MODEL TO ESTIMATE THE NUMBER OF SESSIONS HANDLED BY CONFERENCE SERVER IN SESSION INITIATION PROTOCOL CONFERENCING SOLUTION WITH LESS DELAY AND THE OPTIMUM PROCESSING CAPACITY

A Thesis by

Natarajan Venugopalan

Bachelor of Engineering, Anna University, India, 2005

Submitted to the Department of Electrical Engineering and Computer Science and the faculty of the graduate school of Wichita State University in partial fulfillment of the requirements for the degree of Master of Science

July 2009
MODEL TO ESTIMATE THE NUMBER OF SESSIONS HANDLED BY CONFERENCE SERVER IN SESSION INITIATION PROTOCOL CONFERENCING SOLUTION WITH LESS DELAY AND THE OPTIMUM PROCESSING CAPACITY

The following faculty members have examined the final copy of this thesis for form and content, and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

_________________________________
John Watkins, Committee Chair

_________________________________
Kameshwara Rao Namuduri, Committee Member

_________________________________
Edwin Sawan, Committee Member

_________________________________
Ravi Pendse, Committee Member

_________________________________
Krishna Krishnan, Committee Member
DEDICATION

To my parents, my brother and my dear friends
ACKNOWLEDGEMENTS

I would like to thank my advisors, Dr. Kamesh Namuduri, Dr. Edwin Sawan, and Dr. John Watkins for their guidance and support in my thesis work at Wichita State University. I would also like to thank Dr. Ravi Pendse and Dr. Krishna Krishnan for their time spent in reviewing this thesis work.

I appreciate all my friends for their continued support and providing me with valuable suggestions during my course of stay in Wichita. I am grateful to all my roommates for supporting me morally through the years I have known them.

Finally, I am very grateful to my family members for their love and support.
ABSTRACT

The recent introduction of conferencing products such as telepresence has led to immense growth in the number of voice conferencing calls on the internet protocol (IP) network. Currently, voice conferencing users are provided with many options, such as free session initiation protocol (SIP) soft phone and voice over internet protocol (VOIP) calls at a lower rate. As technology and social networking sites continue to grow, there is a good possibility that these sites could be integrated with VOIP conferencing solution, which would lead to enormous growth in IP voice traffic. The recent addition of core routers by network software companies also indicates an increased prediction for real-time and multimedia traffic. With such a prediction in the growth of voice traffic, it becomes essential to estimate the delay as well as voice quality analytically.

The SIP conferencing solution includes a key centralized entity called a conference server, the role of which is not limited to maintain the media sessions between participants and forwarding traffic from the active speaking user to other participants. Considering the finite end-to-end delay of 150msec for VOIP traffic, the conference server should handle the job efficiently so that more delay is not introduced to voice traffic.

Since voice traffic must pass through several middle agents such as Session Border Controllers and proxy servers for specific purposes, the delay increases with these centralized devices in addition to that introduced by devices such as routers, firewalls, and switches. Therefore, the delay management of voice traffic in the conference server becomes prominent as the voice traffic becomes futile beyond the finite end-to-end delay of 150msec.

