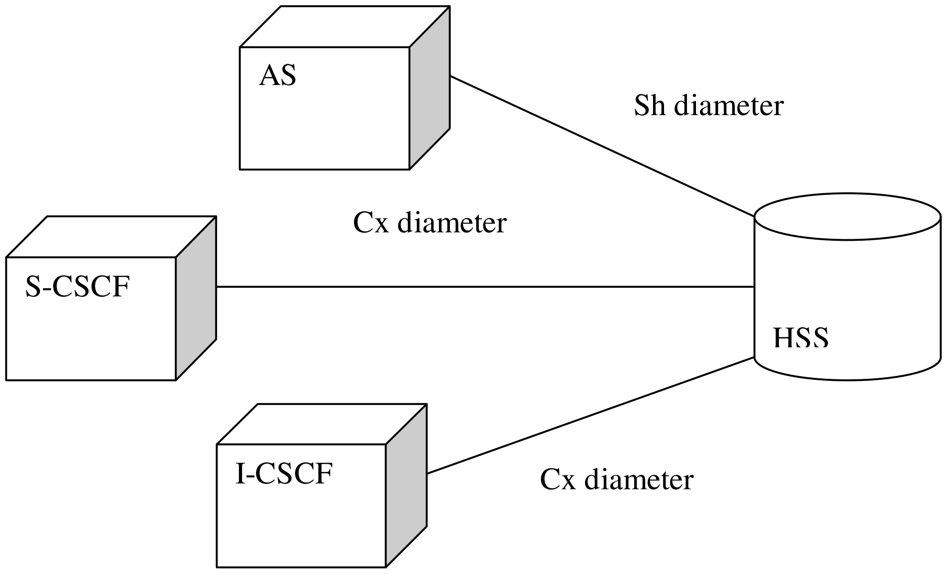


Figure 4-8 Architecture for Authentication and Authorization in the IMS

The Cx interfaces are specified between CSCF and HSS. The Sh interface is specified between AS and HSS. The voice mail server is playing the AS from the figure 4-8. The HSS is the database that stores all the user-related information. In order to get the updated information from the database, The AS connects to the HSS via Sh diameter interface through the authentication and authorization. The implementation of the Sh diameter in the voice mail server is done as a client node in the Diameter. The HSS is the server node in the Diameter.

The Sh interface provides notification and subscription type of functionalities, such as that the voice mail server is notified when the user comes online, and voice mail server subscribes for the change of the user status. The protocol over the Sh interface is Diameter (RFC 3588) with Diameter application (specified in 3GPP TS 29.328[22] and 3GPP TS 29.329[35]). In the voice mail application, it is required that user should be notified when the new message is stored in the mailbox when the user comes online. The implementation of the Sh interfaces is to solve the notification problem. There are few commands codes defined in the diameter application for the Sh interface to fulfill the requirement of the notification, like SNR (Subscribe Notifications Request), SNA (Subscribe Notifications Answer), PNR (Push notification Request), and PNA (Push notification Answer). The voice mail server can subscribe to changes in the user data by sending a SNR message to HSS. When the changes occur in user data stored in the HSS and the voice mail server is subscribed to changes in these user data, the HSS sends a PNR to the subscribed application server. The PNR message includes a copy of the changed data. See Figure 4-9

