BILLING SYSTEM DESIGN FOR

VOIP IMPLEMENTED USING SIP PROTOCOL

A Project

Presented to the faculty of the Department of Computer Engineering

California State University, Sacramento

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Engineering

by

Karthik Ram

SUMMER 2012
BILLING SYSTEM DESIGN FOR
VOIP IMPLEMENTED USING SIP PROTOCOL

A Project

by

Karthik Ram

Approved by:

__________________________________, Committee Chair
Dr. Jing Pang

__________________________________, Second Reader
Dr. Preetham Kumar

____________________________
Date
Student: Karthik Ram

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the Project.

_____________________, Graduate Coordinator  
Dr. Preetham Kumar  Date

Department of Computer Engineering
Abstract

of

BILLING SYSTEM DESIGN

FOR VOIP IMPLEMENTED USING SIP PROTOCOL

by

Karthik Ram

VoIP (Voice over Internet Protocol) refers to a communication mechanism in which voice and/or multimedia maybe transmitted over the internet in the form of digitized packets. VoIP quickly became an important part of digital communication and audio engineering discipline because of its versatility to support various features. It advanced tremendously over the last few years mainly because of the internet boom, cost benefits and the flexibility of communication that it offered over the internet.

Depending on their implementations, VoIP sessions can use a wide variety of protocols to establish and terminate sessions. Based on the type of application or scenario in which VoIP is used, the speech signal can be encoded and decoded using different standard codecs before they are transmitted over the network as packets.

The main aim of the project was to design a billing system and write Java code to enable VoIP communication between two computer systems using SIP protocol. This project used open source sip-communicator based code as the main framework to implement VoIP communication. At the Application layer, Session Initiation Protocol (SIP) controlled Signaling (including session initiation, session termination) and
communication sessions while Real-time Transport Protocol (RTP) defined the format for voice/media data over the Internet. This project also used two test accounts created on a back end SIP server to test the VoIP implementation and transmission in real time. A local system Oracle database was designed using (Structured Query Language) SQL and it stored and retrieved all the call statistics such as caller, callee, start time, end time, total duration information, using which the total billing amount was calculated and displayed along with the caller and caller user accounts information.

_______________________, Committee Chair
Dr. Jing Pang

_______________________
Date
ACKNOWLEDGEMENTS

Firstly, I would like to thank my project advisor Dr. Jing Pang immensely, for having provided me this great opportunity to work on billing system design for VoIP implemented using SIP protocol project, which gave me a great exposure to the field of networking, databases and audio processing. This project would not have been possible without the continuous support and encouragement from Dr Pang. She was always there to listen, to give advice and to share her knowledge and expertise in the field of networking, databases and audio processing. I would also like to thank Dr. Pang for being patient and helpful towards the successful demonstration of this billing system designed for VoIP implemented on the eclipse platform, which is widely used across the industry.

I would also like to thank the second reader of my project, Dr. Preetham Kumar, for his unfaltering dedication and support to his students. He also provided me his valuable suggestions to improve the project report and guided me throughout the program.

Finally, the continuous motivation and warmth that I received from my family members and friends were invaluable and they kept me going through all the phases of this project and my master’s degree in general. I would also like to thank my friend, Navin Varma for all his inputs during the course of this project. I would like to dedicate the success of my degree completion and all other achievements throughout my life, to my late father Sri. B. Seetharam Bhat, who was my idol and inspiration.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgements</td>
</tr>
<tr>
<td>List of Tables</td>
</tr>
<tr>
<td>List of Figures</td>
</tr>
</tbody>
</table>

## Chapter

1. **INTRODUCTION** .................................................................1
2. **VOIP OVERVIEW & ARCHITECTURE** .................................7
   2.1 Different types of VoIP connection .................................7  
     2.1.1 Computer System Connection .................................7  
     2.1.1.1 Using Free VoIP Services .................................7  
     2.1.1.2 Using Paid VoIP Services .................................9  
     2.1.2 VoIP Phones .........................................................9  
     2.1.3 Traditional Phones Using VOIP Adapters .................9  
   2.2 VoIP Protocol Architecture ...........................................10  
   2.3 SIP Protocol Standards ................................................11  
     2.3.1 SIP User Agent (SIP UA) ......................................13  
     2.3.2 SIP Server ..........................................................14  
   2.4 RTP Protocol Standards ................................................16  
3. **IMPLEMENTATION OF VOIP** ...........................................19  
   3.1 Introduction ...............................................................19  
   3.2 Requirement Specification ..........................................20 |
3.2.1 Functional Requirements ...............................................................20
3.2.2 Non-Functional Requirements .......................................................21
3.2.3 Hardware Requirements .................................................................22
3.2.4 Software Requirements .................................................................22
3.3 VoIP Implementation ........................................................................23
3.4 Common SIP Methods ..................................................................25
3.5 Comparison of Protocols .................................................................27
4. BILLING SYSTEM DESIGN ..................................................................29
4.1 Introduction ...................................................................................29
4.2 Open Service Gateway Interface (OSGi) ..............................................30
4.3 SIP Communicator Design .................................................................31
4.4 Billing System Design ..................................................................32
5. DESIGN TESTING & ANALYSIS ..........................................................37
5.1 Introduction ...................................................................................37
5.2 Commands Used in Testing ...............................................................37
5.3 Debug ..........................................................................................41
5.4 Testing the Billing System .................................................................43
6. CONCLUSION AND FUTURE WORK ..................................................45
6.1 Introduction ...................................................................................45
6.1.1 Issues faced ............................................................................45
6.1.2 Quality of Service .................................................................46
6.1.3 Security ................................................................................46
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Tables</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Table 3.5.1: SIP Protocol Comparison</td>
<td>27</td>
</tr>
<tr>
<td>2. Table 4.1 Main Ports used in VoIP communication</td>
<td>36</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figures</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Figure 1.1: Generic VoIP connection setup</td>
<td>4</td>
</tr>
<tr>
<td>2. Figure 2.1.1: Trace route for Iptel.org</td>
<td>8</td>
</tr>
<tr>
<td>3. Figure 2.2.1: VoIP protocol level communication</td>
<td>10</td>
</tr>
<tr>
<td>4. Figure 2.3.1: Simple SIP Architecture</td>
<td>13</td>
</tr>
<tr>
<td>5. Figure 2.3.2: Call flow using SIP</td>
<td>15</td>
</tr>
<tr>
<td>6. Figure 2.4.1 RTP header format</td>
<td>17</td>
</tr>
<tr>
<td>7. Figure 3.3.1: General operation in VoIP communication</td>
<td>23</td>
</tr>
<tr>
<td>8. Figure 3.3.2 GUI window for placing VoIP calls</td>
<td>25</td>
</tr>
<tr>
<td>9. Figure 4.1.1 Components of the billing system</td>
<td>34</td>
</tr>
<tr>
<td>10. Figure 4.1.2 Billing system page showing VoIP call statistics</td>
<td>35</td>
</tr>
<tr>
<td>11. Figure 5.1.1 Output of describe login script</td>
<td>38</td>
</tr>
<tr>
<td>12. Figure 5.1.2 Empty Login table before any signup</td>
<td>38</td>
</tr>
<tr>
<td>13. Figure 5.1.3 Filled up login table after signup</td>
<td>39</td>
</tr>
<tr>
<td>14. Figure 5.1.4 Output of describe transaction script</td>
<td>40</td>
</tr>
<tr>
<td>15. Figure 5.1.5 Filled up transaction table after placing calls</td>
<td>41</td>
</tr>
<tr>
<td>16. Figure 5.1.6 Tomcat server errors</td>
<td>42</td>
</tr>
<tr>
<td>17. Figure 5.1.7 PID that was using port 8080</td>
<td>42</td>
</tr>
<tr>
<td>18. Figure 5.1.8 Empty billing history</td>
<td>43</td>
</tr>
<tr>
<td>19. Figure 5.1.9 Updated billing history</td>
<td>44</td>
</tr>
</tbody>
</table>
CHAPTER 1