The delay of voice traffic in the SIP conference call scenario increases due to many factors. The factors that were influenced by the conference server are the application processing
capacity of the server and traffic intensity. In this thesis, a model has been proposed to estimate the number of sessions that the conference server can handle at a specific processing capacity with less delay. This model was simulated using Matlab, and the observed results verify the proposed model, with graphs showing the necessary optimum processing capacity of the conference server.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>INTRODUCTION</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>LITERATURE SURVEY</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>2.1 Overview of VOIP Protocols</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>2.1.1 Session Initiation Protocol</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>2.1.2 Session Description Protocol</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>2.1.3 Real-Time Transport Protocol</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>2.1.4 Real-Time Streaming Protocol</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>2.2 Framework for SIP Conferencing</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>2.3 Related Work</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>MATHEMATICAL MODEL</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>3.1 Characteristics of Conversation Speech</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>3.1.1 Speech Temporal Parameters</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>3.2 IP Traffic Model</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>3.3 State Transition Diagram and Transition Matrix</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>3.4 Calculation of Delay</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>3.5 Estimating the Number of Sessions</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>SIMULATION AND RESULTS</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>4.1 Simulation Logic</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>4.2 Simulation Results</td>
<td>26</td>
</tr>
<tr>
<td>5</td>
<td>CONCLUSIONS AND FUTURE WORK</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>REFERENCES</td>
<td>34</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Successful SIP Session Establishment</td>
<td>5</td>
</tr>
<tr>
<td>2.</td>
<td>SIP INVITE Method</td>
<td>6</td>
</tr>
<tr>
<td>3.</td>
<td>Session Description Protocol Parameters</td>
<td>7</td>
</tr>
<tr>
<td>4.</td>
<td>SIP Conferencing Architecture</td>
<td>9</td>
</tr>
<tr>
<td>5.</td>
<td>Centralized Conference Server Architecture</td>
<td>11</td>
</tr>
<tr>
<td>6.</td>
<td>State Diagram Representing Number of Sessions</td>
<td>19</td>
</tr>
<tr>
<td>7.</td>
<td>Two State Markov Chain</td>
<td>21</td>
</tr>
<tr>
<td>8.</td>
<td>Delay Graph in the Unicast at 10kbps Processing Capacity</td>
<td>27</td>
</tr>
<tr>
<td>9.</td>
<td>Delay Graph in the Multicast Scenario at 10kbps Processing Capacity</td>
<td>28</td>
</tr>
<tr>
<td>10.</td>
<td>Comparison of Delays between the Unicast and Multicast Scenarios at 10kbps</td>
<td>28</td>
</tr>
<tr>
<td>11.</td>
<td>Comparison of Delays in the Unicast Scenario</td>
<td>29</td>
</tr>
<tr>
<td>12.</td>
<td>Comparison of Delays in the Multicast Scenario</td>
<td>30</td>
</tr>
<tr>
<td>Acronym</td>
<td>Definition</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>IDI</td>
<td>Index of Dispersion for Intervals</td>
<td></td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
<td></td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
<td></td>
</tr>
<tr>
<td>ITU</td>
<td>International Telecommunication Union</td>
<td></td>
</tr>
<tr>
<td>LRD</td>
<td>Long-Range Dependence</td>
<td></td>
</tr>
<tr>
<td>MCU</td>
<td>Multipoint Control Units</td>
<td></td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Squared Error</td>
<td></td>
</tr>
<tr>
<td>NAT</td>
<td>Network Address Translation</td>
<td></td>
</tr>
<tr>
<td>QOS</td>
<td>Quality of Service</td>
<td></td>
</tr>
<tr>
<td>RFC</td>
<td>Request for Comment</td>
<td></td>
</tr>
<tr>
<td>RTCP</td>
<td>Real-Time Control Protocol</td>
<td></td>
</tr>
<tr>
<td>RTSP</td>
<td>Real-Time Streaming Protocol</td>
<td></td>
</tr>
<tr>
<td>RTP</td>
<td>Real-Time Transport Protocol</td>
<td></td>
</tr>
<tr>
<td>SDP</td>
<td>Session Description Protocol</td>
<td></td>
</tr>
<tr>
<td>SIP</td>
<td>Session Initiation Protocol</td>
<td></td>
</tr>
<tr>
<td>SRD</td>
<td>Short-Range Dependence</td>
<td></td>
</tr>
<tr>
<td>SSRC</td>
<td>Synchronization Source</td>
<td></td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
<td></td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
<td></td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
<td></td>
</tr>
<tr>
<td>VOIP</td>
<td>Voice over Internet Protocol</td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td>Meaning</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
<td></td>
</tr>
<tr>
<td>$\Pi$</td>
<td>Pi</td>
<td></td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>Summation</td>
<td></td>
</tr>
<tr>
<td>$\mu$</td>
<td>Micron</td>
<td></td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Lamda</td>
<td></td>
</tr>
<tr>
<td>$\rho$</td>
<td>Rho</td>
<td></td>
</tr>
<tr>
<td>$\delta$</td>
<td>Delta</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Alpha</td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

The inception of media servers, telepresence, and other voice over internet protocol (VOIP) products has augmented the number of VOIP conference calls on the internet protocol (IP) network. Since most corporate functions currently depend on voice conferencing, the necessity of having a VOIP conference call with high quality and less delay becomes imminent. Voice traffic on the IP network in the present scenario must traverse through a number of devices, such as Session Border Controller for network address translations (NAT) and proxy servers for determining the end user location. Therefore, the processing delay of traffic gets marginally increased on the way from source to destination due to such devices. The challenge is that media information in the VOIP call must be delivered to recipients within a specific time period of 150msec.

Voice traffic consists of two parts, namely signaling and media. Signaling takes care of call establishment, parameters negotiation, and call teardown, whereas media involves the actual user speech. This thesis particularly concentrates on the media portion of voice traffic. Media mixing is accomplished by the conference server in a centralized manner.

The challenge of transmitting media information in a conference call within a specific time delay depends on the performance of the conference server. The processing of information obtained in each user session of the call and then transmitting it to other user sessions is the important task performed by the conference server.

The conference server can be either a dedicated server or any other session initiation protocol (SIP) entity that is part of the conference call, and it handles the media mixing. Possible ways for media distribution are as follows:
a) Multi-unicast: Each client in this model has N-1 media streams with “N” representing number of users. Each client here distributes the media to every other client.

b) Multicast: In this model, all clients become part of an established multicast group, and traffic is distributed among this group.

c) Single source multicast: A conference server is considered in this model and it redistributes the media traffic from one client to every other client.

It is logical that by controlling the amount of traffic, depending on the application processing capacity of the conference server, the delay of voice traffic could be kept within a specific range. Therefore, estimating the number of sessions handled by the conference server becomes imperative for having an effective voice conference.

The objective of this thesis is to estimate the benchmark number of sessions handled by the conference server and its optimum application processing capacity margin. This was achieved by proposing a mathematical model that analyzes the conversational voice traffic and extends it to voice traffic on the IP network. The model involves studying speech characteristics and speech temporal parameters, followed by the establishment of Markov chain to determine number of active traffic sessions.