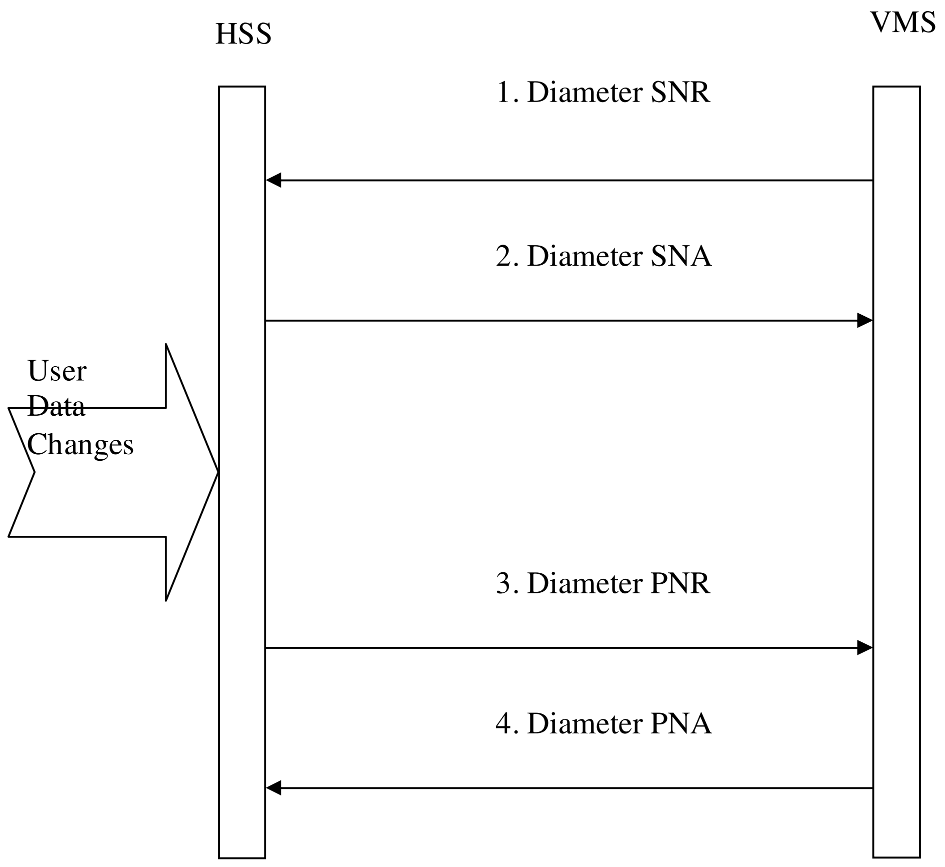


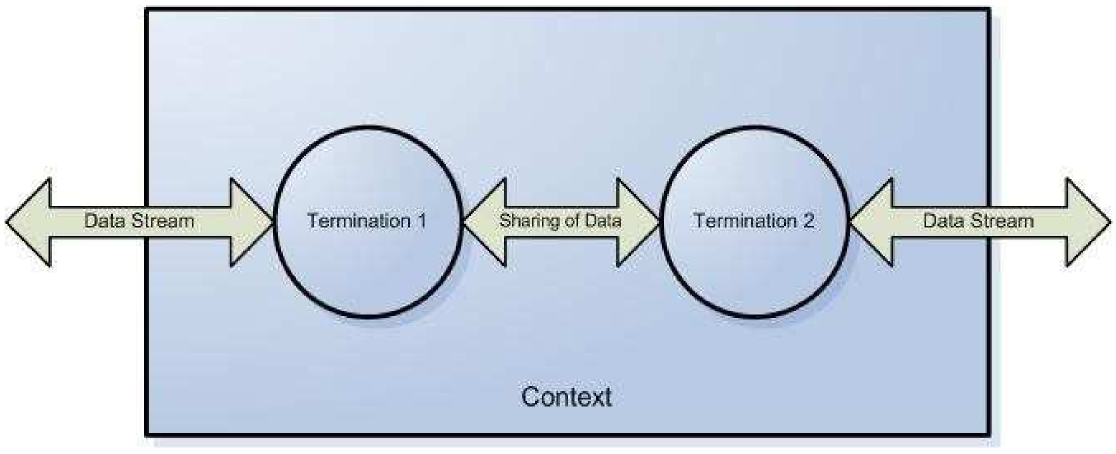
Figure 4-9 SNR/SNA and PNR/PNA messages in Diameter

The diameter protocol over the Sh interface is to let the Voice mail server be able to access the user data in the HSS database. The notification functionality is done by implementing the Sh interface. However, the Diameter stack only applies these 4 commands in the voice mail application. The protocol stack is extensible and the new command can be added if there are requirements.

### 4.3.4 MEGACO/H.248

MEGACO is a protocol that is used for controlling Media Gateways. It was developed jointly by IETF who calls it MEGACO, and ITU who calls it H.248.
This protocol obsoletes the older Media Gateway Control Protocol, MGCP. This protocol involves a series of transactions between two entities called the Media Gateway, and a Media Gateway Controller. In the general case, the MGC will send a TransactionRequest, and the MG will send a TransactionReply, though cases exist were

the MG is the initializing party. The protocol can run on both TCP and UDP, and both a text encoding using ABNF, and a binary encoding using ASN.1 can be used.
MEGACO defines Terminations, which act as sources or sinks of media streams. In the IMS these terminations usually consists of a RTP stream, and only exist for the duration of a call. A context is an association between a number of terminations, with the intention that terminations located within the same context can share media. A termination can only exist within one context at any time, and can not share media with other terminations if they are not located in the same context. For the purpose of this application, a context can be viewed as a call, with two terminations, the caller and the receiver (the database).



MEGACO defines 8 commands that are designed to control and manipulate contexts and terminations, they are:

**Add**
Adds a termination to a context. If no context is given, a new is created. Likewise, if no specific termination is given, a new is created.

**Modify**
Modifies the properties or behavior of a termination.

**Subtract**
Removes a termination from a context. If after the removal, no terminations remain in the context, the context is removed as well.

**Move**
Moves one termination from one context to another.

**AuditValue**
Is used by the MGC to request values for properties, signals or events associated with terminations.

**AuditCapabilities**
Similar to AuditValue, but instead of returning current values, it returns *all possible* values.

**Notify**
Sent by the MG to notify the MGC that an event which the MGC asked to be notified about has occurred.

**ServiceChange**
Sent by the MG to inform the MGC that certain MG-wide events will take place, such as a reboot or a number of terminations being taken out of service. Can also be sent by the MGC when it transfers the control of the MG to another MGC.

A stack for running the MEGACO protocol is included in the Erlang library. This stack implements the transport both over UDP and TCP, including sending of ACKs etc, and supports all text and binary encodings. So for us remains to implement a user callback module that handles all the messages we consider necessary. Since all transport and encoding is already implemented, adding more messages in the future is a simple matter. For the purpose of this application, it was deemed to be enough to implement the ADD, SUBTRACT, MODIFY, NOTIFY and SERVICECHANGE messages. Note however that these commands can contain a number of different parameters and values, some of which are not supported. If a message does not have the expected properties, an appropriate error message is always sent back. Of note when implementing this stack is the requirement that all commands must be executed in the order they arrive. This means that different commands can not be processed simultaneously on different worker processes. Instead the main process must queue up any additional MEGACO commands until the current command has finished executing.