Introduction

With time, the ability to communicate efficiently over long distances became an important part of society and businesses worldwide. A few decades ago when internet technology was not available, the main mechanism to convey messages, make calls and communicate was circuit switching based Public Switching Telephone Network (PSTN). As the digital and wireless technology advanced, several forms of communication mechanisms came into existence such as Pager, Fax, Cell Phones, and Email. However, all of those mechanisms except Email used circuit switching based networks, which increased the cost of communication since they used dedicated lines to establish one connection. In addition, earlier cell phone and landline communication devices did not support conferencing feature with multiple people at the same time. Tremendous growth in internet and multimedia technology combined with the need for a cheap means of communication over long distances fuelled the birth of Voice over Internet Protocol (VoIP) and VoIP based services. Further, the electronic devices in the emerging market like smart phones, tablets, ultra books, notebooks, laptops that supported internet communication led to the popularity of VoIP and IP telephony.

Packet switching based VoIP communication involved voice and multimedia data transmission over the internet, rather than a telecommunications network. This technique effectively reduced the cost of communication since it used the existing internet infrastructure in place and broadband line to route the calls as digitized packets over the network. Two or more devices that were on a network, placed free calls to each other,
shared data files seamlessly and conferenced with multiple people simultaneously through VoIP, which were not possible in earlier landline or mobile phones. This provided a rich user experience and hence the demand for Voice over Internet Protocol communication grew enormously over the last decade. While some of the VoIP services worked only on a computer system or a special VoIP phone, other services allowed you to communicate using a traditional phone connected to a VoIP adapter. The next chapter highlights an abundant set of useful sophisticated features that VoIP supported and offered freely, in addition to the basic ones that made deployment of VoIP in personal and business solutions affordable.

Several designs for VoIP communication were implemented using different standard protocols, each offering its own advantages and disadvantages. SIP communicator design was one such implementation of VoIP, which used the existing network in order to establish connection between the two participants and made it easy for them to communicate. Of all the signaling protocols, SIP Communicator design project chose Session Initiation Protocol (SIP) because it was the most lightweight, widely accepted industry standard protocol, which ensured greater reliability of communication and signaling. In addition, VoIP design based on SIP protocol was also compatible with most of the industry standard transmission protocols, including the Real-time Transport Protocol (RTP). RTP protocol in conjunction with Real-time Transmission Control Protocol (RTCP) specified the voice and media data format and controlled the Quality of service and synchronization of data packets during real time transmission in this project.
The sequence of steps on the transmitting side of the system in a VoIP communication call included, signaling. This was followed by the media channel setup. Once the channels were set up, digitization of the analog voice/video signals took place. These digitized signals were then encoded and transmitted as Internet Protocol (IP) packets over a packet-switched network such as Internet. On the receiving side, the received IP packets were decoded and subjected to digital-to-analog conversion. This reproduced the original voice/video stream [1].

Figure 1.1 shows various ways in which different communication devices interacted with one another over the internet when a VoIP connection was established. Personal computer systems (like desktops, laptops, ultra books, notebooks, tablets), smart phones, VoIP phones and standard phones were used as end devices to establish connection and communicate with each other without any boundaries. Modem devices connected personal computers and IP phones to the internet directly which in turn allowed them to make calls using VoIP applications installed in them, while ordinary phones required a special adapter to be connected to the internet through the modem in order to make VoIP calls. On the other hand, smart phones directly connected to the internet through their Internet Service Providers (ISP) and using their installed VoIP based services, took advantage of placing calls over the internet. Although the fundamental frequency of speech signal ranged from 85 to 180 Hz for males and 165 to 255 Hz for females (which were below the voice frequency band supported in the telephony that ranged between 300 Hz to 3400 Hz), there were enough harmonics present in the speech signal which in turn recreated the impression of hearing fundamental tone.
VoIP enabled numerous big companies to combine their communications and their networking infrastructures. This was the biggest advantage that VoIP offered over the regular telephone system. This allowed joining of voice and multimedia services together. A single broadband line allowed multiple calls along with multimedia broadcast. This meant significant reduction in the number of dedicated leasing lines and hence cost reduction.

The free VoIP solutions had a few drawbacks. For instance, they did not offer custom service like the ability to send and receive fax over VoIP, which was an important aspect to the success of the business segment. They lacked security and mobility since all forms of communications required connection to public internet. As there was no
dedicated bandwidth for a particular user, adding more number of users on the same network caused traffic congestion resulting in loss of data packets and poor Quality of Service (QOS). Since the free VoIP communication used only the public internet infrastructure that was available, the calls and video data packets had to traverse a regular path (that involved taking multiple hops) to reach their destination. This resulted in a poorer quality of communication because the router introduced latency along each hop of the network node. There was no dedicated support for maintenance as well, in case there was an issue with the connection or an outage. The free VoIP services that the open source applications offered were adequate for the household consumers but not for the small scale, medium and large-scale companies, which depended upon the QOS and reliability of their communications. This led to the beginning of paid VoIP services by several vendors who offered to overcome most of the limitations that existed with the free VoIP implementations. Paid VoIP services were still more cost effective, efficient and offered more sophisticated options than compared to the traditional PSTN based telephone lines. This motivated the second stage of this project, which involved designing a billing system for the VoIP implementation.

This project dealt with the intricacies that were associated with the implementation of open source VoIP communication and the challenges it faced. It also provided an adept insight into various implementations that were prevalent and their differences.
Chapter 2 gives VoIP overview and architecture, explains the details about VoIP protocol, its architecture, and the format of signaling, communication protocols used in the implementation.

Chapter 3 shows implementation of VoIP, specifies the functional and non-functional specifications, hardware requirements, software requirements for this project. It also gives details on the technical aspects surrounding VoIP mechanism, including the comparison of the main communication protocols used with the ones that were not used in this implementation.

Chapter 4 presents OSGI framework, describes the actual design of the VoIP system in general and billing system in particular.