Finally, the Markov chain was reduced to two states and used to segregate the traffic of the server between underload and overload depending on the estimated time delay in processing the media traffic. This model can help in estimating the number of concurrent conference calls as well as number of sessions per conference call handled by the conference server.

This thesis is organized into five chapters. Chapter 2 provides a brief technical overview of the SIP conferencing solution and related works. In Chapter 3, the proposed
model is discussed and in Chapter 4 the simulation results using Matlab are explained. Chapter 5 contains the conclusions and future work.
CHAPTER 2
LITERATURE SURVEY

This chapter is divided into three sections. The first section discusses VOIP protocols, followed by a section that explains the framework of SIP conferencing. The final section provides a brief overview of related work.

2.1 Overview of VOIP Protocols

This section includes a technical overview and brief discussion of session initiation protocol (SIP), session description protocol (SDP), real-time transport protocol (RTP), and real-time streaming protocol (RTSP).

2.1.1 Session Initiation Protocol

SIP is one of the signaling protocols in VOIP technology and usually works on user datagram protocol (UDP) port 5060. SIP plays a role in identifying the SIP endpoints and establishing the session between them whenever required. The identification of SIP endpoints is carried out with the help of proxy servers, which in turn list the SIP endpoints and their contact information.

SIP is a text-based application layer protocol similar to hypertext transfer protocol. It initiates the session between the endpoints, modifies the session parameters, and finally terminates the session when the call is finished.

The internet engineering task force (IETF) has standardized the specifications of SIP. SIP works in tandem with other protocols to build a complete multimedia architecture. SIP works with SDP to describe the multimedia session information, RTP to transport the real-time data and quality of service (QOS) feedback, and RTSP to transport the streaming media.
SIP was designed as a layered protocol and works independently in each layer with only a loose coupling between stages. Syntax and encoding is the first and lowest layer of SIP. The second layer is the transport layer and defines how an SIP client and server should send as well as receive responses. The transaction layer becomes the important layer of SIP.

The transaction layer includes the request sent by the client and all responses received by the client associated with that particular request. It also handles application-layer retransmissions and timeouts. The transaction user is the next layer. All SIP entities except stateless proxies become a part of this layer.

Figure 1[15] explains the call session establishment between two users, namely Alice and Bob.

Alice sends an invite to Bob, followed by Bob sending an optional provisional 180-ringing response message back to Alice. Once caller attributes are verified, the final response, 200 OK was sent back to Alice from Bob. The RTP media session is then established between the two
users. Any user can end the call and in Figure 1, Bob ends the call by sending a BYE message to Alice and Alice responds back with response 200 OK.

The registration process binds the address-of-record uniform resource identifier (URI) of an SIP entity with one or more contact addresses for a particular domain. Thus, when the proxy server receives the request, it forwards the request to the SIP entity whose contact address that was registered to the address-of-record matches with request URI in the received request. A registrar server acts as the front end to the location service per domain, and proxy servers are responsible for routing the request within the domain.

The INVITE method [15] in Figure 2 shows an SIP message with all necessary headers in order to establish the call.

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9faced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276295220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 151
```

Figure 2: SIP INVITE Method

The “Invite” header is the request URI followed by “Via”, which identifies the location to which the request should be sent. The “Via” header has a branch parameter that identifies the transaction created by the request. The branch parameter must be unique. The “To” header indicates the final recipient of the request. The “Call-ID” header is a unique field to group together a series of messages, and it should be same for all requests and responses sent by the user agent in the call dialog. The “CSeq” header provides a way to
identify and order the transactions; has a sequence number followed by the name of the method. The “Contact” header represents the address at which the user agent would like to receive any subsequent request. The “Content-Type” header specifies that there is an SDP associated with the SIP message. The “Content-Length” header specifies the length of the SDP body.

2.1.2 Session Description Protocol

SDP provides a standard representation to convey media details, and to transport addresses and other session description metadata to the participants. A sample SDP body [13] is shown in Figure 3.