# 5 Evaluation

Evaluation of the system was done in three steps: unit testing, system integration and performance tests.

## *5.1 Unit testing*

The purpose of the unit testing is to test individual functions and modules of the code. The test always starts with the smallest and simplest functions first. The functions are tested with both correct and incorrect values to make sure the outcome is the expected. When doing this we follow the standard practice for testing functions, using typical values, border values and extreme values. Since erlang is not a strongly typed language, we also tested what happens when values have the wrong type. When the simplest functions have been tested the next set of functions, those that use one or more of the simpler functions, are tested. This way, the errors encountered are most likely to be found in the actual function being tested rather than in one of its subroutines, which simplifies the process of finding and correcting bugs. Finally we test the functionality of the module as a whole. When testing protocol stacks, it is important to remember that the input will be provided from an outside source, so the initial checks of the input should handle any errors and not crash or leave the application open to exploits. When testing messages we sent one message at a time, testing the following:

-Correct messages with correct and supported contents
-Correct messages with incorrect or unsupported contents
-Incorrect messages, i.e. messages with syntax errors etc

The result of these tests indicated that the protocol stacks behaved in an acceptable manner, and conformed to the standards. Also, since we did not implement all the messages, only those needed for the project, we verified that the unimplemented messages returned a simple error.

## 5.2 System integration

During this phase, the different modules are combined to one large application, and the complete application is tested. All the requirements are tested in this phase. Errors discovered during this phase tend to be fewer but more serious than errors discovered in the unit testing. The reason for this of course being, that the simple errors are already discovered in the unit testing, while the errors left are the big flaws in thinking about how the different parts are supposed to work together. This is especially true when two modules that work together have been written by different persons. As a result of this, errors discovered in the system integration may lead to a bigger redesign of a module, rather than simply ‘fixing a bug’. When testing the system as a whole, our starting point was the typical user cases. In the first case, we simulated a user that tries to call someone who is not available, and gets redirected to the voice mail system. The first task of the system in this case is to correctly see that it is a user being redirected to the system, and not a user calling the system himself, i.e. a message is to be deposited, not retrieved. In this case, the result was a success as the simulated caller was able to deposit his message, which was also verified by using special tools to look into the database. Next up was a test of someone retrieving this message. This test was successful as well, the received message was the same as the one deposited earlier. In the case of testing the notification, a message from a simulated HSS had to be created, indicating to the application that the user to be notified was now online. A problem with this scenario is that we do not really know where to send the message since we don’t have a fully configured network with Sip proxys etc, so a temporary solution to send the message to a specific target had to be used. The result of this test was also satisfactory. As a whole the testing was a success, with most of the features working as planned.

## 5.3 Performance testing

It was the intention of the authors to also test the performance of the application, measuring such things as incoming calls per second, number of ongoing simultaneous sessions etc. However, the lack of time towards the end of the project led to the decision to skip this step of testing, since setting up an environment for this testing would take a substantial amount of time.

# 6 Conclusion

This is the final chapter of the thesis work and of the entire report. It contains the concluding aspects of the thesis work done so far. This chapter once again focuses on the areas that are identified for improvements in our validation process. Finally this chapter provides a conclusion on the achievement of our goals addressed in Chapter 1 introduction.

## *6.1 Future development*

The final prototype of the voice mail server meets all the initial requirements in Chapter 2 about the voice mail system, however as a prototype, it can be improved in many aspects. Generally there are functionality improvements, availability improvements, interface improvements, and performance improvements.

Functionality improvement involves the ability to add more features to the voice mail system. As we describe in the previous chapter, the user should be able to deposit a message into a mailbox, the user should be able to retrieve/delete a message from the mailbox and the voice mail system should be able to notify the user when there is new message available in the mailbox.

The prototype covers all these aspects, however in the future development, the voice mail server should allow direct voice messaging. The direct voice messaging feature is conceptually similar to SMS or email. The audio SMS is the next goal to achieve, and for this some modifications need to be made in the number analysis module. The project is well modeled and the changes to the number analysis module will not affect the other functionalities that are already working well. So the direct voice messaging is one add-on feature in the future development.

Availability improvement concerns how well the voice mail service can be reached from different devices. Since in the prototype, the address of the mailbox in the system is a SIP URI that means that the user must have a SIP device to reach the voice mail on the server. However in the SIP protocol, normal telephone users are also supported. An example of this is if a user would like to retrieve a message through a public telephone, instead of a SIP device. Through dialing a particular number, the user can receive the voicemails in his mailbox. A mapping between number and mailbox address should be maintained in the voice mail server. When the particular number is being reached, the Telephone URI is converted into the SIP URI. In future development, this is a good way to make the voice mail server more compatible with older devices.