Chapter 5 shows design testing and analysis.

Chapter 6 gives conclusion and future work.
CHAPTER 2

VoIP Overview and Architecture

The previous chapter introduced the VoIP technology and highlighted the motivation behind the billing system design for this project. This chapter covers the how VoIP connected the users over the internet, the main VoIP architecture and the protocols that this SIP communicator project used, in detail.

2.1 Different Types of VoIP Connections

Each of the VoIP connection mechanisms required internet connection (Broadband or dial up) and had its own advantages and disadvantages. Depending on the usage scenario and priorities, the corresponding VoIP implementations were deployed.

2.1.1 Computer System Connection

Computer systems that used VoIP, connected either using free VoIP services or paid their service providers for custom services.

2.1.1.1 Using Free VoIP Services

Two or more computer systems that used free VoIP application, communicated with each other using the public internet. The main advantages of this system were that it used free resources, existing infrastructure and did not require any additional investment. The main disadvantages were that, it did not offer custom solutions or dedicated support, lacked security and mobility, no dedicated bandwidth, which often caused traffic congestion leading to loss of data packets and poor Quality of Service (QOS). The data packets had to traverse a regular path (that involved taking multiple hops) to reach their
destination. This resulted in a poorer quality of communication because the router introduced latency along each hop of the network node. When we tried to connect to the Iptel server, the number of hops it would take before a connection was established from this computer system in Boise, Idaho location would be around 28 hops, as seen from the trace route command output in Figure 2.1.1. The delays seen in milliseconds were the latencies introduced by each router along the way. In this test an average of three trials was sent by the router every time for each hop, before the system could successfully connect to the server and get a response. Therefore, when a larger voice/video data packet was transmitted through this kind of regular connection mechanism using public internet, a lag or poor quality was observed sometimes.

Figure 2.1.1 Trace route for Iptel.org
2.1.1.2 Using Paid VoIP Services

To overcome the disadvantages of free VoIP services, the internet service providers offered custom services to their customers at a premium price. It included assigning a dedicated bandwidth, round the clock customer support and access to their customized network configured through their backend servers. This method allowed the subscribed users to connect to their destinations taking a much shorter path over their network compared with a regular path over public internet. As these services improved the customer’s VoIP communication experiences, there was a need for proper billing mechanisms that ensured that the customers would only pay for the services they actually used, rather than a flat rate. This prompted us to design a dynamic billing system for VoIP communication that could be used by the users and the service providers alike, to access and monitor the call statistics.

2.1.2 VoIP Phones

VoIP phones offered many advantages. They offered mobility and direct connection to the broadband. In addition, they operated just like a traditional telephone. By using VoIP phones, customers could avoid paying for both a broadband connection and a traditional telephone line.

2.1.3 Traditional Phones Using VoIP Adapters

VoIP adapters allowed the existing traditional phone users, who did not want to invest in new VoIP phones, to switch over to the VoIP communication mechanism by just using an additional piece of hardware. VoIP adapters managed the interface conversion from the traditional analog phones to the digital packet conversion before
transmitting data over the internet. The advantage of this mechanism is that it allowed customers to place calls during power outage (provided the service provider had a backup power supply) unlike some VoIP phones that did not work during power outages [2][9].

2.2 VoIP Protocol Architecture

Various steps involved in VoIP communication at the system protocol level are shown in Figure 2.2.1 below [4].
Multiple User Agents are connected to each other over the internet network through the SIP servers. For simplicity, the Figure 2.2.1 showed only two user agents involved in the VoIP communication. The user agent 1, referred as the caller, may initiate the communication by sending out an INVITE packet, as per the SIP protocol standards operating at the application layer of the client system. The SIP server is responsible to forward this request to the corresponding SIP user Agent 2, as the intended destination. The SIP protocol that operates at the Application Layer of the user agent 2, responds by sending out a 200 OK response packet that indicates the acceptance of this session. Next, the 200OK packet will be transmitted to the user agent 1, which in turn will send back an ACK packet to indicate acknowledgement. Once this communication session is established, the G722 codec implemented in the sip-communicator application will convert the analog voice signals transmitted through the microphone to the digital format. These digitized encoded packets, are then transmitted through the RTP protocol implemented at the application layer. The backend Iptel SIP servers implement the real time data transmission and routing. Once these encoded packets arrive at the destination, a reverse decoding process will be applied to recover the analog voice signals at the callee side client system.

2.3 SIP Protocol Standards

Session Initiation Protocol (SIP) helped to set up voice-over-IP calls. Many applications used SIP to support features such as text and general messaging. SIP was also used to signal transport layer UDP protocol, which transmitted the user data, such as voice in this scenario, after it was formatted by the application layer Real-time Transport
Protocol. The architecture of SIP protocol was similar to that of HTTP client-server protocol and since SIP was a lightweight, text-based protocol, it was widely adopted as the industry standard. SIP was IETF’s preferred standard for establishing VOIP connections and was identified by the RFC 2543 standard, until it became obsolete and was replaced by the updated RFC 3261 standard. It operated at the application layer and was mainly developed to create sessions, modify sessions by using session descriptors and terminate sessions with one or more user agents. Requests were originated at the client and were sent to the server. The server processed those requests and sent back a response to the client. If the SIP entities allowed independent request–responses, they formed stateless entities. If they kept track of the request–response sequence, then they formed stateful entities [6]. VoIP communication used stateful entities since it was important to keep track of the sequences of the request-response sessions to make complete sense in communication and synchronization. SIP depended on the Session Description Protocol (SDP) for carrying out the negotiations. It allowed user mobility by supporting proxy and redirection requests to the user’s current location. Further details will be explained in the next chapter.

The basic building blocks of SIP protocol were [5]

- SIP user agents: Included IP phones or Personal Computer running the application.
- SIP redirect servers: Returned new locations for requests in case of congestion or unavailability.
- SIP proxies: Routed all the call requests to the appropriate destinations.
A simple SIP Architecture indicated by the Figure 2.3.1 below was recommended by the RFC 3261 standards for implementation of VoIP using the Session Initiation Protocol. It consisted of two main modules.

**Figure 2.3.1: Simple SIP Architecture**

### 2.3.1 SIP User Agent (SIP UA)

SIP UA referred to the actual user interface that was required to interact with the system in order to use all of its functionalities. User agent could be hardware or software. In a network that had broad system support functionalities, a special hardware device such as a VoIP phone would act as the UA. In this project, the SIP user agent was the SIP-communicator application that ran on the computer systems. It was also packaged into a platform independent executable JAR file.
2.3.2 SIP Server

This represented the core component of the SIP system architecture. The Iptel SIP server was the main module responsible for the smooth and efficient functionality of the VoIP system. A daemon process always ran in the background that helped to set up connections, accept new users into the system and kept track of activities performed.

- The Registrar was the module of the SIP server that handled all the registration and authentication activities of the system. It kept record of all the registered users with the system, and thus provided the base for authentication activity [5].