```
v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=
c=IN IP4 192.0.2.101
i=0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000.
```

Figure 3: Session Description Protocol Parameters

Session description protocol parameters are described here:

“v” represents the protocol version

“o” represents the originator and session identifier

“s” represents the session name

“c” represents the connection information

“t” represents the time when the session is active

“m” represents the media name and transport address

“a” represents the zero or more session attributes
2.1.3 Real-Time Transport Protocol

RTP provides transport services for real-time data such as interactive audio and video. It runs on top of UDP and thereby utilizes its multiplexing service, especially in conference calls. It supports the data transfer to multiple destinations using the multicast distribution method if it is supported by the underlying network. The RTP consists of two parts: the RTP to transport the real-time data and the real-time control protocol (RTCP) to monitor the quality of service feedback in an on-going session. Multiplexing is provided by the unique transport address for each RTP session. The Synchronization Source (SSRC) identifier is used in the RTP header to identify the source for a particular RTP session. It must be globally unique within an RTP session. The RTCP periodically generates control packets to all participants in the session and provides feedback on the quality of data distribution.

2.1.4 Real-Time Streaming Protocol

RTSP initiates and controls the media streams of continuous audio and video. These streams can be either single or several with time-synchronization. An RTSP session is not a transport control protocol (TCP) session, and an RTSP client may open and close many reliable transport connections to the server. RTSP uses UDP and is used particularly for live presentations to distribute information among all participants in an existing conference. An RTSP URL identifies each media stream, and the properties are prescribed by a presentation description.

2.2 Framework for SIP Conferencing

This section explains the framework of SIP conferencing call and the available models. Figure 4 [9] shows the architecture of SIP conferencing.
Figure 4: SIP Conferencing Architecture

Two important models are used by SIP for multi-party communications. In the fully distributed multi-party conferencing model, each client maintains a signaling relationship with every other client. The other model is known as a tightly coupled conference in which a centralized point of control is utilized, and each client is connected to this central point. In this model, the central point performs the function of media distribution and media mixing. The central point can also perform media conversions depending upon the protocol support.

A single-user agent is referred to as a “Conference server” which in turn maintains a dialog with every other participant. It plays the role of centralized manager in a conference call. The conference server is identified by conference URI and maintains a signaling relationship with every other user. The important job role of the conference server is to forward the information to the mixer, which ensures that each participant receives the media. The mixer combines the set of media streams that it receives and redistributes it to the each participant in the conference.

A conference can be created in many ways. It is most important to create a conference server and its associated conference policy. The conference server is identified by
a conference URI. Each user joins the conference by sending an INVITE to the conference URI, and conference server accepts or rejects the invitation depending upon the conference policy. A user who is unaware of the conference URI but knows about the dialog can join the conference by using the “Join” header. A user can also use the “REFER” request to encourage other participants to join the conference. The “REFER” request will have the conference URI.

The participant exits the conference by sending a BYE request. The conference server can also send a BYE request and remove the participant. Each conference call consists of a particular set of media streams, which are managed by the conference server. Rejection of a stream by the participant implies that the particular participant will not receive the stream information but the stream will still be a part of the conference call.

The physical instantiations of the conference model can be achieved in two ways. Either, the conference server could be a dedicated server as a single box solution performing all essential job functionalities required for SIP conferencing, such as establishing the policies for the conference and centralized media distribution or the endpoint acts as a conference server when more than two users join the conference call; it also assumes the responsibility of media mixing.

Figure 5 [9] shows the architecture for centralized conference server. The conference policy, focus, mixer, and conference notifications are the unique logical functionalities performed by the conference server. It maintains the signaling, and media session with each of the participant.
The conference setup can use distributed media mixing or cascaded media mixing. In distributed media mixing, each participant becomes the mixer. The third party call control mechanism is used to transport the media information to the mixers. In the case of cascaded mixing, the media mixing is implemented by one or more mixers, and each participant is connected to one or more mixers. The conference server in this case uses some kind of control protocol to connect the mixers.

On the implementation side, a conference should include the “isfocus” feature in the contact header field, unless the conference server wishes to hide the fact that it is also a focus.

2.3 Related Work

Related work to this thesis focuses on analyzing the aggregate VOIP traffic. A model for conversation based VOIP traffic was proposed [7] to estimate the voice quality by observing the traffic characteristics between two users in a VOIP session. The metrics