Interface improvement concerns the communication between different components, internal as well as external. In the voice mail system, various protocols are being used to communicate between the different components. There could be a lot of enhancements in the protocol stacks. The voice mail server talks to the CSCF via SIP message. Only a small subset of the SIP methods has been implemented:   INVITE BYE, ACK, and CANCEL. It is able to set up the session with these methods. However, beside the core

SIP protocol as defined in (RFC 3261), the extensions of SIP protocol should be brought in to let the voice mail server have a negotiation mechanism. Three headers: supported, required and unsupported, should be handled in the following development. Even more, in order to maintain the QoS, both server and client need some precondition to establish the session. In some case, if the network can not ensure a certain QoS for the duration of the session, the caller prefers not to establish the session at all. Therefore, the SIP stack should be extended with these add-on features. The interface between the voice mail server and HSS is Diameter Sh could also use some improvements. The implementation for the Diameter stack is not fully completed, because it is not necessary to build up the whole protocol stack to fulfill the requirements of the voice mail service. It can be enhanced in the future development as well. Most importantly, the media is carried by the RTP packets. However, RTCP, which provides a number of messages that are exchanged between session users and that provide feedback regarding the quality of the session is not implemented. RTCP is a useful protocol to ensure QoS, but there was not enough time to implement a RTCP stack in this prototype. In the future development, the RTCP messages should be added to ensure the QoS during the RTP session and to send statistics. For MEGACO messages, the protocol stack is complete, because we use a mature Erlang MEGACO module built by Ericsson. However, in the service logic part, Megaco message handling could be enhanced in some way to handle more different messages to ensure the QoS and have abilities to negotiate with media gateway. Besides all above, the interface improvement also includes the user interface, which is the interaction between the voice mail server and the users. When the user dials the right number to retrieve the message on the server, some announcements should be played and when the user chooses from the options, the DTMF should be handled well by the voice mail server.

## 6.2 Summary

During recent years, IMS has become the next generation telecommunication service framework. It closes the gap between wired and wireless networks, and offers reduced complexity for new services. Many telecommunication companies make huge investments in this area and conduct much research around the topic. Research includes both server side service and client side applications of the IMS network.

The goal of the thesis was to create a stand alone application server into the IMS architecture, which provided a voicemail service to subscribed users. This thesis focused on exploring the possibilities to make such an application server based on existing technologies, and finally implemented a prototype utilizing these technologies. The voice mail service is ubiquitous, however this project attempted to find another solution in the new telecom framework IMS. The result of the work was positive in that a voice mail service was implemented in the IMS environment. Both functional and system integration testing were conducted with fair results. SIP, RTP, Diameter, and MEGACO protocol stacks were implemented in the course of this work, which gained the authors a lot of insight into the IMS world. The modules we built for the communication protocols can easily be used in future development since they were made as general as possible.

Although it is expected that eventually IP will be available on all mobile phones and operators, it is not clear how much of the 3GPP/3GPP2/TISPAN IMS as it exists today will be deployed. According to its proponents, the future of the IMS is to become the backbone of telecom. It can provide users with more exiting multimedia services and operators with an easier way of creating new applications and billing traffic etc. Hopefully this thesis gives an inspiration how to create application servers in the IMS architecture.

# 7 Vocabulary

**3GPP**
3:rd Generation Partnership Project .A collaboration of telecommunications associations with the goal to create globally applicable 3G mobile phone system specifications

**AAA**
Authentication, Authorization, Accounting

**ABNF**
Augmented Backus-Naur Form. A Meta language used for describing formal systems of a language which is a protocol

**AS**
Application Server

**ASN.1**
Abstract Syntax Notation One. A standard that describes data structures for representing, encoding, transmitting and decoding data.

**BER**
Basic Encoding Rules. One of the encoding formats defined in the ASN1 standard

**CAMEL**
Customized Applications for Mobile networks using Enhanced Logic

**CSCF**
Call/Session Control Function

**DCCP**
Datagram Congestion Control Protocol

**DTMF**
Dual Tone Multi Frequency. Signaling used in the voice-frequency band to the call switching center. Used when pressing digits on a phone.

**GGSN**
Gateway GPRS Support Node

**GMLC**
Gateway Mobile Location Center

**GMSC**
Gateway MSC

**GPRS**

Voice mail system for IP Multimedia Subsystem

General Packet Radio Services 3GPP TS 23.060 [27])

**HLR**
Home Location Register
**HSS**
Home Subscriber Server: A master database that provides subscription related information and performs authentication and authorization