- Proxy Server module of SIP server acted as the main kernel which handled and coordinated all the activities of the system. It was responsible for handling all the activities related to voice transfers and rendered the same with negligible delay [5].

- Redirection Server came into picture when the other end user was unavailable at that destination. In this case, the Redirection server forwarded the call directly to the user’s new destination [5].

Thus, SIP provided the following services: [5] [10]

- Determined user location and established connection.

- Checked user availability.

- Determined the media parameters supported in the communication.

- Setup ringing and established call parameters.
Handed calls.

The following Figure 2.3.2 published by Microsoft Real-Time Communications: Protocols and Technologies article in 2003 [8], explained the basic mechanism involved in VoIP call based on SIP. For Example, two persons A and B wanted to communicate over the Internet. Part of this communication would be done using SIP. Both parties that were interested in communication had to use a user agent.

![Call flow using SIP](image)

The first thing A would do to get started was to send a message to B on the standard SIP port (5060/5061). This message called INVITE message requested call session initiation. When B received this INVITE message through one of its standard SIP ports, it responded back by sending a message with code 100 that indicated that B was trying to accommodate this request. This would kick off the timer and A would have to
wait until it timed out, before it could retry. In the meantime, if user agent B was available, it would send out a 200 OK to which user agent A would respond with an ACK message. At this point, both parties would know what type of media to expect, the IP address ports that would be used for the communication and other details. This would start the real-time communication over the established session utilizing the RTP protocol. When the communication completed, either A or B would send a BYE message to end the session and the other user agent would acknowledge it with a 200 OK.

2.4 RTP Protocol Standards

Real-time Transport Protocol standards known as RFC 1889, developed and maintained by the Internet Engineering Task Force (IETF) were used in most of the early Audio real-time applications. They were then superseded by RFC 3550, which included more usage scenarios and supported compatibility with other protocol suite. RTP when used along with the Real-time Transport Control Protocol (RTCP) and SIP, allowed end-to-end network transport features and provided complete control to the applications transmitting real-time data, such as audio/video [11]. However, RTP used the underlying UDP or other transport layer protocols for their multiplexing and checksum services. An RTP data packet consisted of fixed RTP header, a list of optional contributing sources and a payload data. While RTP carried the media streams (audio/video), RTCP monitored transmission statistics and quality of service (QoS) and helped in synchronization.

The first 12 bytes seen in Figure 2.4.1 of the RTP header format were mandatory, while the CCRC and the other fields were optional and depended on the application
scenario. RTP header was followed by the RTP payload, the format of which again was decided by the application.

RTP header had a 2-bit Version field, which indicated the version of the protocol currently used. It also used a 1 bit Padding field to indicate if there were extra padding bytes at the end of the RTP packet. 1 bit Extension field indicated presence of an Extension header between standard header and payload data. This was application specific. 4bit CC field contained the number of CSRC identifiers that followed the fixed header. 1 bit Marker field was used at the application level and defined by a profile. If set, it meant that the current data had some relevance to the application. 7 bits of Payload Type field indicated the format of the payload and determined its interpretation by the application. This was again specified by an RTP profile in the RFC 3351 standards. In
this project, this was set to value 9, since the sip-communicator used G722 encoding format at 8000Hz [15]. 16 bit Sequence Number was incremented by one for every RTP data packet sent and was used by the receiver to detect packet loss and to restore packet sequence. The RTP did not specify any action on packet loss; it was left to the application to take appropriate action. 32 bit Timestamp field was used to enable the receiver to play back the received samples at appropriate intervals. 32 bit SSRC Synchronization source identifier uniquely identified the source of a stream. CSRC, Contributing source ID’s enumerated contributing sources of a particular stream which was generated from multiple sources. Extension header was optional [11][14][15]. The RTP payload followed this extension header.

The main functions provided by RTP hence included sequencing, payload identification (which was required to change the encoding of the media dynamically to adjust to the changing bandwidth availability), frame identification (which indicated the beginning and end of the audio/video frames using a frame marker bit), source identification and synchronization using time stamps.
CHAPTER 3

Implementation of VoIP

This project provided a detailed understanding of various steps involved in VoIP implementation using SIP standards. It explained how SIP protocol helped to establish sessions and terminate sessions, how the application transmitted session descriptors and how it allowed us to control the sessions established between caller and callee. The Real-time Time Protocol (RTP) standards were also analyzed in detail to understand how they worked along with Real-time Transport Control Protocol (RTCP) to provide successful data transmissions from source to destination.

3.1 Introduction

In this project, the SIP communicator application connected to the backend Iptel SIP server using the login accounts that were created for this communication. Once connected, users with accounts on SIP server could be added from the interface. After the end user accepted this add request, they could communicate with each other through this application and make voice/video calls. The real-time media transmission and routing were handled at the application layer of the SIP server. Once the call went through successfully, the user accounts could communicate with each other. The second phase of this project involved monitoring this call statistics and save them in a database implemented in Oracle. The data in this database was then used for calculating and displaying the usage statistics along with the billing information for each user account. Vendors and administrators could use this application to monitor the user accounts and...
their usage and could take advantage of this application to charge a premium for their users in return for their support and maintenance.

3.2 Requirement Specifications

The developed VoIP system was platform independent. It was designed to operate without any additional hardware equipment, except microphone, speaker and LAN, which were commonly found in most of the households. JAVA was the preferred programming language, because it not only provided sufficient methods and classes for the implementation of VoIP communication system in eclipse IDE, but also supported its integration with an Oracle database, that was created for the billing system implementation.

The first step in the system development lifecycle was to identify the need for this development. Usually this need came from the end user or customer and his application requirements. Nevertheless, this design was enforced by the following functional requirements, non-functional requirements, hardware requirements and software requirements. The merits, demerits and feasibility of implementation were discussed and analyzed before the project was implemented.

3.2.1 Functional Requirements

Proposed functional requirements for this project included –

- Connection establishment and voice transmission between the participants over an IP network. This constituted the basic VoIP communication.
- Portability- should be easily transferrable in the form of an executable.
- Call status- online, offline.
• Good and easy to use GUI.
• Creating a database with two tables: one for login (that stored email address/ID, passwords, first name, and last name) and the second table for transactions (that stored ID, from username, to username, call start, call end)
• Registration form access, for the new users to register and access all of their call statistics in the database.
• Storing and retrieving caller/callee, call start/end times and ID into/from the database.
• Using those call statistics, calculate the call duration and billing costs and make it available to the users of the system after they logged in.

3.2.2 Non-Functional Requirements

Proposed non-functional requirements for this project included –
• Security: The program should not harm user’s data, corrupt files or cause any viral activity in the event of program crash.
• Ease of use: a reasonably legible interface has to be provided. The user interface should not confuse the user with too many options.
• Portability: System should run on all JAVA enabled systems connected to LAN and should be platform independent.
• Efficiency: System should utilize the resources efficiently and give a better quality of output and user satisfaction.
• Reliability: The application must be reliable and it should not degrade the existing system performance and must not lead to the system crash.