analyzed were end-to-end delay, jitter, and loss rate. The model was realistic, as it was based on conversation rather than the simulated environment. G711 and G729 are some of the codecs considered along with the voice activity detection technique. Due to the consideration of voice activity detection technique, the packet sending rate was directly proportional to the user’s talking period. This paper stresses the importance of user characteristics in a voice call. The model describes the user behavior in terms of four states:

State 1: User A is talking and User B is silent
State 2: User A is silent and User B is silent
State 3: User A is talking and User B is talking
State 4: User A is silent and User B is talking

It was observed [7] that the traffic from A to B had less loss rate compared to the traffic from B to A. VOIP metrics were estimated for different codecs in both directions. The maximum number of VOIP streams that could be supported in a specific network before actual deployment was not estimated, and thereby large-scale traffic of VOIP streams need not be deployed in multiple nodes in the network. Instead, the objective [7] was to monitor VOIP quality variation in a single VOIP stream for a particular network medium.

Hassan, Garcia, and Bockstal [6] proposed an aggregate traffic model for superposition of homogeneous and heterogeneous traffic under both light and heavy traffic intensities. An aggregate model plays an important role when the number of traffic sources is on the higher side. As a result, the model was accurate and efficient in estimating the intended characteristics of VOIP traffic. Sources of voice traffic were considered to exhibit on-off patterns, (i.e) there would be no traffic during the off period. This work discusses both homogeneous and heterogeneous traffic conditions. Traffic originated from the same voice
codec was considered homogeneous traffic, whereas traffic originated from different voice codecs was considered to be heterogeneous.

Packet inter arrival time was mathematically found for both homogeneous and heterogeneous traffic conditions. A superimposed model was compared with exponential approximation in terms of the number of packets in a queue. It was observed that there was a significant difference between the outputs of both models. According to the superimposed model, there were 40 packets in the queue, whereas in the exponential approximation case, there were only 3.5 packets. Also, it was observed that exponential approximation was efficient only under light traffic conditions.

The later part of the work by Hassan, Garcia, and Bockstal [6] explores the fractal characteristics of traffic by determining the self-similarity. The Hurst parameter and the index of dispersion of intervals (IDI) were used to identify the self-similarity. It was observed that the IDI of the aggregated VOIP model increased with an increase in the number of traffic sources, and as a result, the exponential approximation could not be accurate. Queue lengths were estimated by varying the number of traffic sources in the two models.

Toral, Torres, Hernandez, and Estrade [8] observed the quality of VOIP calls in terms of self-similar process with short-range dependence (SRD) and long-range dependence (LRD) characteristics. They found that when the Hurst parameter was lower than 0.5 (SRD), average packet loss and jitter maintained the stable values. When the Hurst parameter went above 0.5, the QOS levels had diminished. At a larger payload size, the average jitter was observed to be at a higher value. It was noted that the Poisson approximation would be
acceptable for IP networks only under special conditions. But the self-similar process is often used for modeling traffic on IP networks.

Self-similarity was verified [8] in the VOIP traffic by pumping calls into a H.323 protocol-based VOIP network. The quality of service in a VOIP environment was analyzed based on the Hurst parameter. Various methods, namely variance and a modified Allan variance, were used for the Hurst parameter estimation. Estimators were chosen based on the Mean Squared Error (MSE). Experimental results [8] showed that the VOIP traffic exhibits self-similar traffic characteristics. It was also observed that jitter traces representing LRD signify lower voice quality than the ones that presented SRD. Finally, it was found that average jitter becomes less for a smaller packet size. Correlation was also found between the peak values of average packet loss and jitter. It was shown [9] that the same self-similar process analysis of VOIP traffic could be extended to SIP-based VOIP architecture. H.323 and SIP network differ largely in terms of hardware requirements. H.323 requires gateways, gatekeeper, and MCU equipment, whereas SIP works with less hardware.
CHAPTER 3
MATHEMATICAL MODEL

This chapter discusses the proposed model to estimate the number of sessions that the conference server could handle so that media traffic reaches the intended recipients within the allowed finite delay as well as its optimum application processing capacity margin. The chapter is divided into five sections. The first section discusses speech characteristics and speech temporal parameters. The IP traffic model is proposed in the next section. The third section shows the state transition diagram and the transition matrix. Delay calculations are explained in the fourth section, and the final section describes an estimate of the number of sessions using the delay calculations.

3.1 Characteristics of Conversation Speech

The nature of voice traffic can be described in terms of speech activity, talkspurts, and user silence periods. Talkspurts are considered traffic bursts in this model.

The following notations are used to identify the speech temporal parameters:

\( T_h \Rightarrow \) Talk spurt duration

\( s_h \Rightarrow \) Silence duration

\( R_h \Rightarrow \) Talk spurt rate

\( \alpha_h \Rightarrow \) Speech activity

The probability density function (pdf) of talkspurts [1] is given as,

\[ f_{A} ( k ) = (1 - r ) r^{(1 - k )}, \]  

(1)
where \( k \) represents the number of frames, and the value of \( k \) ranges from 1, 2,...,L. Each frame represents the length of speech with “L” being the longest speech frame. Talkspurt is represented by \( r \), which is the pdf co-efficient.

The pdf of silence [1] is,
\[
f_1(k) = c_1 (1 - r_1) r_1^{(k-1)} + c_2 (1 - r_2) r_2^{(k-1)},
\]
where \( r_1 \) and \( r_2 \) are the pdf co-efficients representing the short-silence and long-silence periods, respectively. The pause between words and syllables are considered short-silence periods. Long-silences are the pauses between phrases and sentences. In the eq (2), \( c_1 \) and \( c_2 \) account for the weighting and truncation of silence distribution. They are given by,