**IM-SSF**
IP Multimedia Service Switching Function

**IMEI**
International Mobile Equipment Identity

**IMS**
IP Multimedia Subsystem

**IMS-ALG**
IMS Application Layer Gateway

**IMSI**
International Mobile Subscriber Identity (stored in sim-card)

**I N**
Intelligent Network

**INAP**
Intelligent Network Application Protocol

**ISC**
IMS Service Control

**ISC**
International Switching Centre

**ISDN**
Integrated Services Digital Network

**ISIM**
IP multimedia Services Identity Module

**ITU**
International Telecommunications Union

**MEGACO**
MEdia GAteway COntrol protocol, a 3GPP and ITU-T

Voice mail system for IP Multimedia Subsystem

Jointly specified protocol used for controlling media resources

**MGCF**
Media Gateway Control Function
Does protocol conversion between SIP and ISUP

**MGW**
Media Gateway

**MR**
Message Reference (3GPP TS 23.040)

**MRF**
Media Resource Function

**MSC**
Mobile-services Switching Centre

**PLMN**
Public Land Mobile Network

**PSTN**
Public Switched Telephone Network

**QoS**
Quality of Service

**RTP**
Real-time Transport Protocol

**RTCP**
RTP Control Protocol

**SC**
Service Centre (3GPP TS 23.040)

**SCA**
Service Centre Address (3GPP TS 23.040)

**SCCP**
Signaling Connection Control Part

**SCF**
Service Control Function

**SCTP**
Simple Control Transmission Protocol

**SCTS**
Service Centre Time Stamp (3GPP TS 23.040)

**SDP**
Session Description Protocol

**SGSN**
Serving GPRS Support Node (3GPP TS 23.060 [27]

**SGW**
Signaling Gateway

**SIM**
Subscriber Identity Module

**SIP**
Session Initiation Protocol

**SLF**
Subscriber Location Function
If more than one HSS is used in a system, the SLF maps the user to the right one.

**STCP**
Stream Control Transmission Protocol

**TrGW**
Transition Gateway

**UICC**
Universal Integrated Circuit Card

**UMTS**
Universal Mobile Telecommunication System

**UPSF**
User Profile Server Function

**URI**
Universal Resource Identifier

**USI**
User Service Information

**USIM**
Universal Subscriber Identity Module

# 8 References

**Books& Links:**

[1] "The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds", Gonzalo Camarillo, Miguel-Angel GarcÃa-MartÃn (John Wiley & Sons, 2006, ISBN 0-470-01818-6)

[2] "Carrier Grade Voice over IP", Daniel Collins, 2001

[3] "The IMS: IP Multimedia Concepts and Services" by Miikka Poikselka, Aki Niemi, Hisham Khartabil, Georg Mayer (John Wiley & Sons, 2006, ISBN 0-470-01906-9)

[4] SIP/IMS Technical Portal. http://www.tech-invite.com/index.html

[5] Erlang OTP http://www.erlang.org

**Papers:**

[6] "IP media servers and IMS in voicemail and unified messaging", RadiSys White Paper, April 2007

[7] "A Messaging Service Gateway for the IP Multimedia Subsystem", Johan Persson, Roger Jakobsen, March 6 2007

**RFCs:**

[8] RFC 3550 Real-time Transport Protocol
[9] RFC 2327 Session Description Protocol
[10] RFC 3261 Session Initiation Protocol
[11] RFC 3264 SDP offer/answer model
[12] RFC 3588 DIAMETER Base Protocol
[13] RFC 3525 MEGACO/H.248
[14] RFC 3665 SIP Basic Call Flow Examples

**3GPP Specification:**

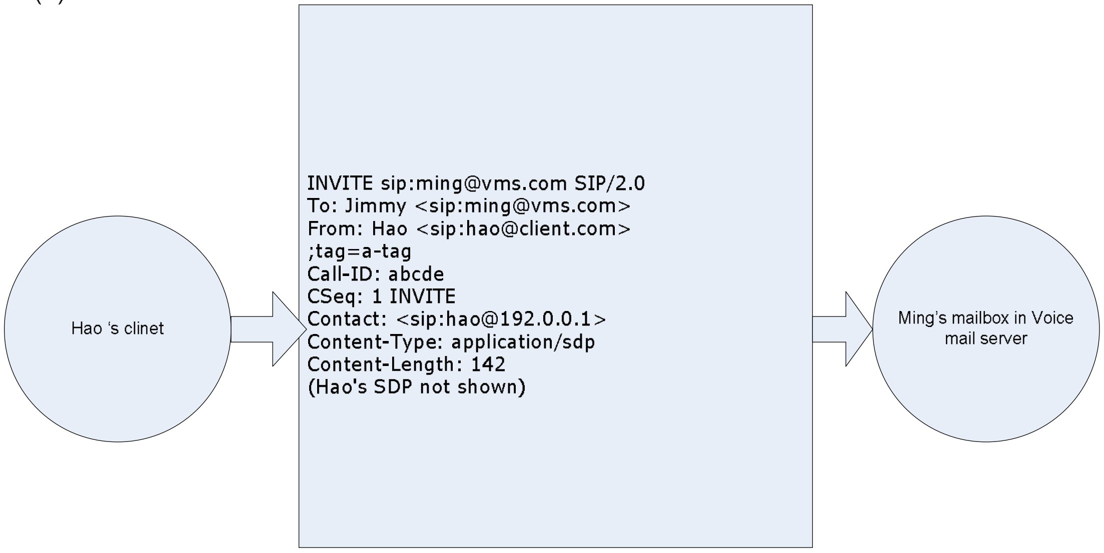
[15] TS 29.328 DIAMETER Sh interface: signaling flows and message content
[16] TS 29.329 Sh interface based on the Diameter protocol; Protocol details
[17] TS 23.002 IMS network architecture
[18] TS 22.228 Service requirements for the IP multimedia core network subsystem
[19] TS 29.228 IMS Cx and Dx Interface: signaling flows and message content
[20] TS 29.229 IMS Cx and Dx Interface based on Diameter protocol, protocol details

**Lectures:**

[21] An overview of the IMS and of voicemail systems, by Lars Kari, Mobile Arts.