3.2.3 Hardware Requirements

Hardware requirements for this project were –

• Pentium IV
• HARD DISK 150GB and above
• 2GB and above RAM
• Microphone
• Headset or Loud speaker
• Optimum LAN speed for smooth communication between the participants.

3.2.4 Software Requirements

Software requirements for this project were –

• Windows XP and above.
• JDK v1.6 including Java Run time Environment (JRE)
• Eclipse SDK v3.7.2 (Not required for the end user using the executable)
• Apache-ant v1.8.2 (drives the processes described in the build files as targets)
• Apache-Tomcat v7.0.29 (web server that powers the web application, the billing system in this project)
• SQL developer v3.1 (creates the database schema to store the data)
• Oracle DB express 11g (hosts and manages the database).
3.3 VoIP Implementation

SIP protocol used e-mail type addressing to initiate, control and terminate sessions. This project used user@iptel.org type of addressing to allow one user to connect and communicate with other user using sip-communicator application, based on SIP standards.

![Diagram](image_url)

Figure 3.3.1: General operation in VoIP communication [6]

When a person registered as a user, say jingpang, on the “iptel.org” server wanted to initiate a VoIP communication call to another user registered as, say karthikram, on the “iptel.org” server or any other SIP server on the internet, they had to go through the
following list of sequences before being able to connect and communicate. In step1, the
user agent called jing pang@iptel.org would initiate the call by sending INVITE message
to the iptel server through the sip-communicator application running on the personal
computer system. In the step2, the iptel server would try to find the user karthikram by
forwarding the INVITE request containing the destination information, to its location
server, which would determine the location to route the incoming INVITE request.
Assuming the user karthikram was not connected to the iptel server, the location server
would determine the local host location for that user in step3 and would route the
INVITE request to his home PC in step4. If the users home PC was connected to the
internet online and running the sip-communicator daemon process, he would be notified
in step5 to which his system would send back a 200 OK in step6, to indicate its
willingness to establish a session. This would be relayed to the user jing pang at iptel.org
in step7, to which that system would respond with an ACK message in step8 after
agreeing to the session descriptors for the established session. Once the session was
established the real-time media communication would start from step9 onwards using the
RTP, RTCP and underlying UDP protocols over the network. Once the communication
completed, either users could chose to end the communication by sending the BYE
message and the other system would acknowledge it and close the session using SIP
protocol. As mentioned previously the real-time voice was encoded by the sip-
communicator application that ran on the end user systems, as per the G722 encoding
format specified by the RTP header payload type field. This encoded data was
encapsulated in the payload within the RTP packet at the application layer and sent to the iptel server, which routed them to the correct destination established in the session.

While Figure 3.3.1 explained the protocol level implementation of VoIP system, Figure 3.3.2 below shows the corresponding GUI that was implemented by jitsi to place VoIP calls using SIP protocol, view call statistics and other details. This implementation adapted from the open source sip-communicator project looked similar to a chat window from which calls could be placed to another user who was running the same application on a remote system over the internet.

![Figure 3.3.2 GUI window for placing VoIP calls](image.png)

### 3.4 Common SIP Methods

As mentioned earlier, this implementation mainly used SIP protocol for VoIP communication. SIP requests or methods acted as verbs in the protocol, since they
requested a specific action to be taken by another user agent or server. The INVITE, REGISTER, BYE, ACK, CANCEL, and OPTIONS methods were the original six methods in SIP. The REFER, SUBSCRIBE, NOTIFY, MESSAGE, UPDATE, INFO, and PRACK methods were described in separate RFCs. The following SIP methods were generally involved in most of the VoIP communication. As seen from their description, they were clearly used for signaling purposes. Detailed information on these method can be found in RFC 3261 standards [6].

- **INVITE** - This method initiates the call
- **ACK** - Confirms the final response
- **BYE** - Terminates the call
- **CANCEL** - Cancels searching and ringing to the destination callee
- **OPTIONS** - Features supported by the other side client
- **REGISTER** - Register with location service
- **INFO** - Supplies mid call information
- **COMET** - Verifies Pre conditions are met
- **PRACK** - Provides provisional acknowledgement before proceeding with the call
- **SUBSCRIBE** - Subscribe to events
- **NOTIFY** - Notify user availability or non-availability
- **REFER** - Ask recipient to issue SIP request (call transfer)
3.5 Comparison of Protocols

This project mainly used SIP and RTP protocols which were the recommended standards for VoIP communication. There were a host of other protocols that also supported the audio encoding/decoding. Some of these included H.323, an ITU-T umbrella standard; Media Gateway Control Protocol (MGCP) from IETF; Media Gateway Control (MEGACO), a joint protocol by IETF and ITU-T; and proprietary protocols such as Cisco’s Skinny Client Control Protocol (SCCP). However not all of them were widely accepted. Table 3.5.1 showed the basic differences between these protocols.

<table>
<thead>
<tr>
<th>Features</th>
<th>SIP</th>
<th>H.323</th>
<th>MGCP</th>
<th>MEGACO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>Peer2Peer</td>
<td>Peer2Peer</td>
<td>Master/Slave</td>
<td>Master/Slave</td>
</tr>
<tr>
<td>Standardization</td>
<td>IETF</td>
<td>ITU-T</td>
<td>IETF</td>
<td>ITU-T and IETF</td>
</tr>
<tr>
<td>Network Usage</td>
<td>Intranet and Internet</td>
<td>Intranet and Internet</td>
<td>Intranet</td>
<td>Intranet</td>
</tr>
<tr>
<td>Scalability</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Applications</td>
<td>Voice, Video, Data</td>
<td>Voice, Video, Data</td>
<td>Voice</td>
<td>Voice, Video</td>
</tr>
</tbody>
</table>

Table 3.5.1: SIP Protocol Comparison

By looking at the above table, we can see that the SIP and the H.323 protocol almost provide similar services. These protocols are hence crucial in IP Telephony. These protocols illustrated two different approaches to the same signaling challenge. However, some architectural differences made them better suited for one application over the other. The main ones were:
• H.323 defined too many elements in order to provide rich multimedia experience and support for various protocols, while SIP had fewer elements, that made it lighter than the H.323 [13].

• H.323 encoded its messages using binary format while SIP encoded its messages using text based format. Hence, H.323 needed special code generators to decode its messages, whereas SIP decoding was quite simple. This also meant that, debugging SIP message was more straightforward and simple [13].

• IP Telephony demanded a large number of different functions to keep it up to date. Though these functions changed over time, it was necessary to have them as separate modules so that they could be easily updated. SIP supported modularity much better than H.323.

• H.323 was designed for use on a single Local Area Network (LAN). Hence H.323 faced big scalability issues when it was ported to larger networks.

• Crucially, H.323 did not provide a good quality of service as that of SIP.