\[
c_1 = c_2 = c_i = \frac{a_i}{(1 - r_i)^L},
\]
where \( \sum_{i=1}^{2} a_i = 1 \).

3.1.1 Speech Temporal Parameters

This subsection describes the speech temporal parameters relevant to the conferencing call solution.

**Total silence duration (\( S_h \))**

The total silence duration is defined as the overall silence duration incurred by the system. The total silence duration is given by
\[
S_h = S_1 + \sum_{i=1}^{N-1} S_i,
\]
where $s_i$ represents the short-silence as well as long-silence periods of the user who had spoken. The second term in the equation represents the silence period of other users who are all listening with $i$ representing the users.

**Talkspurt rate ($R_h$)**

The talkspurt rate is defined as the total number of talkspurts observed per total time period.

$$R_h = \frac{\text{num of talkspurts}}{\text{time}},$$

$$= \frac{N_h}{T},$$

where $N_h$ represents the total number of talkspurts in the total time period $T$. Now, the number of talkspurts is given as,

$$N_h = \frac{T}{(T_h + s_h)},$$

$$N_h \approx \frac{T}{T_h},$$

where $T_h$ and $s_h$ are the talkspurt and silence duration, respectively. In the conference call, $s_h$ is of negligible value, as there are at least three users in the call and thereby it was neglected. Therefore, the talkspurt rate becomes,

$$R_h = \frac{1}{T} \times \frac{T}{T_h},$$

$$R_h = \frac{1}{T_h}. \quad (4)$$
**Speech activity rate ($\alpha_h$)**

The speech activity rate is,

$$\alpha_h = \sum_i \frac{\alpha_i}{T}, \quad (5)$$

where $\alpha_i$ represents the speech activity of each user $i$ in time period $T$. Speech frames of longer length are considered the speech activity.

### 3.2 IP Traffic Model

In this section, the arrival rate of VOIP traffic is analyzed and the distribution is proposed. The probability distribution function (PDF) of packet arrival with $\lambda$ as the arrival rate is

$$F(t) = (1 - e^{-\lambda t})U(t - t_0) + \frac{1}{n} \sum_{i=1}^{m} X(i) \quad (6)$$

where $U(t-t_0)$ represents the unit step function, with $t_0$ as the initial time period when a user begins to speak. The inter-arrival time of the traffic is considered to be exponentially distributed. The second part of the equation represents the bursty nature of traffic, which is otherwise denoted as talkspurts. The bursty nature of traffic will be a self-similar nature with $X(i)$ representing the traffic sequence in $n$ levels of aggregation. The duration of voice traffic is represented by $t$.

### 3.3 State Transition Diagram and Transition Matrix

Each state in the state transition diagram signifies the number of active traffic sessions in the conference call. The number of traffic sessions is denoted as $n$. Figure 6 shows the state transition diagram of number of sessions.
Figure 6: State Diagram Representing Number of Sessions

The transition probabilities are shown below:

\[
\begin{align*}
 p_{01} &= p_{\alpha} + p_{\tau}, \\
 p_{10} &= 1 - p_{01}, \\
 p_{11} &= p_{\alpha}, \\
 p_{12} &= p_{23} = p_{34} = 1 - p_{12}, \\
 p_{n (n - 1)} &= 1,
\end{align*}
\]

where \( p_{\alpha} \) and \( p_{\tau} \) represent the probability of speech activity rate and probability of talkspurt duration, respectively. The transition matrix (Q) of the markov chain is as follows:

\[
\begin{bmatrix}
0 & (p_{\alpha} + p_{\tau}) & 0 & 0 & 0 & \ldots & \\
1 - (p_{\alpha} + p_{\tau}) & p_{\alpha} & p_{\tau} & 0 & 0 & \ldots & \\
0 & 1 - p_{\tau} & 0 & p_{\tau} & 0 & \ldots & \\
0 & 0 & 1 - p_{\tau} & 0 & p_{\tau} & \ldots & \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \\
0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}
\]
3.4 Calculation of Delay

This section explains and discusses the delay incurred by the system for processing the information received from all active sessions. The total number of input bytes \( B \) that the SIP entity must process at a particular time period is

\[
B = \frac{n(\lambda_{\text{max}} \delta)}{w_{\text{link}}},
\]

where \( n \) represents the total number of sessions, with \( \lambda_{\text{max}} \) as the maximum of the arrival rate of all sessions that are involved in the call, and \( \delta \) is the average packet size. The download link speed of the system is represented as \( w_{\text{link}} \).

The processing time \( P \) of incoming bytes with \( \rho \) as the processing capacity of the SIP application becomes,

\[
P = \frac{B}{\rho}.
\]

Therefore, the total delay \( T_{\text{delay}} \) is

\[
T_{\text{delay}} = P + q_{\text{in}} + q_{\text{out}}
\]

where \( q_{\text{in}} \) and \( q_{\text{out}} \) are the delays incurred in the input and output buffers.

The difference between arrival time of the chunk of bytes and start of processing time is calculated as \( q_{\text{in}} \). However, it was assumed that as the chunk of bytes is processed, it is transferred to the destination at the same time without \( q_{\text{out}} \) and hence was neglected.

Hence, the total delay with \( q \) denoting the input buffer is,

\[
T_{\text{delay}} = P + q.
\]
3.5 Estimating the Number of Sessions

The number of possible sessions that the conference server could support was estimated by using reducing the Markov chain into two states, namely the underload and overload states. Given the SIP application processing capacity in the system, the amount of traffic it could support efficiently depends upon the queuing delay as well as processing delay. The number of sessions that a conference server should handle increases with the bytes of information that it must process. The nature of voice traffic was categorized into three different cases. The first scenario is when user speaks alone. Speech activity from one user coupled with a burst at random intervals is the second scenario. Constant burst traffic generated in all sessions leading to active traffic in all sessions is the final scenario.

The model was studied by assuming the final scenario. The Markov chain was reduced to a two state Markov chains with states as underload and overload as shown in the Figure 7. Assuming that a system can handle the traffic generated from $m$ sessions, then the traffic intensity generated out of $1:m$ connections are considered as underload, whereas the traffic from $m, m+1...n$ are considered as overload leading to an ineffective functioning of the system.

![Figure 7: Two State Markov Chain](image)

$p_{uo}$ denotes the transition probability from the underload state to the overload state, and $p_{ou}$ is the transition probability from the overload state to the underload state. Assuming that the system can effectively handle $m$ number of sessions with its process capacity and any increase
in further sessions beyond \( m \) will transfer the system to overload state, then the probability distribution function (PDF) of aggregate traffic from all sessions becomes,

\[
F(t) = 1 - \prod_{i=1}^{n} e^{-\lambda_i t},
\]

where the value of \( i \) ranges from 1,2,3...\( m, m+1...n \). The arrival rate of traffic in a session is represented by \( \lambda_i \). Letting \( \lambda_a \) be the arrival time of the traffic, then the servicing time becomes

\[
\mu = \lambda_a + T_{delay}.
\]

The inflow of a large number of bytes due to an increase in the number of active sessions may lead to queue delay, which in turn depends on the processing capacity of the system. Therefore, the incoming traffic at a specific time depends on the present amount of traffic being serviced. As a result, considering that service time distribution is Markovian and exponentially distributed, thereby the service time distribution becomes,

\[
F(\mu = m) = 1 - e^{-\mu t},
\]

\[
= 1 - e^{-(\rho+q)t},
\]

\[
F(\mu = m) = \frac{1}{1 + \rho} e^{-(\rho+q)t},
\]

(10)

where \( B, \rho \) and \( q \) denotes the input bytes to the system, processing capacity and delay respectively.

The value of \( P_{uo} \) depends on \( B, \rho \) and \( q \) as show in eq(10). As the input bytes \( B \) increase to be processed for a constant processing capacity \( \rho \), the queue delay increases as well as \( P_{uo} \) too. Since the input bytes depend on the number of active traffic sessions \( n \) which in turn depend on the number of users \( N \) and thereby controlling the number of users for an acceptable
delay $q$ encountered by the system, an effective conference call could be achieved. Since $\lambda_u$ is the arrival timestamp of the incoming traffic, then $p_{uo}$ becomes,

$$p_{uo} = 1 - \frac{\lambda_u}{\lambda_u + T_{delay}}.$$  \hspace{1cm} (11)

Hence, the probability that the system could move from overload to underload is,

$$p_{ou} = 1 - p_{uo}.$$ \hspace{1cm} (12)

From eq(11), it is clear that the system continues to stay in the underload state as the delay is lessened. As the delay increases, the probability of system moving into the overload increases.

In the case of voice traffic on an IP network, the one-way end-to-end delay should be less than 150ms. Therefore, $p_{uo}$ is lower when the queue delay is less. Assuming that the conference server is at a centralized location in the established conference call, then the delay should be less than 75 msec. Hence, for an effective voice conference call, the delay should be less than 75msec, and thereby, $p_{uo}$ should be less than 0.07, which will increase with any increase in delay. When $p_{uo}$ becomes greater than 0.07, the system moves into the overload state. As the system changes into the overload state, the number of sessions at that point translates into the maximum number of sessions handled by the conference server.
CHAPTER 4
SIMULATION AND RESULTS

This chapter includes a discussion of the simulation logic and the observed results from the simulation. The Matlab simulator tool was used to simulate the proposed model. The arrival traffic and servicing time follow the Poisson distribution. This chapter is divided into two sections. The first section explains the simulation logic and the second section will discuss the achieved results.

4.1 Simulation Logic

The metrics of the system under a simulation environment are obtained at the beginning of the program. The metrics that were obtained are,

a) Simulation time
b) Number of users
c) SIP application processing capacity

Depending upon the number of users provided, the number of possible sessions was estimated. For example, if “N” is the number of users, then the number of sessions becomes \( n = N \) in the multicast and \( n=N(N-1) \) in the unicast scenario. The simulation was considered in three scenarios,

a) Active traffic in only one session \( (n = 1) \)
b) Active traffic in one session and traffic in certain other sessions
c) Traffic burst in all the possible sessions

Since the objective of this thesis is to estimate the benchmark number of sessions that the system could support, results obtained using the third scenario are discussed here. The estimation
of maximum delay incurred by the system when it reaches the final state leads to the estimate of number of users that the conference server can handle effectively.

The amount of data received by the system was considered to be 57600 bit/second. This value was chosen in accordance with the G711 codec rate, which is 64kbits/second. Also, the codec G711 was almost supported entirely by all the SIP endpoints and therefore, the requirement to consider the transcoding was neglected.

The simulation time was considered to be one second; therefore one chunk of data in size of 57600 bits would be received by the system, and the arrival time was noted. As the data is received, depending upon the SIP application processing capacity, the received data is processed.

The conference server can work in two possible scenarios. Either the received media information is unicasted to the other intended recipients, or the information is multicasted to all other intended recipients. The simulation was also executed similarly. In the unicast scenario, the total number of traffic sessions to be handled was considered to be $N (N-1)$ with $N$ as the number of users. For example, if there were three users in the call, then total number of sessions becomes six. In the call, one user talks and the media is forwarded independently to two other users. Therefore, the total amount of traffic that needs to be handled is doubled. In the traffic burst scenario, all incoming sessions will carry the active traffic and it must be forwarded to other participants. Hence, the amount of traffic becomes twice that of each of the session. As a result, the number of sessions is doubled and becomes six for three users. In the case of the multicast scenario, the traffic from one active incoming session is multicasted to two other participants at a time and therefore, the number of sessions is equal to three in the conference call of three participants.
The simulation was executed for five different application processing capacity values: 10kbps, 50kbps, 100kbps, 500kbps and 1000kbps. Three to seven users were considered. The application data size was considered as 45 packets with 1280 bytes of data each for a unit second. Therefore, the system received the media size of 57600 bytes for a time period of one second.

It was also assumed that the buffer that the application uses to keep the data before processing it varies depending upon the processing capacity. The buffer size for this simulation was directly proportional to the processing capacity of the system. Hence, the buffer for less processing capacity was less than buffer for higher processing capacity to efficiently utilize the system resources. The impact of buffer on delay of voice real-time traffic is observed later in this chapter. The results are explained separately for both the unicast as well as the multicast scenarios.

4.2 Simulation Results

This section includes the results observed from the simulation experiment and brief description about the result.

The graph in Figure 8 shows the delay incurred by the traffic in the unicast scenario at 10kbps processing capacity. As an example for three users, the total number of sessions becomes six. All the sessions were considered active; therefore, the server received 57600 bits of data from each of the sessions. As the graph shows, the maximum delay grew along with an increase in the number of users and it reached the value of 120ms for seven users in the conference call. This result suggests that 10kbps of processing capacity does not seem to be enough for more than five users in the call. In comparison with the mathematical model, it can be noted that $p_{wo}$ becomes greater than 0.07 for more than five users in the call.
Figure 8: Delay Graph in the Unicast at 10kbps Processing Capacity

The results shown in Figure 9 were captured for the same 10kbps processing capacity but in the multicast scenario. In this case, the number of session became three, and the maximum delay incurred by the traffic grew slightly with the number of users in the call. The maximum delay incurred by the system was 35ms when there were seven users in the call. In comparison with mathematical model, $p_{uo}$ was lesser than 0.07 and therefore, the processing capacity of 10kbps does worked well in multicast scenario for seven users.
Figure 9: Delay Graph in the Multicast Scenario at 10kbps Processing Capacity

The figure 10 is the comparison of delay incurred by the server between unicast and multicast scenarios with 10kbps of application processing capacity.

Figure 10: Comparison of Delays between the Unicast and Multicast Scenarios at 10kbps
The results show that the delay is much higher in the case of unicast scenario compared to the multicast scenario with 10kbps of processing capacity.

The graph in Figure 11 displays a comparison of the delay incurred by the conference server at various processing capacities and number of users.

![Figure 11: Comparison of Delays in the Unicast Scenario](image)

It can be observed that the delay is in the higher range for all the values of n with processing capacity of 10 kbps. Also, the delay is almost less than 5ms for all values of n with processing capacity of 50 and 100 kbps. The delay is as low as 2 ms for processing capacity of 100kbps. But the delay rises sharply for 500kbps of processing capacity due to the considerable increase in buffer size. The delay for 1000kbps declines and settles at 40 msec. Hence, it can be
concluded that the efficient operating margin of the system is at 100kbps of application processing capacity in the unicast scenario.

The graph in Figure 12 was captured similarly but in the multicast scenario. The results show that delay is less in comparison to that of in the unicast scenario. Even in the multicast scenario, it can be observed from the results that the optimum operation margin is between 50kbps and 100kbps of processing capacity. Delay rises sharply for processing capacity of 500kbps to 80msec but declines to 40msec at the processing capacity of 1000kbps.

![Multicast Graph](image)

**Figure 12: Comparison of Delays in the Multicast Scenario**
From the obtained simulation results, the optimum processing capacity margin of the conference server and number of sessions it can handle when there is traffic burst in all sessions can be noted. Hence, it can be concluded that the efficient operating margin of the conference server is between 50 and 100kbps for even a bursty nature of traffic. The optimum operating capacity value becomes 100kbps. It should be noted that the optimum processing capacity margin was observed for seven users in the simulation.
CHAPTER 5

CONCLUSIONS AND FUTURE WORK

The proposed model analyzes the effective voice data handling capacity of the conference server. From the model, it can be concluded that for effective voice conference, the probability that the system should stay within the underload state depends on the value of $p_{uo}$. The queue delay depends on the amount of incoming traffic and system processing capacity. In turn, the incoming traffic depends upon the number of users in the conference call. Therefore, in order to have less delay and effective voice conference operation, the proposed model showed that the value of $p_{uo}$ should remain less than 0.07.

Outputs from the Matlab simulation showed an increase in delay when the processing capacity was less. The optimum performance was achieved only with a moderate range of processing capacity. Due to the increased buffer, a higher margin of processing capacity does not yield the better expected results. From the output, it can be observed that the efficient effective processing margin was between 50kbps and 100kbps for seven users in the conference call. The optimum operating capacity value was 100kbps.

The location of the conference server could be an interesting future work. Since voice is a real-time traffic, the location of conference server could be a factor in delay and voice quality. In this thesis, it was assumed that the conference server is at the middle point but the dynamic selection of its location could be an interesting investigation.
REFERENCES