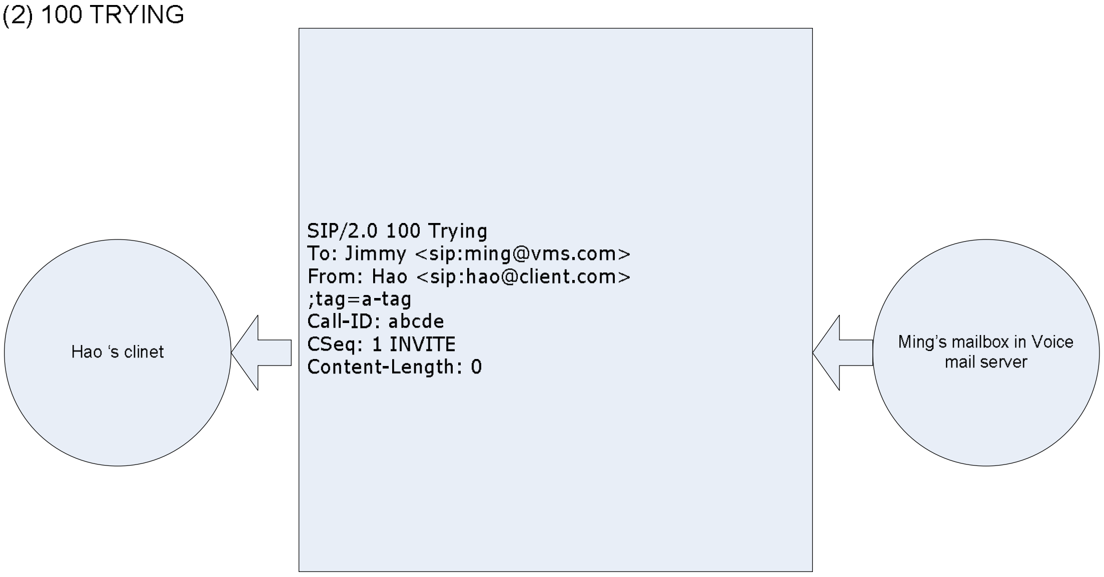
# Appendix A SIP message flow

(1) INVITE

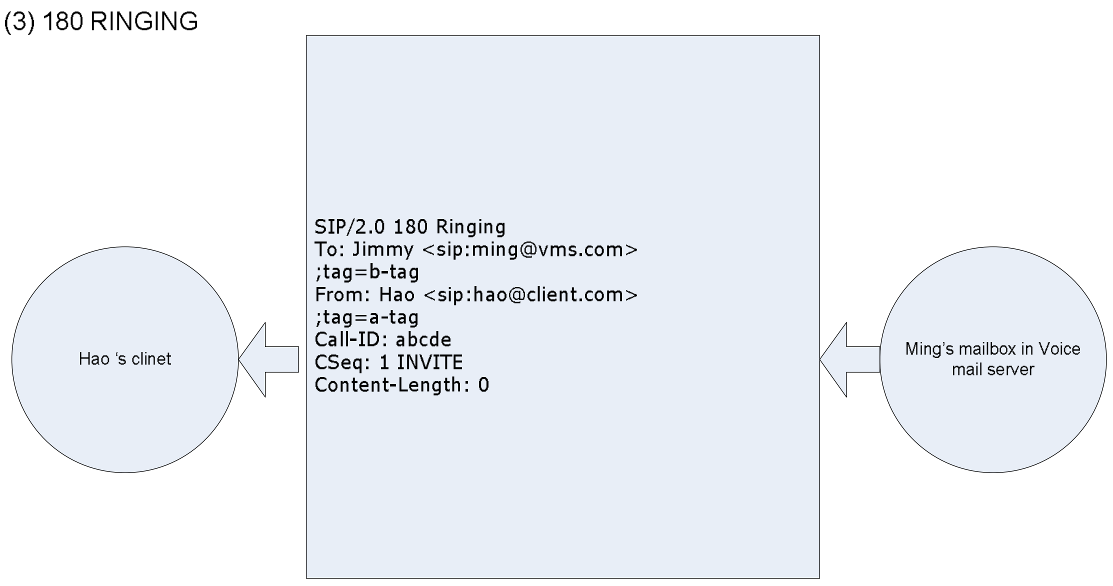


```
INVITE sip:ming@vms.com SIP/2.0
To: Jimmy <sip:ming@vms.com>
From: Hao <sip:hao@client.com>
;tag=a-tag
Call-ID: abcde
CSeq: 1 INVITE
Contact: <sip:hao@192.0.0.1>
Content-Type: application/sdp
Content-Length: 142
(Hao's SDP not shown)
```