• There were some tradeoffs between SIP and H.323 and one was better suited than the other was, depending on the application. SIP was sufficient for Client/Server applications and basic signaling functionality. However, in order to create a network with complete audio, video and advanced data control features, SIP would fall short of expectations.
CHAPTER 4

Billing System Design

The idea for designing a billing system for this VoIP implementation was inspired from a journal [3] by Wei-Zu Yang and others. Although their VoIP architecture used the same SIP Protocols and User Agent (UA) and proxy servers to establish communication, their VoIP systems as a whole was different than our setup.

4.1 Introduction

The experiment [3] done by Wei-Zu Yang and his team was based on communication from web phone software to a VoIP phone. In my project, we used VoIP software (built from open source SIP-Communicator by jitsi) on two computer systems to establish connection and communicate. Their experiments were based on a pre-paid calling system unlike my design, which offered credit, and billed the users later for using the VoIP services. They used SIP Express Router (SER) as their main call server. They also used Free Radius server integrated with Back-to-Back User Agent, to provide billing and call authentication services. Since they had access to the BSD of the call server, they were able to integrate the database functionality into the server as well. In my project, a single SIP Iptel server handled both call authentication and media management for the VoIP calls. Unlike their implementation, which used mySQL database, my project used Oracle database. The reason behind this was that the Oracle database allowed easy management of billing for the system administrators at the enterprise level and offered wider support and reliability. The open source sip-communicator project was developed on an OSGI framework.
4.2 Open Service Gateway Interface (OSGi)

OSGI framework emphasized modularity in programming, so that different applications developed using this framework, could re-use the components. It allowed the applications to load these components dynamically during run time and made those services available to all other modules within the framework while hiding the implementations. The performance of an application developed on an OSGI framework, was better than a standalone Java Virtual Machine (JVM) or Java Enterprise Edition (JEE) because the bundles were loaded manually as and when required rather than specifying everything in the class path. The main components of the OSGi framework included:

- Bundles - that allowed service registration, importing and exporting of other services as well.
- Services - connected the bundles in a dynamic way. They used publish-find-bind model for the Java objects.
- Life Cycle – specified when to install a bundle, start or stop a bundle and when to uninstall them.
- Modules – defined how a bundle could import and export code.
- Security – defined how the services and methods could be accessed by other components in the framework.
- Execution Environment – defined what methods and classes were available.
4.3 SIP Communicator Design

The open source Sip-Communicator nightly build v3507 project formed the starting point for the implementation of billing system for VoIP communication using SIP protocol. The following modules of the SIP-Communicator project played a key role in the establishment and control of the VoIP communication. The packages in the source code that started with the wordings “net.java.sip.communicator.impl” contained the actual implementation for the classes and the interfaces. This would be made available to the packages called “net.java.sip.communicator.service” that allowed different implementations to use these interfaces, without allowing them access to the actual source code. When the sip-communicator project was run in the eclipse IDE, the “SIPCommunicator” class of “net.java.sip.communicator.launcher” package invoked the method “launchargumenthandler”. This in turn loaded the felix properties file. The felix properties file in turn defined all the various Java Archive (JAR) files that needed to be loaded to the OSGi framework, one of them being “ui-service.jar” which actually provided the User Interface services in the sip-communicator project. The “SIPCommunicator” main class of the project also checked if there were multiple instances of the sip-communicator running and verified other environmental variables as well, before it launched the jitsi login pop up window by calling the “PopupDialogImpl” class defined in the “net.java.sip.communicator.impl.gui” package. Once the pop up window was brought up, it waited for the arguments from the user. Since the users were pre-registered with the iptel.org server, passing those login arguments through the GUI window instantiated the “SIPAccountID” class defined in the
“net.java.sip.communicator.impl.protocol.sip” package, which then processed the input parameters. The “AddressResolverImpl” class after resolving the address, helped to establish the sip connection with the backend SIP iptel server over the internet. Once the preliminary SIP communication completed, the user were seen online, in the jitsi call window. Users were able to add others and perform a host of other activities. When the user decided to make a call to other contact list user, the session got set up by the “OperationSetBasicTelephonySipImpl” class which in turn implemented all the call management logic and provided basic telephony support. Once the end user accepted the connection, the RTP protocol kicked in. RTP specified the encoding/decoding format of the transmission media and other details of communication in its packet, as indicated in the chapter 2. The classes defined in “net.java.sip.communicator.impl.neomedia”, “net.java.sip.communicator.impl.neomedia.codec.audio.g722” and “net.java.sip.communicator.impl.neomedia.codec.audio.alaw” packages handled this real-time communication. Once the SIP Iptel server received this media, it took care of routing them to the destination address in real time. At the destination, the end user system decoded this data to reproduce the original signal.

4.4 Billing System Design

The second phase of the project involved creating new packages that allowed us to connect from SIP Communicator java application implemented in the OSGi framework, to the Oracle database schema created using SQL developer. The “create_jitsi_schema.sql” script run from the SQL developer or command prompt console(requiring administrative rights), created the two tables login and transaction in
the oracle database. It performed this action after checking for existing tables/views. The major components of the designed billing system are as shown in Figure 4.1.1. It consisted of

- SIP-Communicator Application
- Oracle Database
- Tomcat webserver
- Client HTML browser

Once the call communication was completed using the modified SIP-Communicator applications running on the end user systems, it inserted the caller statistics to the Oracle database. This implementation was achieved by modifying the SIP-Communicator class called “CallHistoryServiceImpl” and by creating and adding a new class called “CallRecordToDatabase”. The Tomcat web server had two fold functions. It not only accessed the database and parsed through the data of the schema after it received the request from the HTML browser, but also displayed them as a response on the webpage of the local host system after the user logged in. This was achieved by the creation of new sub project called jistsi-app that had sub classes called Billing, Login, Logout and Signup in its package “com.jitsi.app”. Connection to the Oracle database and other accesses were implemented in the “com.jitsi.DOA” package.

Data Access Objects (DAO) handled all the interactions with the data source within the database. In this project, the DAO performed the following functions

- Read the data from the container using “Getter” method to get the value of variables and matched it with the database values.
• If the data from the user matched the database data, it retrieved the table contents from the database and saved it to the bean using the “Setter” method from which it was made available to the user through the Java serve-let pages (JSP)

The tomcat web server, which communicated between the client browser and the database, deployed these packages developed for the billing system. The HTML client browser on the local system was used to display the billing statistics. The Oracle DB maintained and managed all the call records.

Figure 4.1.1 Components of the billing system

After working through all the major quirks and debugging outlined in the next chapter, the billing system application was finally up and running. It displayed the call
statistics according to the scope of the project as seen in the Figure 4.1.2. I also added CSS Styles to the webpage to make it more presentable.

Figure 4.1.2 Billing system page showing VoIP call statistics

As evident from this chapter, this project required multidisciplinary understanding of various technologies involved. It included audio signal processing, various communication protocols, networking, database management, Java programming and web design.

The key aspect for successful communication was, using the right ports and addressing for different protocols involved. The table 4.1 indicates the lists of ports that this Billing system used. It was interesting to note that the default port of the tomcat web
server was being used by some other application. Because of this, there was a conflict in accessing the billing webpage as explained in the next chapter. The next chapter Design Testing and Analysis also explains the details of the call statistics displayed in Figure 4.1.2 and how they were calculated.