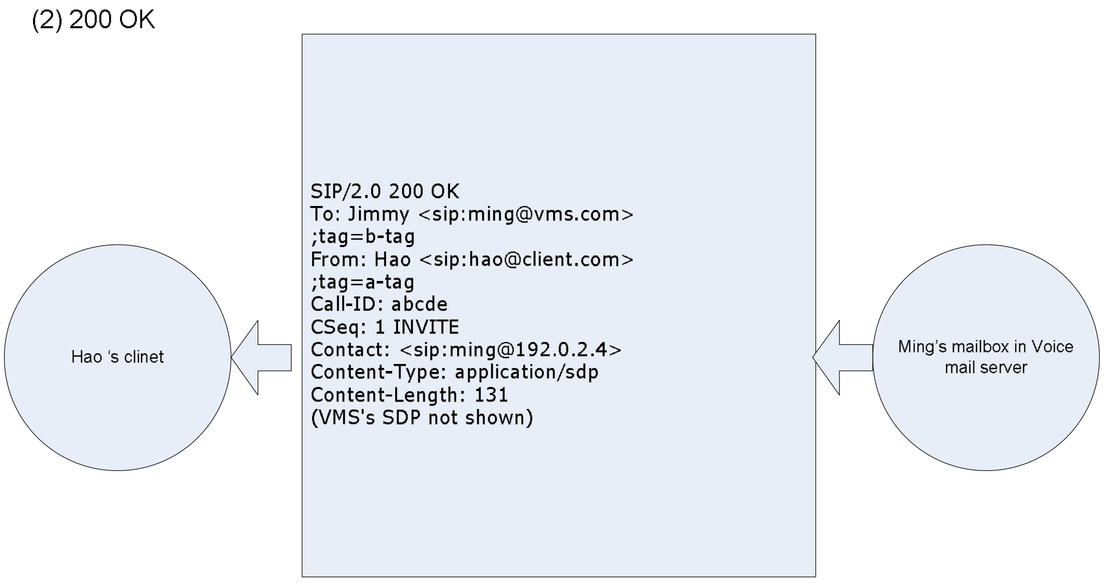
(2) 100 TRYING



```
SIP/2.0 100 Trying
To: Jimmy <sip:ming@vms.com>
From: Hao <sip:hao@client.com>
;tag=a-tag
Call-ID: abcde
CSeq: 1 INVITE
Content-Length: 0
```

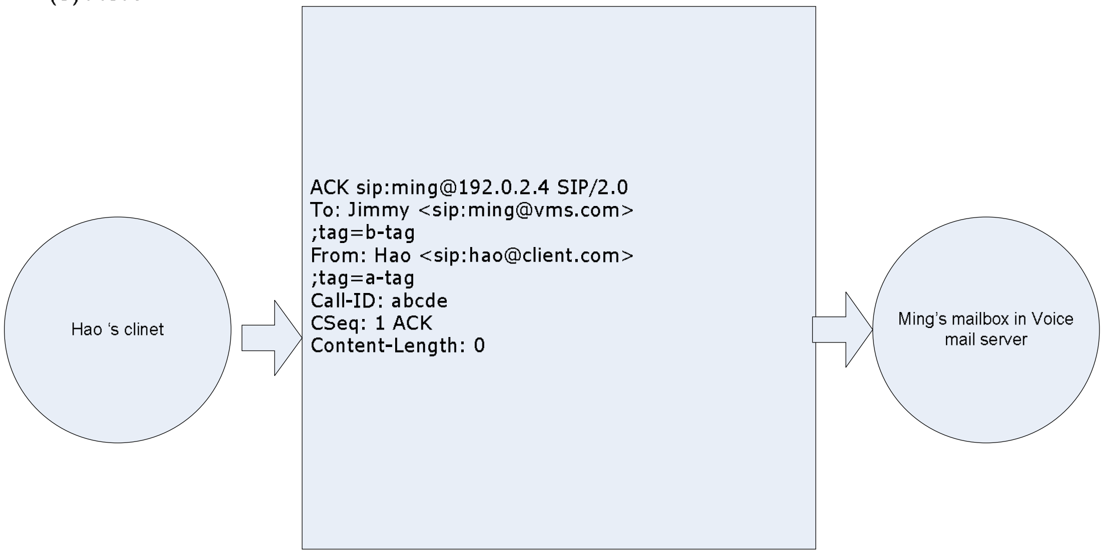


```
SIP/2.0 180 Ringing
To: Jimmy <sip:ming@vms.com>
;tag=b-tag
From: Hao <sip:hao@client.com>
;tag=a-tag
Call-ID: abcde
CSeq: 1 INVITE
Content-Length: 0
```



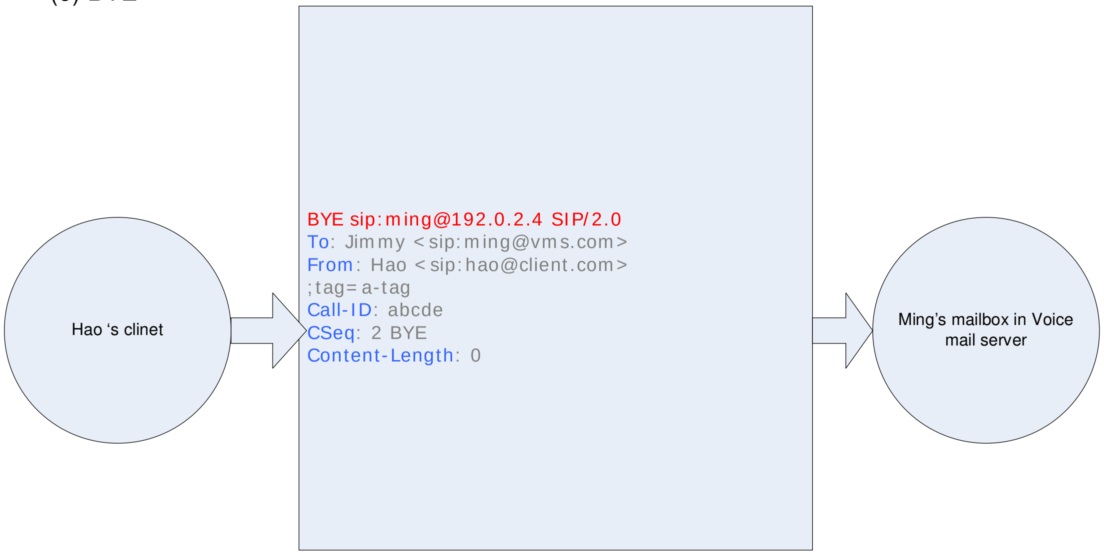
```
SIP/2.0 200 OK
To: Jimmy <sip:ming@vms.com>
;tag=b-tag
From: Hao <sip:hao@client.com>
;tag=a-tag
Call-ID: abcde
CSeq: 1 INVITE
Contact: <sip:ming@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131
(VMS's SDP not shown)
```

(5) ACK



```
ACK sip:ming@192.0.2.4 SIP/2.0
To: Jimmy <sip:ming@vms.com>
;tag=b-tag
From: Hao <sip:hao@client.com>
;tag=a-tag
Call-ID: abcde
CSeq: 1 ACK
Content-Length: 0
```

(6) BYE



```
BYE sip:ming@192.0.2.4 SIP/2.0
To: Jimmy <sip:ming@vms.com>
From: Hao <sip:hao@client.com>
;tag=a-tag
Call-ID: abcde
CSeq: 2 BYE
Content-Length: 0
```

(6) 200 OK     Similar to the previous one

## Appendix B Megaco Message Flow

1) Message from VMS to MG to set up two new terminations

```
MEGACO/1 [IP of MGC]:Port
          Transaction = 1 {
              Context = $ {
                        Add = $ {
                                  Media {
                                          Stream = 1 {
                                                  LocalControl {
                                                            Mode = sendreceive
                                                                           }
                                                            Local {
                                                                        v=0
                                                                        c=IN IP4 $
                                                                        m=audio $ RTP/AVP 4
                                                                        v=0
                                                                        c=IN IP4 $
                                                                        m=audio $ RTP/AVP 0
                                                                        },
                                                            Remote {
                                                                          v=0
                                                                          c=IN IP4 [IP of client, found in SDP]
                                                                          m=audio [Port] RTP/AVP 4
                                                                          v=0
                                                                          c=IN IP4 [IP of client, found in SDP]
                                                                          m=audio [Port] RTP/AVP 0
                                                                          }
                                                              }
                                              }
                                        }
                        Add = $ {
                                Media {
                                          Stream = 2 {
                                                LocalControl {
                                                          Mode = receive
                                                                        }
                                                          Local {
                                                                      v=0
                                                                      c=IN IP4 $
                                                                      m=audio $ RTP/AVP 4
                                                                      v=0
                                                                      c=IN IP4 $
                                                                      m=audio $ RTP/AVP 0
                                                                      }}}}}}
```

2) Reply from the MG with values for the created terminations

```
MEGACO/1 [IP of MG]:Port
      Reply = 1 {
               Context = 1 {
                           Add = Term1 {
                                 Media {
                                          Stream = 1 {
                                                     Local {
                                                              v=0
                                                              c=IN IP4 [IP MG will use for session]
                                                              m=audio [Port MG will use]
RTP/AVP 0
                                                              }
                                                           }
                                          }
                                             },
                                          Add = Term2 {
                                 Media {
                                          Stream = 2 {
                                                     Local {
                                                              v=0
                                                              c=IN IP4 [IP MG will use for session]
                                                              m=audio [Port MG will use]
RTP/AVP 0
                                                              }}}}}
```

3) Message from VMS to MG ordering it to delete a context and all its terminations

```
MEGACO/1 [MGC IP]:Port Transaction = 2 {
         Context = 1 {
              Subtract = Term1 {Audit{Statistics}},
              Subtract = term2 {Audit{Statistics}}
         } }
```

4) Reply from MG with statistics for the closed terminations

```
MEGACO/1 [MG IP]:Port Reply = 2 {
        Context = 1 {
           Subtract = Term1 {
                  Statistics {
                      rtp/ps=12454,            %%packets sent
                      rtp/pr=0,                %%packets received
                      rtp/pl=10,               %%percentage of packets lost
                      rtp/jit=27,              %%jitter
                      rtp/delay=48             %%average latency
                      }
              },
              Subtract = Term2 {
                  Statistics {
                      rtp/ps=0,
                      rtp/pr=12561
                      }
                      }
                      }
                }
```