<table>
<thead>
<tr>
<th></th>
<th>Default port</th>
<th>Port actually used</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SIP Protocol</strong></td>
<td>5060/5061</td>
<td>5061</td>
</tr>
<tr>
<td><strong>Tomcat</strong></td>
<td>8080</td>
<td>9090</td>
</tr>
<tr>
<td><strong>Oracle Database</strong></td>
<td>1521</td>
<td>1521</td>
</tr>
<tr>
<td><strong>listening</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>HTTP</strong></td>
<td>443</td>
<td>443</td>
</tr>
</tbody>
</table>

Table 4.1 Main Ports used in VoIP communication.
CHAPTER 5
DESIGN TESTING AND ANALYSIS

5.1 Introduction

The project design was thoroughly tested throughout all the phases of its development. The initial testing stages involved running the example codes and creating hello world programs to test the set up environment and proper functioning of the installed software’s.

The testing of billing system was performed at various levels. In order to test the successful creation of tables, I executed various scripts and commands some of which are highlighted in this chapter. This helped me to understand when a table was created, when a table was not created, what fields were missing, what fields were updated, what fields were not updated and thus guided me through the debug process.

5.2 Commands Used in Testing

The following commands were used to analyze and test the successful creation and working of the tables in the database and when they were being populated.

- Describe login – would show all the login table fields that were defined in the code and if it returned a result as seen below, then the tables were created successfully. Initially when the tables were not created, I faced PL/SQL errors and learned that the views/tables had to be created before it could be accessed or read from. I had to debug the code several times before I could successfully create them.
As seen in Figure 5.1.2 below, the login table was empty, before any users had signed up from the designed billing webpage. This verified the initial stage of the webpage design. Later when the users signed up, this data was populated into the database through the web server. The Figure 5.1.3 shows a populated login table with different test accounts, that were created to verify the working of this webpage.
• Select * from login – would return an empty table of the login fields pointing to null, if the billing application had no registered users. This was seen in the Figure 5.1.2. When the user tried to create a new login using the signup page in the billing system, he had to enter his email address (which had to be the same one that was used for making the VoIP calls), password, first name, last name. This was then populated into the database as seen below.

![Filled up login table after signup](image)

Figure 5.1.3 Filled up login table after signup

• Describe transaction – would show all the transaction table fields that were defined in the code and if it returned the result as seen below then the tables were created successfully. Initially when the tables were not created, the interface gave “HTTP Status 500 - An exception occurred processing JSP page /Billing.jsp”. I had to correct and debug the code
before I could successfully access the billing webpage that contained all
the user transactions.

![Figure 5.1.4 Output of describe transaction script](image)

<table>
<thead>
<tr>
<th>Name</th>
<th>Null</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>NOT NULL</td>
<td>NUMBER(11)</td>
</tr>
<tr>
<td>FROMUSERNAME</td>
<td>VARCHAR2(500)</td>
<td></td>
</tr>
<tr>
<td>TOUSERNAME</td>
<td>VARCHAR2(500)</td>
<td></td>
</tr>
<tr>
<td>CALLSTART</td>
<td>TIMESTAMP(6)</td>
<td></td>
</tr>
<tr>
<td>CALLEND</td>
<td>TIMESTAMP(6)</td>
<td></td>
</tr>
</tbody>
</table>

- Select * from transaction – would return an empty transaction table with
  all fields pointing to null if there were no calls placed but would return a
  populated table as seen in Figure 5.1.5 if the calls were placed successfully
  and if the table was created successfully before the web server tried to
  access it. As seen in the Figure 5.1.5, it showed the fromusername
  (indicating the origin of call), tousername (indicating the destination of
  call), callstart (indicating call start time), callend (indicating call end
  time). Using these statistics the total call duration and billing amount was
  calculated.
5.3 Debug

To ensure that Tomcat web application server was running properly and performed its functions as expected, I had to monitor the console for any error logs. Any errors seen here would not bring up the billing application properly and required some debug. Figure 5.1.6 shows some of the errors that I faced during coding of the billing system application. As seen from the logs the socket bind failed when the tomcat server tried to establish a connection to the local host port and it failed to initialize the connector. This was because the tomcat server was configured by default, to use the 8080 port on the system but this port was already being used by the some other application. In order to discover what application used this port, I used the following command to get the output as shown in Figure 5.1.7 from which I got the PID of the application running on that port. This was then traced back to the corresponding application from the task managers processes.

C:\Windows\system32>netstat -a -b -o > test.txt
C:\Windows\system32>notepad test.txt
Figure 5.1.7 PID that was using port 8080
5.4 Testing the Billing Application

Figure 5.1.8 shown below indicated the billing history being empty initially before any calls were placed using the login account for karthik ram.

![Billing History](image)

Figure 5.1.8 Empty billing history

Figure 5.1.9 shows the updated billing webpage with the caller/callee information and other usage statistics. As seen in the figure, the first call was placed from karthik ram user to jing pang@iptel.org. The call started at 1:22:26 AM and ended at 1:23:02 AM. The total duration was calculated by subtracting the start and end times. As seen in the figure, this call lasted for 36 seconds. The billing rate defined in the project was 10cents per minute or .0017cents per second. Using these values, the code calculated the billing amount to be $0.0612 or 6.12 cents. This confirmed the working of the billing system.
design for VoIP as per the project requirements. These experiments were run on Windows XP and Windows 7 with 32 and 64 bit operating systems.

Figure 5.1.9 Updated billing history
CHAPTER 6
CONCLUSION AND FUTURE WORK

As mentioned earlier, this project requires multidisciplinary understanding of various technologies involved. It includes audio signal processing, various communication protocols in VoIP, networking, database management, Java programming and web design. Understanding all of these technologies and integrating them together to make it work was a challenge in itself, apart from the scope of the project. I learned a lot during every phase of this project starting from design requirements, analyzing what was preferred and why, to the development and testing the implementation. This project will also run successfully on a Windows XP/Windows7 system, provided it has all the adequate hardware, software support. There were a few instances in which there were glitches in compatibility between 32 bit and 64 bit systems.

6.1 Issues faced

The Java based sip-communicator source code used in this project is contained in Open Services Gateway Initiative (OSGI) framework. Generally, Java uses a parameter called “classpath” (defined through an environment variable) to let the compiler of the program know where to find all the user defined classes and packages. But the developers who came up with OSGI framework, didn't want everything in classpath to be loaded for the application to work. Therefore, they made a specification to select every single bundle you want to use. The Oracle database designed for this project did not support OSGI. It was a nightmare getting Oracle drivers to recognize inside OSGI framework.
Therefore, I had to rebuild the oracle driver manually to make it work with this VoIP implementation.

6.1.1 Quality of Service

Quality of service, and loss issues are the major factors that are holding the growth of VoIP back. There is some echoing observed when the end users making a VoIP call are in the same room. Since the idea behind VoIP is that the data is sent in packets, there are problems such as packet delay and packet loss to contend with. Loss recovery techniques and play out adaptation are crucial in keeping the loss low and the quality of service high. There has been a lot of research into these two techniques, and with the advancements of these fields it is expected that VoIP will be able to compete with the high standard of quality of service we have come to expect from the tradition telephone network [7].

6.1.2 Security

There can be a variety of security issues that one may have to deal with when using VoIP since the media/voice that is transmitted goes through the internet which happens to be a place where notorious events may occur thus compromising the information that is being sent. In this project, I faced a ZRTP error when I tried to instantiate 2 instances of Jisti running on the same computer system giving me a warning of a potential risk of data being compromised.

VoIP calls will be subject to various types of security threats, depending on how the information travels over the network before it reaches its destination. Some of them include [16]:
• Spooﬁng From in REGISTER: call redirection
• Spooﬁng From in INVITE: bypass call ﬁltering
• Snoop­ing media packets
• Billing confusion (identiﬁer munging)
• Denial-of-service attacks
• SIP INVITE takes a basic authentication approach and sends the packet as plain text thus sharing sensitive information such as passwords with third parties over the internet.

6.2 Conclusion

The billing system design and implementation in this project, is a robust web application that is capable of managing real time data. Using Session Initial Protocol (SIP) to establish a line of communication between users, the application provides a platform for social interaction. As the world moves to instant communication and the age where more people are connected to the internet, a need for secure, reliable and robust forms of communication emerges. This project lays a path for implementing such a system that works in real-time data storage and retrieval.

The billing system components are constituted of two parts namely, a desktop application that runs Java byte code contained in the Java Virtual Machine (JVM) implementing the Open Services Gateway initiative (OSGi) framework on the Microsoft Windows platform and a web application that runs inside of a Java Enterprise Edition (JEE) container i.e. Apache Tomcat. The data that drives the billing system is stored
using an Oracle Database Management System (DBMS) Express edition. The data exchange between the desktop sip-communicator application and Oracle database as well as the data exchange between the billing system web application and Oracle database is enabled using the Oracle Java Database Connectivity (OJDBC) library. All components are derived from free and open source initiatives making the development and operating costs minimal.

The real world applications of this system make it a user interactive and compelling software to use on a daily basis. Users can register to create their accounts in the web application. The same accounts can be used in the desktop application to enable Voice over Internet Protocol (VoIP) based communication through any open source SIP servers. The feature of displaying a user’s call history containing the callee information, call start, call end, call duration and the monetary charges for using the application help in making a compelling product. This seamless integration of sharing data between the two applications forms the core of the system. The users can rely upon the stability and reliability that the software offers. The billing system fulfills the needs of a user looking to use a way to communicate with their peers at a reasonable cost. During the course of this project, considerable effort went into ensuring that the system was capable of handling calls and managing the data about those calls between users. The data is stored in a central Oracle repository allowing access from both the desktop application and the web application.

The programming concepts, tools and pattern used in the design and implementation of this project are the same as that used in the computer software
industry. N-tier web architecture, Object Oriented Programming (OOP), Transaction Management of database actions and Interaction Design play a vital role in the development the billing system. Managing a large amount of data also means increased reliance on performance tuning of database heavy actions as well leveraging the multi-threading that Java offers.

The billing system is a robust, reliable and scalable system that accomplishes the goal of providing a platform for real world usage by users connected to the internet. With more developments in online connectivity, it is important for users to be able to access their call data and view their bills in a fast non-intrusive way. This project helps meet that goal of delivering a platform for real time data access to users.

6.3 Future Work

Several factors will influence future developments in VoIP products and services. Currently, the most promising areas for VoIP are corporate intranets and commercial extranets. Their IP–based infrastructures enable operators to control who can and cannot use the network and at what cost.

Providing Facsimile over Internet protocol (FAXoIP) services to the public is another area of major development. Another influential element in the ongoing Internet-telephony evolution is the VoIP gateway. As these gateways evolve from PC–based platforms to robust embedded systems, each will be able to handle hundreds of simultaneous calls. The economics of placing all traffic (data, voice, and video) over an IP–based network will pull companies in this direction, simply because IP will act as a
unifying agent, regardless of the underlying architecture (like leased lines, frame relay, or ATM) of an organization's network.

The scope for adding enhancements and new features to the billing system are plenty. As the world moves towards cloud data storage and retrieval, it becomes an important part to keep evolving the lifecycle of the software being developed. The large companies in the business add more features to ensure their users have a great experience. Evolving of the software with new features and improvements in usability will go a long way in creating a lasting product.

Data storage costs have gone down considerably due to the introduction of cloud computing. With the evolution of Software as a Service (SaaS) and Platform as a Service (PaaS), the billing system can be improved further. The Oracle database management system used could potentially be replaced by a cloud based data management system like NoSQL can be hosted in numerous cloud based servers. This would involve minimal change in the application setup other than replacing the JDBC uniform resource location (URL) with the appropriate cloud server running a database instance. There can be considerable savings in buying physical hardware for maintaining a database server, taking consistent backups of data and storage capacity management when cloud hosting is considered.

The web application designed and implemented is currently local to the server that is hosting the JEE container Apache Tomcat. This can potentially be moved to an Infrastructure as a Service (IaaS) such as Amazon EC2 or Microsoft Azure. Savings in server infrastructure as well as performance of the application since the cloud services
harness the power of data centers running multiple systems built using multiple CPU cores.

Another aspect that can be improved on is the features available in the web application. Currently, only the calling history of a user along with the billing charges are shown to a user. Trends and analysis of the calls made can help the user understand how the system is being used. Server side tracking or enabling of analytics will also help the developer understand how users are using the system. Displaying the call history by month, comparing month-by-month or year-by-year usage, amount of money billed over time are few of the uses of collecting large amounts of data over time.

The authentication of the billing system can be enabled to use third party authentication and authorization. Systems such a Facebook Login, Google+ Login or Microsoft Live ID will remove the cost of maintain sensitive data of user login and passwords in the billing system itself. It could help differentiate the usage of the billing system to being focused on the billing aspect of the application rather than the added responsibility of maintaining user authentication and authorization.

The web application in itself could offer services to end users for claiming discounts or free minutes upon certain usage. Heavy users of the system could benefit from such lucrative offers that will result in customer goodwill as well as retention of loyalty in the user base. Linking the billing system to coupon services like Google Offers or Facebook Offers to allow coupon codes for claiming rebates and free minutes will ensure a larger visibility to the system as a whole.
Performance tuning for handling video calls and other forms of communication such as screen sharing could be included in the roadmap. This would ensure the features of the application continue to grow as the usage increases over time. Data collection of reasons for call failures as well as displaying more statistics as part of the billing history could help users know more on their usage of the application. On the whole, accuracy can be improved consistently to ensure there is wider adoption of the application among users.
REFERENCES:


