# TOWARDS OPTIMUM PLAY-OUT BUFFERING DELAY IN VOICE OVER IP

by

RUCHIR PRAMOD SHENDE

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2006

## ACKNOWLEDGEMENTS

I extend my sincere gratitude and appreciation to my supervising professor Dr. Ramesh Yerraballi for his guidance, support and infinite patience which resulted in the shaping of this thesis. I would like to thank Mr. Mike O'Dell and Dr. Gautam Das for being in my defense committee. I also thank my friends for their constant support.

I would like to take this opportunity to dedicate this thesis to my parents and sister who provided me with tremendous moral and emotional support. Without them I wouldn't have made it this far.

April 19, 2006

## ABSTRACT

TOWARDS OPTIMUM PLAY-OUT BUFFERING

DELAY IN VOICE OVER IP

Publication No. ______

Ruchir Pramod Shende, M.S.

The University of Texas at Arlington, 2006

Supervising Professor: Dr. Ramesh Yerraballi

Voice over Internet Protocol (VoIP) or transmission of real-time voice packets over the Internet is slowly emerging as a cost-effective alternative to the traditional Public Switched Telephone Network (PSTN) and this trend is expected to continue in the future. However, varying end-to-end delay and packet loss, which are inherent in a packet-switched network like the Internet, lead to relatively lower quality of VoIP calls. The call quality can be improved by adaptively adjusting the play-out buffer at the receiver to reduce the impact of the delay and jitter during the transmission. A standard play-out strategy uses a weighted moving average of the mean and variance of network delay to adaptively set the play-out deadline. Other techniques used include adaptive adjustment of talk-spurt and silence periods. Adjustments can also be made within talk-spurts by scaling individual voice packets using time-scale modification.

In this thesis, we simulate an adaptive play-out algorithm that uses smoothed average of network delays and analyze its performance using a discrete event-based simulator that is developed in Java. For the observed packet loss rate, we determine the optimum average buffering delay and compare it to the average buffering delay that the packets experience as a result of the algorithm. We then tweak the algorithm so that the average buffering delay reported by modified algorithm is nearer to the optimum value. As a result, the end-to-end delay will be reduced improving the call quality perceived by the end-user.

TABLE OF CONTENTS

# LIST OF ILLUSTRATIONS

## LIST OF TABLES

# CHAPTER 1

# INTRODUCTION

## 1.1 Background

Voice over Internet Protocol (VoIP) is the technology used to transmit voice conversations over a packet switched network using Internet Protocol (IP). It consists of a set of facilities and protocols for managing the transmission of packetized voice using IP. Internet telephony is one of the typical applications of VoIP. It can serve as an alternative to the standard Public Switched Telephone Networks (PSTN) by offering the same or even better services at a reduced cost.

The possibility of voice communications traveling over the Internet first became a reality in February 1995 when Vocaltec Inc. introduced its InternetPhone software. It was designed to run on a Personal Computer (PC) equipped with a sound card, speakers, and a microphone. The software would compress the voice signal and translate it into IP packets for transmission over the Internet. Over the last decade, VoIP has come a long way. The number of residential VoIP lines is expected to grow to 27 million by 2010 in the US alone [1]. The rapid proliferation of the Internet over the last few years has also fuelled the spread of VoIP. The success of VoIP can be attributed to the following reasons:

1. IP networks are more cost-efficient. The major advantage is the reduction in the cost of long distance and international calls since the call cost is

reduced to cost of paying for local Internet Service Provider (ISP) only. Since the operators can avoid paying the interconnect charges incurred over the PSTN, the end-user bypasses per minute charges of the PSTN and pays only a flat monthly rate to the ISP. This also makes switching over to VoIP attractive for companies.

2. IP networks are more bandwidth efficient. The PSTN uses time division multiplexing where bandwidth is dedicated to a particular call even though the channels maybe idle. But IP networks use the bandwidth more efficiently as a number of calls are multiplexed over the same link during times of silence in the conversation. Thus while a PSTN call takes up to 64 kbps, a VoIP call only takes about 6-8 kbps or possibly lower through the use of compression algorithms.

3. IP networks can provide integrative data and voice services. Since many people use a modem to connect to the Internet and have a single phone line, the use of VoIP enables them to place and receive calls while online thus eliminating the need for another phone line to avoid missing calls while online. VoIP can also offer new value-added services such high-fidelity stereo conferencing, multicast conferencing and telephony distance learning applications etc.

4. It provides the users with greater mobility as they can travel all over the world and still make and receive phone calls as long as they have access to the Internet.

## 1.2 Motivation

As the Internet is evolving into a universal communication network carrying both data and voice traffic, it will be expected to provide the toll quality standards set by the traditional telephone companies at the very least, if not better. The current telephone system is based on circuit switching which provides a dedicated path between the two users with guaranteed bandwidth to each call and thus can ensure good call quality as perceived by the end-user. The Internet however inherently provides a best effort service due to its packet-switched nature. It was never engineered to handle real-time delay-sensitive applications such as VoIP. Voice traffic in the Internet incurs variable loss rates and delays which degrade the call quality. Generally, packet loss rate of under 10% and one way end-to-end delays of under 150 milliseconds [2] do not undermine the quality significantly. Ensuring that these values are not exceeded in the Internet thus providing guaranteed service would involve changes to the fundamental design of the Internet which is a difficult proposition. Therefore loss and jitter in the Internet need to be compensated by application–level techniques at the end-user.

Packet loss can be overcome to some extent by a number of mechanisms like silence substitution, noise substitution, packet repetition, packet interpolation, frame inter-leaving, Forward Error Correction (FEC) [3] etc. With continuously improving equipment and technology, the loss rates experienced by packets in the Internet are decreasing. As reported by [4], the global packet loss rate in the Internet stands at about 3% today. Delay and jitter, on the other hand, continue to be roadblocks in successful mass deployment of VoIP. Jitter is compensated primarily through adaptive play-out delay algorithms which take into account delay experienced by packets and calculate the updated play-out times on a talk-spurt to talk-spurt basis. The packets arriving at the receiver are buffered to allow delayed packets to arrive before their play-out deadline. These algorithms trade delay for loss. A larger buffer will increase delay but reduces the loss. Thus it would seem that the buffer size should be small but this will result in more packet loss. Also the calculation of the play-out times for the packets plays a key role. The play-out times need to be calculated based on the current network conditions as experienced by the packets. Play-out algorithms determine to a large extent the quality of VoIP calls. Hence we need to analyze it in detail and design one that strikes a balance between delay and loss and also computes the optimum play-out times.

## 1.3 Thesis Overview

The rest of the thesis is organized as follows: chapter two provides a detailed view of how a VoIP call is placed. It also talks about the protocol stack for VoIP.

Chapter three explains the various factors that affect the performance of a play-out algorithm. It describes how the quality of a call is determined quantitatively and what trade-offs are involved in improving it.

Chapter four explains how the optimum buffering delay, for a given packet loss rate, can be determined.

Chapter five describes the original play-out algorithm and the modified algorithm which performs closer to the optimum buffering delay values.

Chapter six is dedicated to explaining the simulation of the play-out algorithm. It gives details about the simulation set-up, the values of the parameters involved, the assumptions made etc. It presents graphical results of the performance of the original and modified play-out algorithms.

Chapter seven concludes this work.

# CHAPTER 2

# UNDERSTANDING THE BASICS OF VoIP

## 2.1 Making a VoIP Call

There are three different ways [5] to place a VoIP call depending on the equipment being used at the end-points.

1. Computer-to-Computer:

This is the most easiest and common way of using VoIP and certainly a cheaper method. All you need to place a call is one of the readily available free or low cost software, microphone, soundcard, speakers and an Internet connection, preferably a fast one through a cable or DSL modem. Both parties need to use the same software and there is usually no charge for placing these calls irrespective of the distance.

2. Analog Telephone Adaptor (ATA):

ATA is a device that enables a standard phone to be connected to a computer or the Internet connection to place a call. It takes the analog signal generated by the traditional telephone and converts into a digital signal for transmission over the Internet.

3. IP-phones:

IP-phones are specialized phones that look just like traditional phones. But instead of having standard RJ-11 phone connectors, they have RJ-45 Ethernet connectors which allow them to connect directly to the router. All the hardware and software needed to handle VoIP calls is integrated into the phone itself.

Thus depending on the end-terminals, there are three possible configurations of making a VoIP call --- PC-to-PC, PC-to-Phone or Phone-to-PC and Phone-to-Phone. Placing a PC-to-PC call is relatively straightforward since both the end-points are connected to the same network. However placing a PC-to-Phone or Phone-to-Phone VoIP call requires an additional hardware called the Gateway to bridge the PSTN with the Internet. It enables the circuit-switched PSTN to interoperate with the packet-switched Internet. It also overcomes the issue of addressing since a PC user can be located anywhere in the world. Prior to the use of a Gateway, in order to contact a remote PC, the user need to know its IP address which is not obtained easily. The Gateway, however, enabled access to the remote PC by the called party's telephone number provided the remote PC was equipped with a Gateway. The basic functionality of a Gateway can be explained with the help of the following figure [6]:

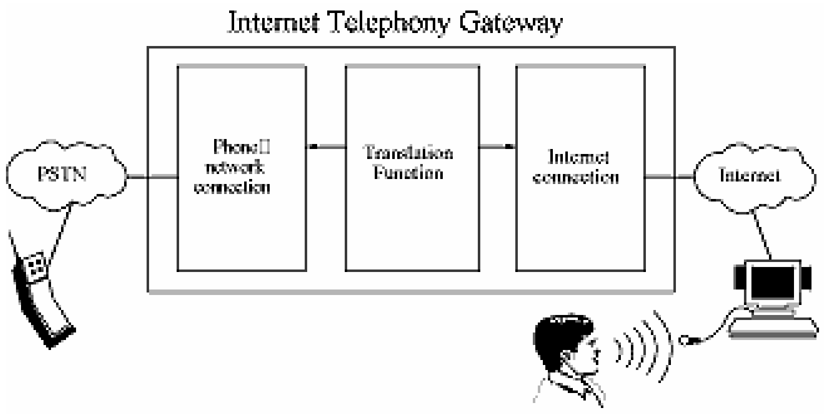


Figure 2.1: Internet Telephony Gateway

When it receives a standard telephone voice signal, it first converts the analog signal into a digital one. It then compresses it, segments the signal into IP packets for transmission over the Internet and finally sends them out on the Internet for delivery to the destination gateway which reverses the whole process.

Using the Gateway, a PC-to-Phone connection is established in the following manner:

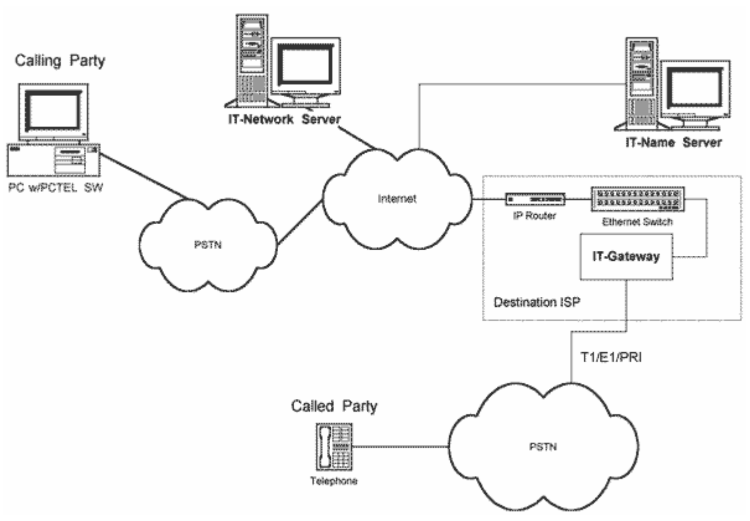


Figure 2.2: PC-to-Phone Connection

After registering with a VoIP service provider and obtaining the necessary software, the caller connects to and provides the ISP with the telephone number of the called party. The ISP provides a gateway to the caller to connect it to the Internet. The caller’s voice signal is then digitized, compressed and converted into IP packets at the gateway. These IP packets are then transferred to the called party’s gateway via a path determined by the ISP gateway. The destination gateway reverses the whole process and converts the IP packets into voice signal. The voice signal is then transferred over the local PSTN of the called party thus establishing the call.

Similarly, a Phone-to-Phone connection establishment is explained by the following figure [7]:

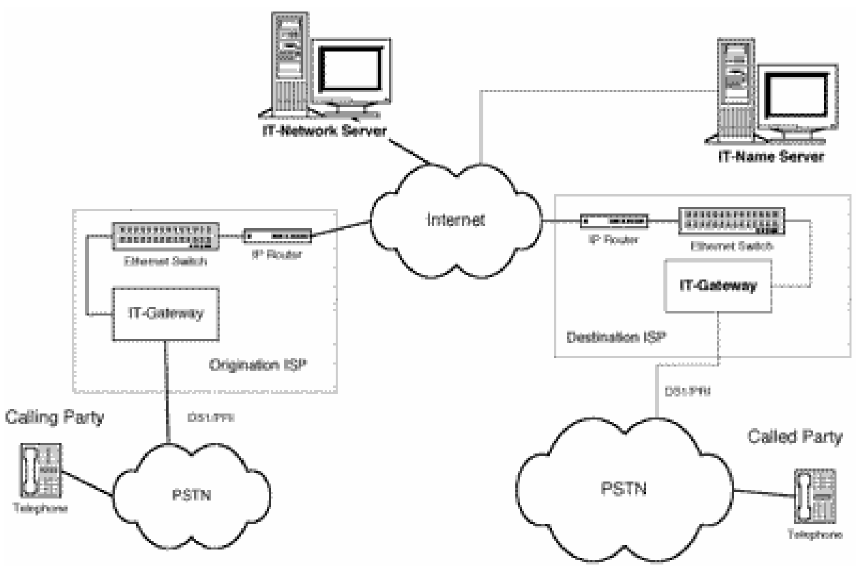


Figure 2.3: Phone-to-Phone Connection

## 2.2 VoIP Protocol Stack

The protocol stack [8] involved in transmitting voice over the Internet can be represented diagrammatically as follows:

| | | | |
|---|---|---|---|
| Voice | | | |
| RTP | | | |
| UDP | | | |
| IP | | | |
| PPP | | | |
| LAPM | | LAPB | |
| HDLC | | | |
| V series | ISDN | ADSL | HFC |

Figure 2.4: VoIP Protocol Stack

| | |
|---|---|
| Layer 7.RTP | Real Time Protocol |
| Layer 4.UDP | User Datagram Protocol |
| Layer 3.IP | Internet Protocol |
| Layer 2.PPP | Point-to-Point Protocol |
| LAPM | Link Access Procedure for modems |
| LAPB | Link Access Procedure Balanced |
| HDLC | High Level Data Link Control |
| Layer1. V Series modems | |
| ISDN | Integrated Services Digital Network |
| ADSL | Asymmetric Digital subscriber Line |
| HFC | Hybrid Fiber Coax |

Layer 1 represents the various technologies by which the user can connect to the Internet such as the ITU-T recommended V Series for modems. The commonly used 56 kbps modems are specified in the V.90 specification. ISDN represents another way of connecting to the ISP but at higher data rates than those provided by modems. ADSL technology provides the user with capacities in Mbps. The HFC network exploits the bandwidth capacity of fiber and coax in different parts of the network and is emerging as an alternative to ADSL.

Layer 2 includes protocols such as PPP which allows two machines to negotiate the type to network layer protocols that will be used, compression and authentication procedures etc. It relies on the HDLC protocol for framing, error-checking and bitstuffing. It may also use the LAPM or LAPB to ensure proper sequencing of frames in the incoming traffic. PPP is also used by the ISP to assign an IP address to the user.

The IP is used in Layer 3 to move the voice packet from node to node in the Internet based on the destination IP address.

UDP provides transport layer services in Layer 4 but it is a connectionless protocol providing no guarantees about reliability and ordering of packets. However it is faster and efficient for time-sensitive applications such as VoIP.

RTP defines a standardized packet format for delivering audio and video over the Internet. RTP is designed for support of real time traffic. It is also an encapsulation protocol in that the real time traffic runs in the data field of RTP packet and RTP header contains information about the type of traffic. The time-stamp field in its header is used to synchronize traffic play-out at the receiver.

Current implementation of VoIP is based on architecture proposed by the ITU-T – the H.323 protocol suite. The following provides an overview of the H.323 standard.

H.323:

H.323 is an umbrella recommendation from the ITU-T that defines the protocols to provide audio-visual communication sessions on any packet network. As explained in [9], it defines how voice, data and video traffic will be transported over IP-based local area networks. It incorporates the T.120 data conferencing standard and is based on the RTP/RTCP protocol for managing audio and video signals. It does not include the network interface, the physical layer or the transport protocol used. It defines the functions of the application layer protocols. H.323 entities may provide real-time audio, video and/or data communications. Support for audio is mandatory while data and video are optional. It addresses core Internet telephony applications by defining how delay-sensitive traffic gets priority to ensure real time communications.

The H.323 entities are as follows:

a. Terminal –

It is an end user device that provides real-time, full duplex voice, video or data communications with another H.323 terminal. It can also communicate with a Gateway Multipoint Control Unit (MCU). A MCU is a terminal that provides support for multipoint conferences.

b. Gatekeeper –

It provides address translation and call control services to the endpoints. Network access is authorized by the Gatekeeper using H.225.0 messages. It translates an incoming E.164 address i.e. telephone number to a transport layer address. It is also responsible for bandwidth control.

c. Gateway –

It is a node on a LAN that communicates with the H.323 terminal or other ITU-T terminals attached to other networks. It also performs translation of transmission formats between terminals. Communications with PSTN devices is accomplished using Gateways. Working with the Gatekeeper, it can set up and clear calls.

H.323 Protocol Stack:

H.323 consists of several standards. For audio applications, G.711 support is required while other standards such as G.722, G. 723, G.728 and G.729 are optional.

However the VoIP forum has recommended G.723.1 specification. The video standards are H.261 and H.263. Data support is through T.120. The various control signaling and maintenance operations are provided by H.245 and Q.931 specifications. The frame size and bandwidth needed by each codec is presented in the table below:

Table 2.1: Voice Codec Overview

| CODEC | BANDWIDTH (kbps) | FRAME SIZE (ms) |
|---|---|---|
| G.711 | 64 | 1 |
| G.722 | 32-64 | 1 |
| G.723.1 | 5.3, 6.4 | 30 |
| G.728 | 16 | 2.5 |
| G.729 | 8 | 10 |

The audio and video packets must be encapsulated into RTP and carried on UDP socket pair between sender and receiver. The RTCP is used to assess the quality of the sessions as well as connections to provide feedback. The data and support packets can operate over TCP or UDP.

Session Initiation Protocol (SIP):

SIP is a signaling protocol for Internet conferencing and telephony developed by the Internet Engineering Task Force (IETF). It is an application layer control signaling protocol that can establish, modify and terminate multimedia sessions with one or more participants such as VoIP calls. It can also invite participants to existing sessions. It transparently supports name mapping and redirection services, which supports personal mobility in that users can maintain a single externally visible identifier regardless of their network location. It supports the following five facets of multimedia communications:

1. User location – determination of the end system to be used for communication.
2. User availability – determination of willingness of the called party.
3. User capabilities – determination of the media and parameters to be used.
4. Session setup – establishment of session parameters.
5. Session management – transfer and termination of sessions, modifying parameters and invoking services.

SIP has proved to be a very efficient and useful support tool for IP telephony due to the following reasons:

1. It can operate as stateful or stateless. The stateless implementation offers good scalability and is very robust.
2. It uses much of the syntax and format of HTTP thus providing compatibility with current browsers.
3. SIP message body is opaque and thus can be of any syntax. Thus it can be described using MIME or XML.
4. It identifies users with a Uniform Resource Identifier (URI) thus providing the user the ability to initiate a call by clicking on a web link.

# CHAPTER 3

# PERFORMANCE CONSIDERATIONS

## 3.1 Factors affecting Performance

The inherent packet-switched nature of the Internet introduces obstacles to providing quality of voice which matches that provided by PSTN. These obstacles stem from the loss, delay and jitter which voice packets have to contend with in the Internet.

Since no end-to-end circuits are established in IP networks, packets of all types including voice and from all sources are queued for transmission on the outgoing link in a router and transmitted one by one from the head of the queue. A packet is lost if there is no more space in the queue. As the traffic increases, the routers face increasing congestion and drop more packets resulting in packet loss. Packets can also be lost due to errors during transmission. However, with improving technology, such errors have reduced significantly to almost being negligible. Thus congestion contributes the most towards packet loss. Packet loss has a severe effect on voice quality. Each packet contains speech information of critical duration called phonemes. While the loss of a few packets and hence phonemes can be reconstructed by the human brain, too much packet loss can render the speech unintelligible. A packet loss rate of around 10% is generally considered acceptable [2]. The effects of packet loss can be mitigated to some extent by using techniques such as Forward Error Correction (FEC), silence or noise substitution, packet repetition, packet interpolation, frame interleaving etc. The figure below [3] shows how voice quality degrades as loss rate increases.

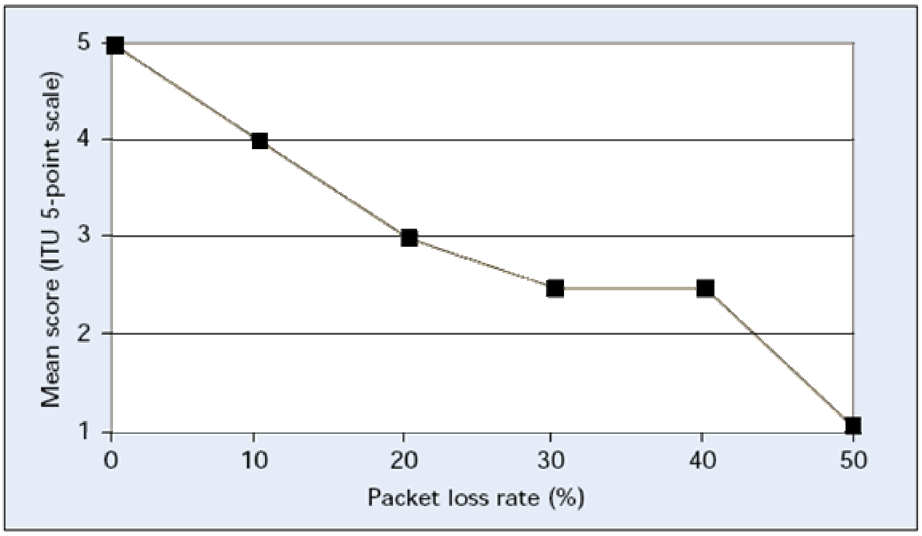


Figure 3.1: Voice Quality as a function of Packet Loss Rate

The second factor contributing to lower voice quality over the Internet is delay and variation of delay called jitter. Packets in the Internet are subject to end-to-end delay which is a function of variable delays due to processing, queuing and fixed propagation delay. This variation introduces jitter which means that the time difference between two packets transmitted at the sender is unlikely to be the same as that observed between their arrival times at the receiver. This timing is important to voice quality since the interval is as much a part of the voice as the uttered syllable. If the delay between the packets is increased, it affects the intelligibility of speech adversely. Significantly long delays can render the conversation between the two parties to a half-duplex communication where one party speaks and the other listens and pauses to make sure the other is done before talking. With this type of communication, the users can end up stepping on each other's speech causing talker overlap. Delays also cause echo by which the speaker can hear his own voice which is reflected from the other end back to his ear-piece. As mentioned in [3], ideally total end-to-end delay should be 150 milliseconds. Delays between 150 and 400 milliseconds are acceptable while voice quality becomes unacceptable with delays of over 400 milliseconds which impair interactive applications such as Internet telephony.

There are a number of different sources of delay in the Internet. Codec delay is introduced when the analog voice signal is digitized at the sender and vice versa at the receiver. Codec also compresses the voice signal to reduce bandwidth requirement. This conversion and compression add to the delay. Different codecs have different amounts of delay. Serialization delay is the time it takes to place a packet on the transmission line and is determined by line speed and frame size used by the codec. Queuing delay is caused at various routers and gateways where voice packets have to wait behind other packets to be transmitted over the outgoing link. This contributes to the variability of the delay. Propagation delay is the time required for packets to travel from one point to another and is determined by the speed of light and distance to be traveled. The figure below [3] illustrates the various types of delay encountered by voice packets as they travel from one end to the other.

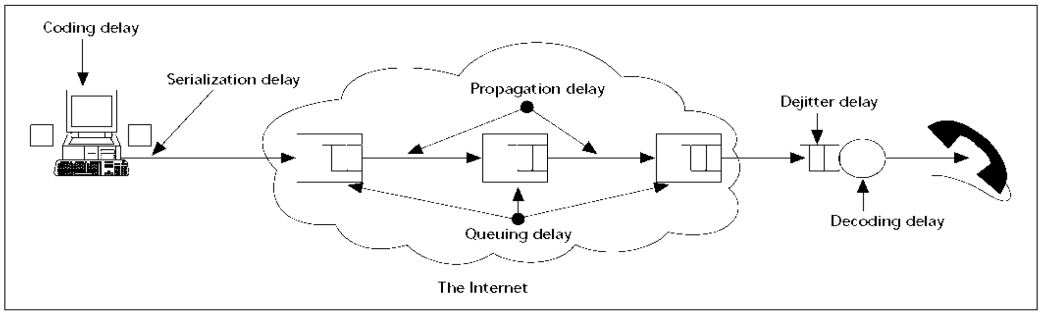


Figure 3.2: Sources of Delay in the Internet

In order to compensate for the jitter introduced in the network, VoIP applications buffer incoming packets and delay their play-out at the receiver. This delay is called de-jitter delay or buffer play-out delay. To allow for variable packet arrivals times and still maintain a steady interval between packets as that generated by the sender, the receiver delays the play-out of packets by holding them in a buffer. This adds to the total end-to-end delay playing a significant role in the end-user's perception of quality.

## 3.2 Determination of Quality of VoIP Call

The E-model is a computational model standardized by the ITU-T that uses transmission parameters to predict the subjective quality of packetized voice. As described in [1], the E-model is used to calculate the R-factor which is a simple measure of voice quality ranging from a best case of 100 to a worst case of 0. It combines the impairment caused by different parameters such as delay and loss into a single rating and is based on the principle that the perceived effects of the impairment are additive. The R-factor is given by:

$$R = R_o - I_s - I_d - I_e + A$$

In the above equation, '$R_o$' is the base factor determined from noise levels, loudness etc. '$I_s$' is the impairment that occurs due to quantization. '$I_d$' is the impairment associated with the mouth-to-ear delay while '$I_e$' is the impairment associated with signal distortion caused by low-bit rate codecs and packet losses. The advantage factor 'A' represents the deterioration that callers are willing to accept because of access advantage certain systems have over traditional telephony e.g. mobile telephony. The factor '$I_d$' takes into account the end-to-end delay comprising of codec, transmission, propagation, queuing, and buffering delays while '$I_e$' takes into account the impairments caused by all types of losses.

The R-factor is then used to uniquely determine the Mean Opinion Score (MOS) where a MOS value of 1 is unacceptable while 5 is excellent. The R-factor is related to the MOS in a non-linear manner through the following equation:

$$MOS = 1 + 0.035 * R + 7 \times 10^{-6} * R * (R - 60) * (100 - R)$$

The mapping [1] between R-factor and MOS values is given by the following table:

Table 3.1: Mapping between R-factor and MOS

| R-Factor | Quality of Voice Rating | MOS |
|---|---|---|
| $90 < R < 100$ | Best | 4.34 – 4.5 |
| $80 < R < 90$ | High | 4.03 – 4.34 |
| $70 < R < 80$ | Medium | 3.60 – 4.03 |
| $60 < R < 70$ | Low | 3.10 – 3.60 |
| $50 < R < 60$ | Poor | 2.58 – 3.10 |

Thus VoIP call connections with R-factors of 60 or less are expected to provide poor quality while those of 80 or above provide high quality.

## 3.3 Performance Improvement and Trade-offs Involved

In order to improve the performance and quality as perceived by the user, there are a number of factors such as packet size, buffer size, packet loss rates and latency to be considered as discussed before. The packet loss rate needs to be kept as low as possible. Figure 3.1 in Section 3.1 shows that voice quality decreases drastically as loss rate increases. The packet latency also needs to be kept at a minimum. As discussed earlier, end-to-end delays of less than 400 milliseconds are acceptable.

Packet latency can be reduced by higher connection speeds. Cable and DSL connections give better performance than dial-up connections. Packet size can also help to reduce latency. Packet size is a function of number of user bits and the coding rate of the signal. Smaller packet size would lead to reduced latency. However, too small packet sizes can be detrimental as well since the overhead incurred on them would be very high. Studies indicate [8] that a payload size of 32-64 bytes is acceptable.

The buffer size and hence the buffering delay plays an important part in the user's perceived quality of voice. At the sender, voice packets are generated according to a schedule (for example, every 20 milliseconds). For faithful and accurate play-out at the receiver, this schedule must be maintained. However, the jitter introduced by the network means that packets arrive at intervals different from those when they were generated. If the receiver plays out the packet as and when they are received, there will be gaps in the play-out as some packets arrive later than their scheduled play-out time.

This situation is depicted in Figure 3.3(a) [10] below. Hence play-out scheduling is required at the receiver to smooth out the effects of delay variation in the network. A play-out buffer operates by introducing an additional buffering delay and holding the packets until their scheduled play-out time as shown in Figure 3.3(b) [10] below.

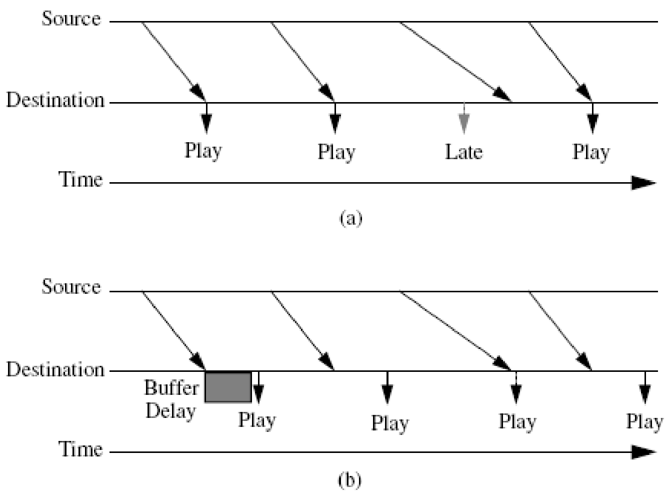


Figure 3.3: Play-out problem (a) packets arrive too late (b) solution using play-out buffer

Any packet arriving later than its scheduled play-out deadline is considered to be lost. Scheduling a packet at a later deadline allows for late arriving packets to be played out in time resulting in a lower loss rate. However, this reduced loss rate comes at the cost of increased buffering and hence overall delay. Setting a buffering delay too high would ensure that almost all packets make it before their deadline but the delays would too high which is undesirable in an interactive application such as VoIP. Vice versa, reduced buffering would mean lower delays but higher number of packets missing their deadlines. Clearly there exists a trade-off between delay and loss. Since the play-out algorithm greatly affects the quality delivered to the user, an efficient algorithm should achieve the best possible trade-off between delay and loss.

# CHAPTER 4

# DETERMINING OPTIMUM PLAY-OUT BUFFERING DELAY

In a VoIP call, the conversation between the end-users consists of alternating periods of talk-spurts and silence. During each talk-spurt, the voice signals are encoded into packets of fixed size. The talk-spurts are followed by periods of silence during which no packets are generated. Packets are buffered at the receiver to smooth out the effects of jitter so that they arrive in time for their play-out. This buffering delay adds to the end-to-end delay and has a significant impact on user interactivity. If the packets are buffered for a long enough time all the packets will eventually arrive before their play-out time but the end-to-end delay would then be unacceptably large. The optimum buffering delay should be as low as possible to reduce the end-to-end delay and at the same time the packet loss should also be minimized.

The adaptive play-out algorithm, operating at the receiver, continuously estimates the network delay and adjusts the buffering delay at the beginning of each talk-spurt. Since the current delay is not known before-hand, it calculates the buffering delay based on the delays experienced by already-received packets. This value of buffering delay is not the optimum one as the algorithm has no knowledge of packets arriving in the future and what their network delays will be. This optimum value of buffering delay can be determined if the arrival times of all the packets in a VoIP call session are known before-hand. Having determined this best possible value, it is then compared to the buffering delay that the algorithm estimates to observe how close the algorithm operates to the optimal value.

Consider a talk-spurt in which packets are numbered from 1 to n. At the receiver, the arrival time of the ith packet is represented by $x_i^a$ and its play-out time is represented by $x_i^p$. The packets should be played out at the receiver at the same rate at which they are generated at the sender. Let the packets be generated at the sender after an interval represented by d. Thus the play-out times of successive packets should be separated by time d. To keep the buffering delay at optimum value, the first packet in a talk-spurt should be played out as soon as it is arrives i.e. $x_1^a = x_1^p$. Thus for the ith packet, its play-out time is given by

$$x_i^p = x_1^a + (i - 1)\, d$$

Let the buffering delay for ith packet be represented by $\delta_i$. This buffering delay is given by

$$\delta_i = x_i^p - x_i^a$$

$$\delta_i = x_1^a + (i - 1)\, d - x_i^a$$

If the $\delta_i$ value is positive, it means that the packet arrives before its play-out time while a negative value means that it arrives after its play-out time and has missed its deadline.

Consider a talk-spurt consisting of 5 packets. The diagram below represents the arrival and play-out times of each packet in this talk-spurt. The play-out times of each packet are represented by upward pointing bold arrows and separated by the interval d. The arrival times are represented by downward pointing bold arrows. In the given talk-spurt, packets 3 and 4 miss their play-out times. Of those, packet 4 misses its play-out time by the most amount of time i.e. $\delta_4$ has the largest negative value. In order to ensure that all packets in the talk-spurt are played out in time, the play-out time of the 4th packet should be moved forward by $| \delta_4 |$. Since packets need to be played out at a constant rate, $| \delta_4 |$ is added to the play-out times of the remaining packets in the talk-spurt as well. These new play-out times $x_i^{p'}$ are represented by the upward pointing dotted arrows.

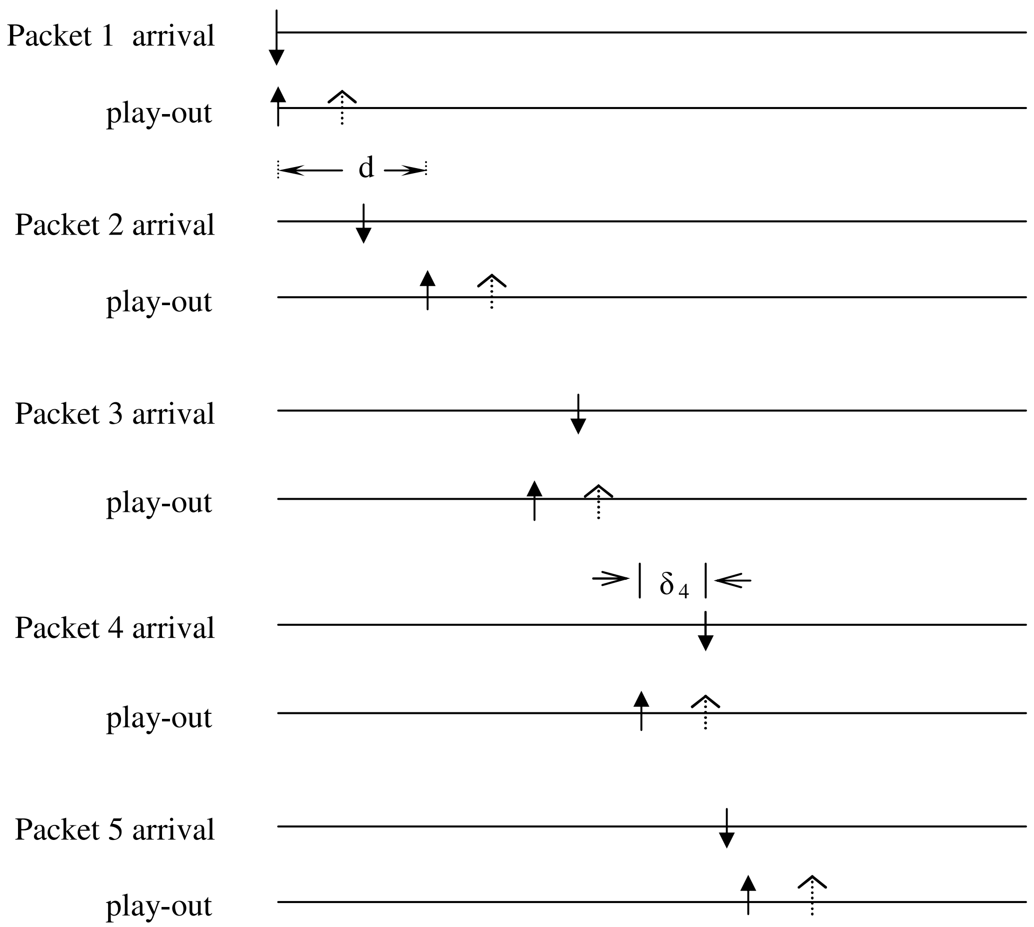


Figure 4.1: Arrival and Play-out times of packets in a talk-spurt

↓ $X_i^a$ -- Arrival time of ith packet

↑ $X_i^p$ -- Play-out time of ith packet

⇡ $X_i^{p'}$ -- Adjusted play-out time of ith packet

As a result of this adjustment, buffering delays of each packet change and are represented by $\delta_i'$. This adjustment is performed across all talk-spurts in the VoIP call session. Now all packets in every talk-spurt are played out in time.

To compare the performance of the algorithm with the optimal buffering delay value, the packet loss rate experienced by both must be the same. This is achieved by sorting packets across all talk-spurts in the ascending order of their new buffering delays $\delta_i'$ and removing the same number of packets as that have missed their play-out times in the algorithm. This ensures that packets having lower $\delta_i'$, values which are most likely to miss their play-out times, are removed. The first packet in a talk-spurt is never removed as its play-out time has already been optimized. After removing these packets, the remaining packets are re-arranged into their proper talk-spurt.

Having re-arranged the packets in their proper talk-spurts, it is possible that the packet that missed its play-out time by the most amount of time (packet 4 in the above talk-spurt) is removed. In such a case, the play-out times of the packets are re-adjusted because now the packets are being buffered by an amount which is more than what is necessary. The play-out times are re-adjusted according to new lowest $\delta_i'$ value. This is achieved by determining the packet with the lowest $\delta_i'$ value in each talk-spurt. This $\delta_i'$

value is then deducted from the buffering delay of every packet in that talk-spurt to give the final optimum buffering delay for each packet in the talk-spurt. This is done for every talk-spurt in the session. The mean buffering delay across all talk-spurts in the VoIP call session is computed. The mean of the mean buffering delays of each talk-spurt is also calculated.

The process of determining the optimum buffering delay can be summarized as follows:

1. For each talk-spurt in the VoIP call session, let the arrival time of the $i^{th}$ packet be represented by $x_i^a$ and its play-out time be represented by $x_i^p$. Calculate the buffering delay of each packet represented by $\delta_i$ in the talk-spurt according to the following equation:

$$\delta_i = x_1^a + ( i -1) d - x_i^a$$

Where d is the interval after which packets are generated at the sender

2. For each talk-spurt, determine the packet with the largest negative value of $\delta_i$. Add the absolute value of this $\delta_i$ to the buffering delay of all packets in the talk-spurt. Perform this adjustment in every talk-spurt. Let the new buffering delays of the packets be represented by $\delta_i'$.

3. Let the number of packets that miss their play-out times as a result of the algorithm be represented by m. To keep the packet loss rate constant for comparison between the optimum buffering delay and the algorithm's buffering delay, sort the packets across all talk-spurts in the ascending order of their new buffering delays $\delta_i'$ and

remove first m packets from this sorted list while ensuring that the first packets in the talk-spurt are never removed as their buffering delays are already optimized.

4. After removing m packets, re-arrange the packets into their proper talk-spurts.

5. For each talk-spurt, again determine the packet with the lowest buffering delay $\delta_i'$. Re-adjust the buffering the delay of each packet in the talk-spurt by subtracting the new lowest buffering delay value from all the packets.

6. These new values now represent the optimum buffering delay for each packet. For comparison, determine the average optimum buffering delay across all talk-spurts and also the average of the average buffering delay of each talk-spurt.

# CHAPTER 5

# ADAPTIVE PLAY-OUT BUFFERING ALGORITHM

## 5.1 Generic Adaptive Play-out Buffering Algorithm

When designing a fixed delay play-out strategy at the receiver, there is an important delay-loss trade-off that arises. In a fixed buffering delay play-out, if the initial play-out delay is kept large, most packets will make their play-out deadlines significantly reducing the loss. But the end-to-end delay will become prohibitively long for an interactive application such as VoIP. Ideally, the buffering delay should be minimized subject to the constraint that the loss is kept below an acceptable limit.

An effective solution to this trade-off is to estimate the network delay and its variance, and to adjust the buffering delay accordingly at the beginning at every talk-spurt. This adjustment is performed for the first packet in the talk-spurt and all subsequent packets in the talk-spurt are scheduled to be played out at fixed intervals following the play out of the first packet. This interval is the same as the interval between packets when they are generated at the sender. The adaptive adjustment at the beginning of the talk-spurt will cause the sender's silence period to be compressed or elongated. However if such variations are reasonably limited, they are not noticeable in perceived speech.

As described in [11], a generic algorithm to adaptively adjust the buffering delay at the receiver is explained as follows.

Let $t_i$ be the time that packet i was generated at the sender

$r_i$ be the time that packet i is received at the receiver

$p_i$ be the time packet i is played out at the receiver

The end-to-end network delay of the ith packet is given by ( $r_i - t_i$ ). Due to network jitter, this delay will vary from packet to packet. Let $d_i$ represent an estimate of the average network delay upon reception of the ith packet. This estimate is determined from the time-stamps as follows:

$$d_i = ( 1 - u ) d_{i-1} + u ( r_i - t_i )$$

Where u = 0.01 is a fixed constant

Thus $d_i$ is a smoothed average of the observed network delays which places more weight on observed network delay values in the past than the currently observed delay value. The parameter u characterizes the memory property of this estimation. Let $v_i$ denote an estimate of the average deviation of the delay from the estimated average delay. This estimate is also constructed from the time-stamps as follows:

$$v_i = ( 1 - u ) v_{i-1} + u \mid r_i - t_i - d_i \mid$$

The estimates $d_i$ and $v_i$ are calculated for every packet received although they are used only to determine the play-out time for the first packet in the talk-spurt. If packet i is the first packet of a talk spurt, its play-out time is calculated as

$$p_i = t_i + d_i + K v_i$$

Where K = 4 is a constant.

The coefficient K controls the delay/packet loss ratio. The larger this constant, the more packets are play-out at the expense of longer delays.

The play-out time for any subsequent packet in the talk-spurt is computed as an offset from the point in time when the first packet in the talk-spurt was played out. The length in time from the when the first packet in the talk-spurt is generated until is played out is given by

$$q_i = p_i - t_i$$

If packet j also belongs to this talk-spurt, its play-out time is given by

$$p_j = t_j + q_i$$

## 5.2 Modified Play-out Algorithm

A closer look at the performance of the above algorithm reveals that two factors are important in determining how close the algorithm operates to the optimum value. The value of the parameter u determines how much weight is given to past values of delay as opposed to the current value. In the original algorithm, this value is set to 0.01. This means that 99% weight is given to observed values from the past while 1% weight is given to the current observed value of network delay. Our experiments have determined that this value of u does not give the best performance. With u = 0.10, the algorithm performs much closer to the optimum buffering delay values. This is substantiated by the results of our simulations.

The second factor that plays a significant role in the algorithm's performance is the initial value of network delay and variance. In the original algorithm, these values are set to 0 i.e. in effect the algorithm makes a 'cold start'. As a result, the algorithm takes a significant amount of time to converge to the actual average of network delay. However, if the algorithm were to start with a value closer to the actual average, then it would perform much better. But it raises the question of estimating this network delay average. Instead of beginning with a value of 0, the network delay can be estimated by measuring the end-to-end network delay during the call set-up phase of VoIP. This value can serve as the starting estimate for the initial network delay. A good estimate of initial value for the network delay variance has been experimentally determined to be a third of the initial network delay. When the algorithm performs with these values, the average buffering delay for the packets is much closer to the optimum value. This is confirmed by the results of our simulations.

# CHAPTER 6

# THE SIMULATION

## 6.1 Simulator Description

The simulator used is a generic event-driven simulator developed in Java. It is a modified version of the one provided by Software Engineering Research Laboratory at the University of Colorado. The environment to be simulated is designed as a collection of entities. Each entity abstracts a component of the environment. It generates events for other entities and processes those that it receives from other entities. Events represent the information that the entities exchange with each other to facilitate communication. The simulator core executes events generated by entities, updates the simulation clock on an event-by-event basis and maintains the internal event queue. It also takes care of creating entities and adding events to the event queue. It executes a time-ordered schedule of discrete events. The event queue contains a list of events to be executed for a particular instance in time.

## 6.2 Simulation Set-up and Parameters

For each set of simulation parameters, the simulation is run for 10, 20, 30 and 60 minutes. In the simulations, the events that are exchanged between entities are represented by packets. A packet is characterized by its sequence number, sender time-stamp, receiver time-stamp, size and type. The size of packet is determined by the rate at which packets are generated e.g. every 20 ms, the number of samples that constitute a single packet and the number of bits per sample. The type of a packet represents whether it is the first, last or intermediate packet of a talk-spurt. In addition, each packet also has a source-id and a destination-id which indicate who has generated this packet and to whom it is sent to.

For simulating a VoIP call, we have designed the following entities:

1. Caller:

The Caller represents the user who is initiating the VoIP call. A voice conversation is simulated by alternating periods of talk-spurts and silence. According to [12], human speech can be modeled as a process that alternates between talk-spurts and silence periods that follow exponential distributions with means of 227 ms and 596 ms respectively. The length of the session is input to the Caller and it alternates between talk-spurts and silence till the session time has not elapsed. When a packet is generated, the Caller creates a packet, determines the size of the packet, enters its sender time-stamp based on the current simulation clock value and sends it to the next entity. The number of samples per packet is set to 8000 with 8 bits per sample. Thus the packet size is determined by the rate at which packets are generated. During simulation runs, the rate at which packets are generated is set to 10 ms, 20 ms and 30 ms.

2. Link:

The Link entity simulates the user's connection to the Internet and is characterized by its link rate. This rate determines the transmission delay from when the packet is sent from the user's machine till it enters the Internet. This connection can either be a dial-up, DSL or cable each having different link rates. The link bandwidth can take values of 54 kbps, 300 kbps and 1.5 Mbps. At the receiver's end, there is a similar Link entity between the Internet and the receiver. The link rate plays a significant role in the end-to-end network delay.

3. Delay:

The Delay entity serves as a black-box representation of the Internet as a whole. The packets generated by the Caller are sent to it and are subject to the delay and loss patterns as observed in the Internet. The packets are then forwarded to the Receiver entity via the corresponding link. The Internet delay is represented by a shifted Gamma distribution [13] with a scale parameter $\alpha = 1$ and shape parameter $\beta = 0.6$. The shift can be either 107.5 ms for long distance calls or 7.5 ms for local calls.

The loss in the Internet is modeled as a discrete-time Markov chain model [14] with two states. In a two-state Markov model, the current state depends only on the previous value. It captures the dependence between consecutive losses. The two parameters, p and q, are the transition probabilities between the two states – the loss state and the loss-free state represented by $X_i = 0$ and $X_i = 1$ respectively.

$$p = P\,[X_i = 1 \mid X_{i-1} = 0]$$

$$q = P\,[X_i = 0 \mid X_{i-1} = 1]$$

The good run length is defined as a number of consecutive packets which are not dropped. Its distribution is given by

$$f(j) = p\,(1 - p)^{j-1} \qquad \text{for } j = 1,2,\ldots\ldots\infty$$

Similarly the loss run length is defined as the number of consecutive packets which are dropped. Its distribution is given by

$$f(j) = q(1-q)^{j-1} \qquad \text{for } j = 1,2,\ldots\ldots\infty$$

The values for p and q are taken from actual Internet traces as given in [14]. The traces were obtained for unicast data from a source located in Amherst, Massachusetts. The chosen traces have the following values for p and q:

Table 6.1: Two-state Markov Chain Model Parameters

| Trace # | p | q |
|---|---|---|
| 1 | 0.0158 | 0.9529 |
| 2 | 0.0109 | 0.7915 |
| 3 | 0.0192 | 0.8454 |

Based on the above values of p and q, the loss model will alternate between sequences of packets which are dropped followed by those which are not dropped. The length of each such sequence is given by distributions mentioned above.

4. Receiver:

The Receiver entity represents the called party of the VoIP call. It consists of two components. The first component contains the implementation of the generic adaptive play-out buffering algorithm. This component named as ‘Callee’ receives packets and stores them in the receiver’s buffer. For each received packet, it enters the receiver time-stamp which is based on the current simulation clock value. This time-stamp along with the one set by the sender is used to perform a continuous estimation of the network delays and compute the play-out times of the first packet in every talk-spurt. However, in the Internet, the clocks on the end-systems are not synchronized and this lack of synchronization needs to be taken into account when estimating the network delays based on sender and receiver time-stamps. This clock skew is represented in the simulations by a standard normal distribution and the values derived from it are subtracted from the estimates of the network delays in the algorithm to account for the unsynchronized clocks at the sender and receiver in the Internet. The arrival time of each received packet is recorded by this component which is then used to compute the optimum buffering delay for the VoIP call session.

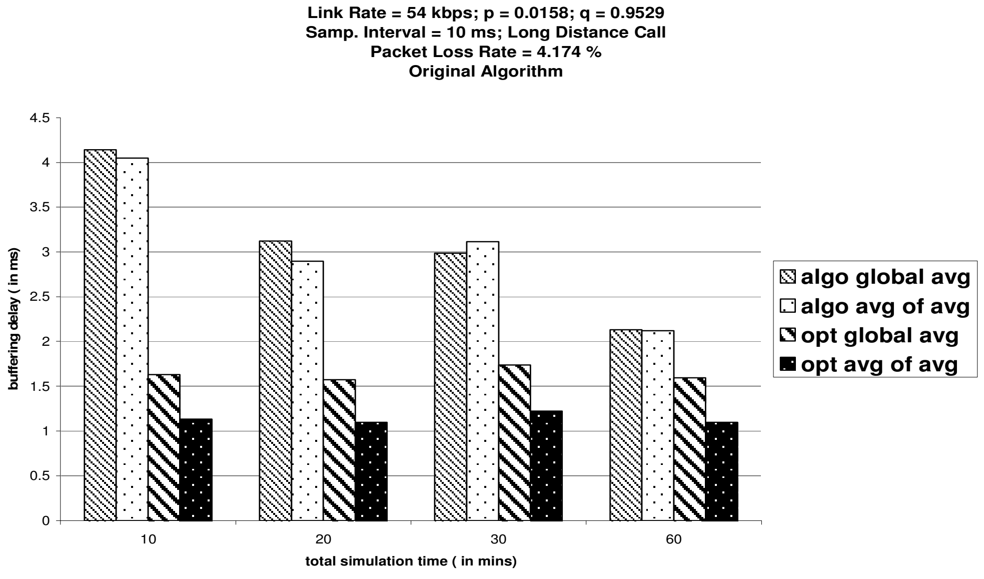
The second component called ‘Player’ is responsible for the play-out of the received packets. When the first packet of the talk-spurt is received, the Callee informs the Player about this event and also indicates the play-out time of that packet. At that instance in time, the Player checks the buffer to see if the packet has been received. If it is found in the buffer, it is recorded to have been played. The Player computes it buffering delay and stores this value which is utilized to compute the buffering delay of the algorithm. For subsequent packets in the talk-spurt, the Player continuously checks the buffer after an interval of d time units which is equal to the interval after which packets were generated at the sender. If the packet is not found in the buffer at its play-out time then it is considered to have missed its play-out point even if it actually arrives later. The packets that are dropped by the network never make it to the receiver and are also considered to have been missed. The Receiver entity records the number of packets that have missed their play-out times because they arrived late and those that have been dropped by the network. These are used in the optimal buffering delay computations to keep the packet loss rates constant in both calculations. The Player stops checking the buffer for subsequent packets in the talk-spurt when it plays-out the last packet of that talk-spurt. In case, the last packet misses its play-out time, it stops checking when the packet is eventually received.

## 6.3 Simulation Results

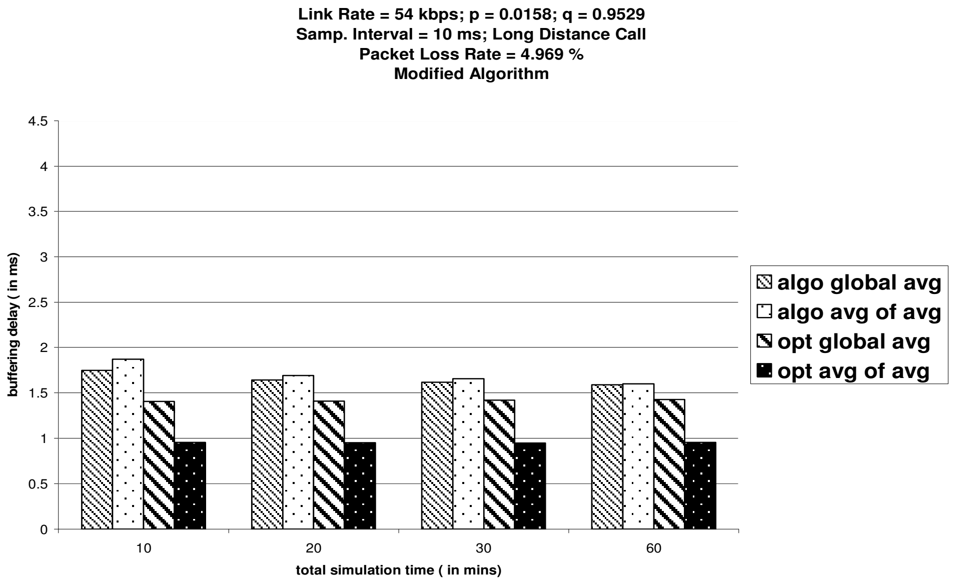
During the simulation runs, different sets of values for the input parameters are used. For each set of values, the average buffering delay across all talk-spurts and the average of the average buffering delay of each talk-spurt is computed. The averages are computed for more than one run of the simulation in order to exclude erroneous data. For each scenario, the buffering delay values of the original and modified algorithms are compared with their respective optimum values through graphs.

For constant values of link rate, loss conditions, and packet generation rate and whether the call is a local or long distance call, the buffering delay values are plotted for different values of total simulation time viz. 10, 20, 30 and 60 minutes.

The figures below present the buffering delays for the original and modified algorithm when the link rate is 54 kbps, the Markov model transition probabilities are p = 0.0158 and q = 0.9529, the packet generation rate called the Sampling Interval is 10 ms and when a long distance call is placed.



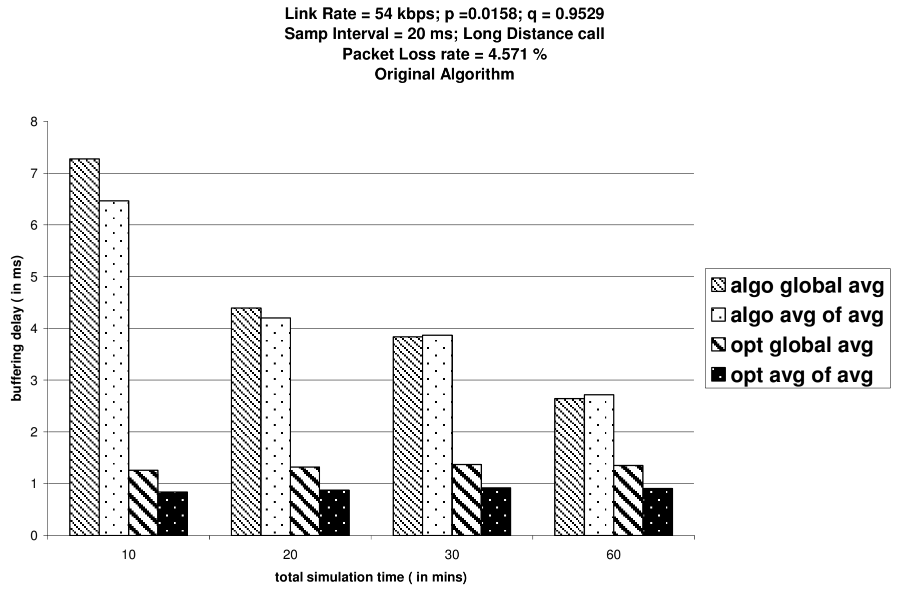
(a)



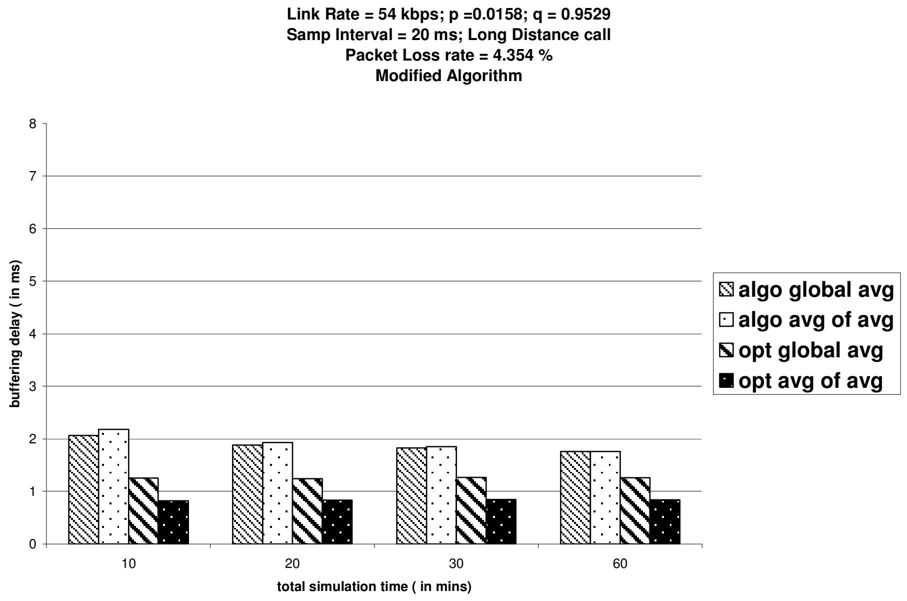
(b)

Figure 6.1: Buffering Delays of (a) Original & (b) Modified Play-out Algorithm in Scenario 1

As the above two figures – Figure 6.1(a) and (b) show, the modified algorithm performs better than the original algorithm for the same scenario. The average buffering delay across all talk-spurts for the modified algorithm is about 1.7 ms while that for the original algorithm is about 4.1 ms. Thus the modified algorithm operates much nearer to the optimum value which is about 1.5 ms. The packet loss rates of the two algorithms are 4.174 % and 4.969 % in the given scenario which are comparable. The remaining graphs in this section represent this comparison for a number of different scenarios and in each case, the modified algorithm performs better than the original one.



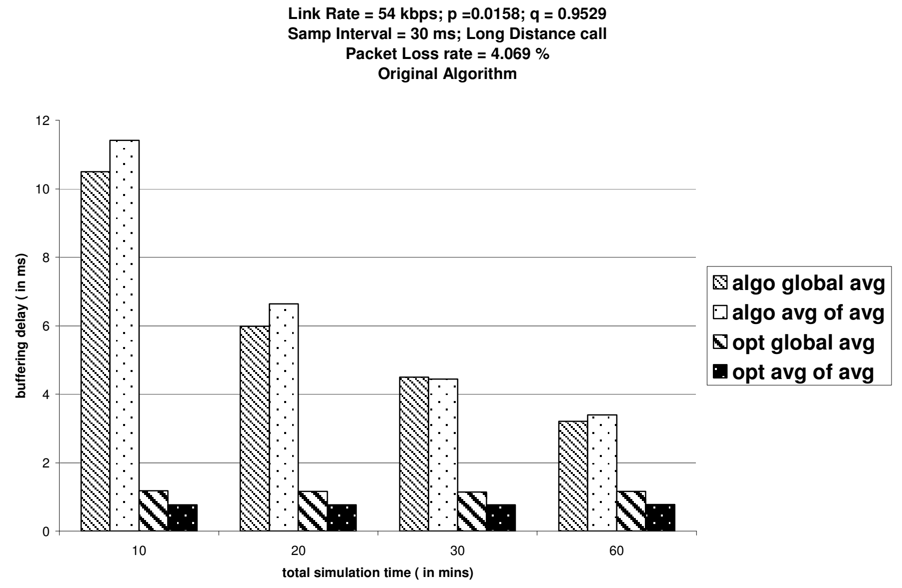
(a)



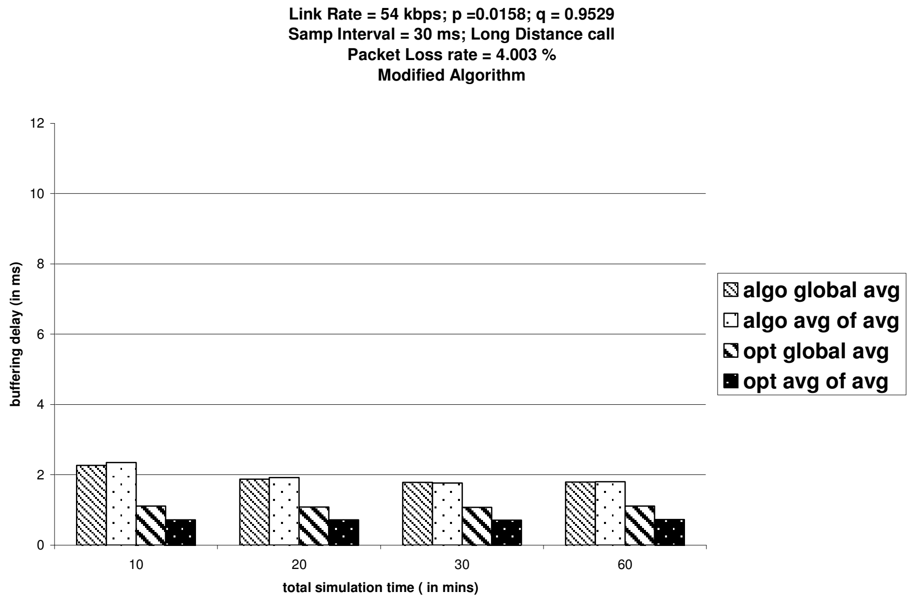
(b)

Figure 6.2: Buffering Delays of (a) Original & (b) Modified Play-out Algorithm in Scenario 2

Figures 6.2 (a) & (b) give the comparison when the input parameters are same as in scenario 1 except that the sampling interval is 20 ms.



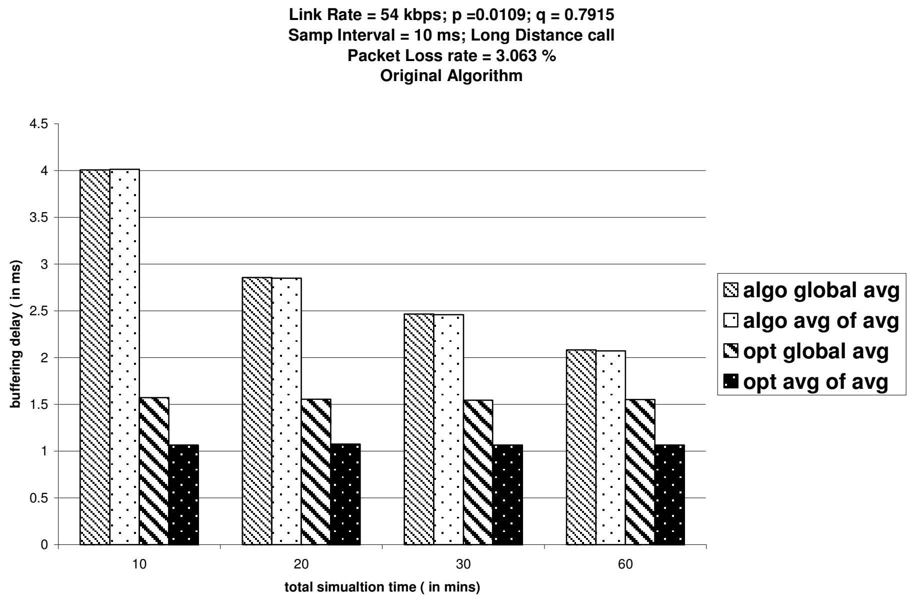
(a)



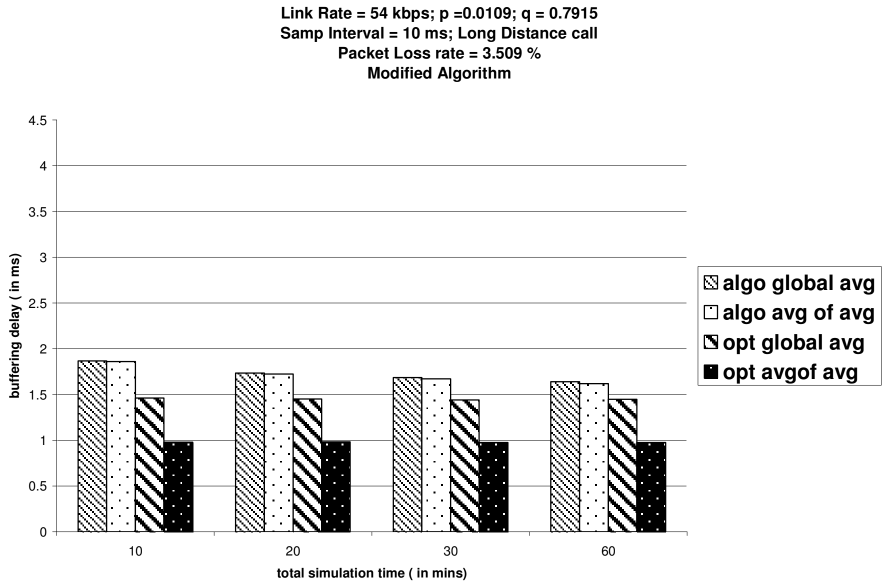
(b)

Figure 6.3: Buffering Delays of (a) Original& (b) Modified Play-out Algorithm in Scenario 3

The above figures 6.3(a) & (b) show that modified algorithm operates much closer to the optimum value when the input parameters are same as in scenario 1 except that the sampling interval is 30 ms.



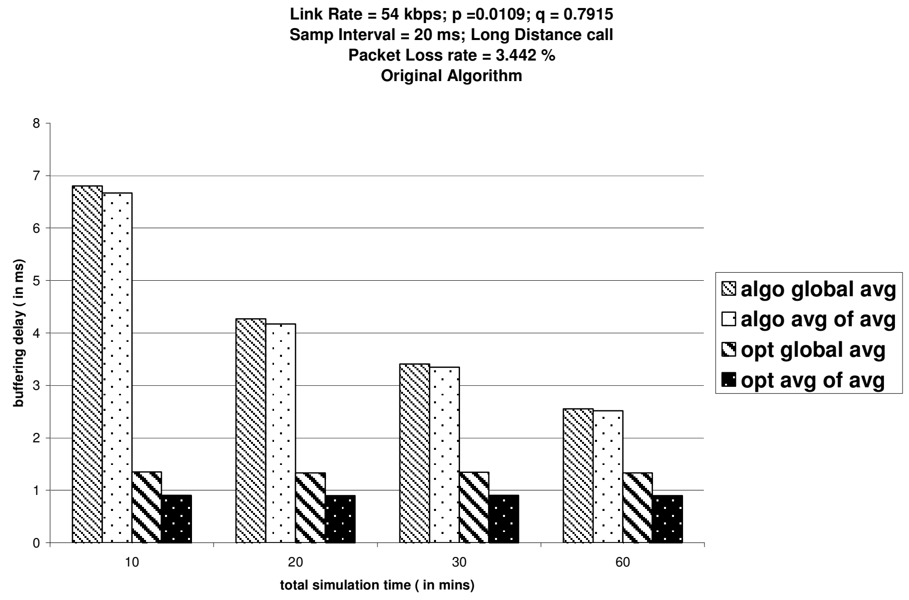
(a)



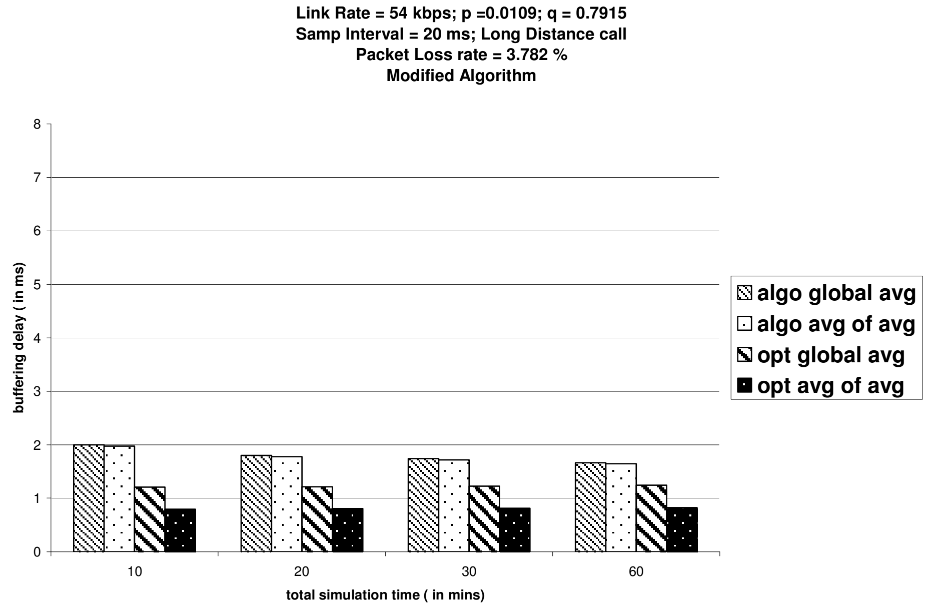
(b)

Figure 6.4: Buffering Delays of (a) Original & (b) Modified Play-out Algorithm in Scenario 4

The above figures 6.4(a) & (b) show performance of the algorithms when the second set of Markov model transition probabilities are input to the simulation and the sampling interval is set to 10 ms.



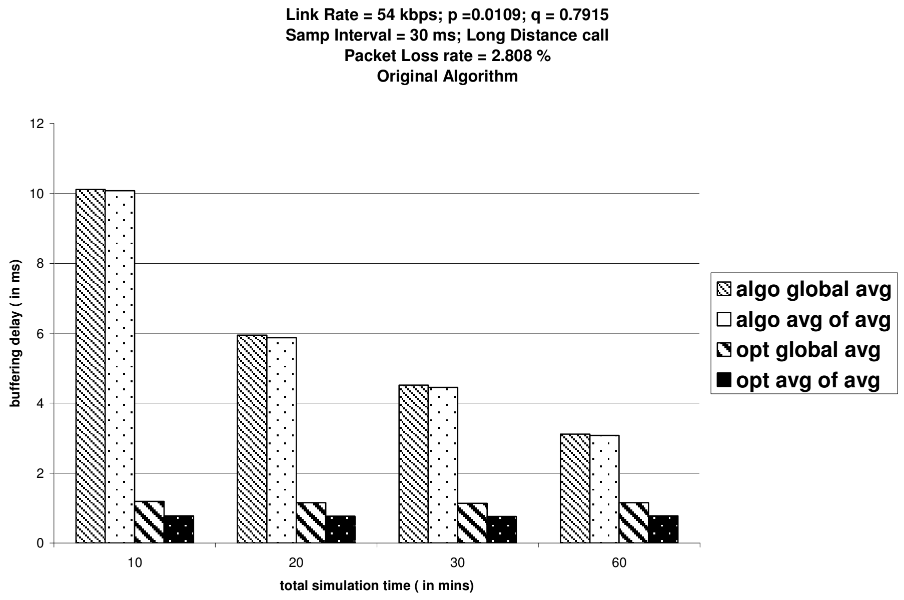
(a)



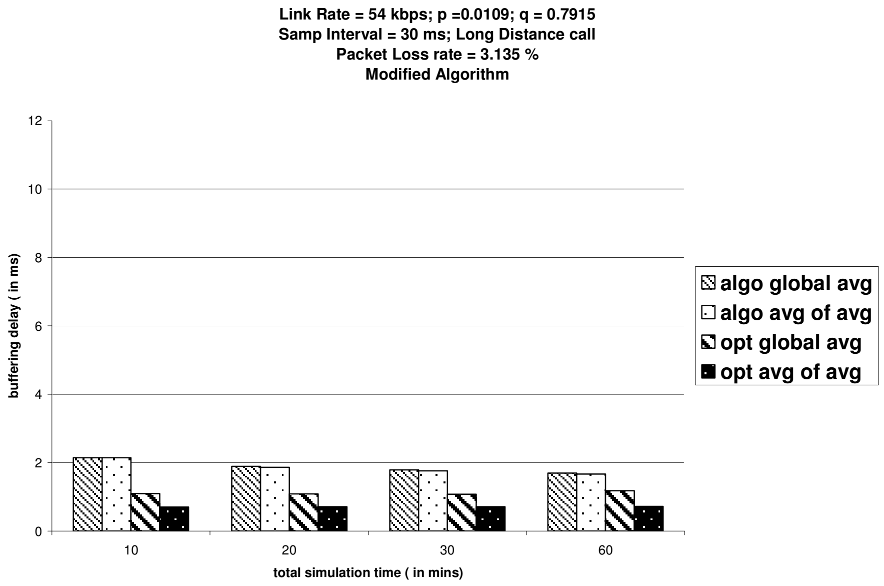
(b)

Figure 6.5: Buffering Delays of (a) Original & (b) Modified Play-out Algorithm in Scenario 5

Figures 6.5(a) & (b) compare performance of the algorithms when the input parameters to the simulation are same as in scenario 4 while the sampling interval is increased to 20 ms. In this case as well, the modified algorithm performs better.



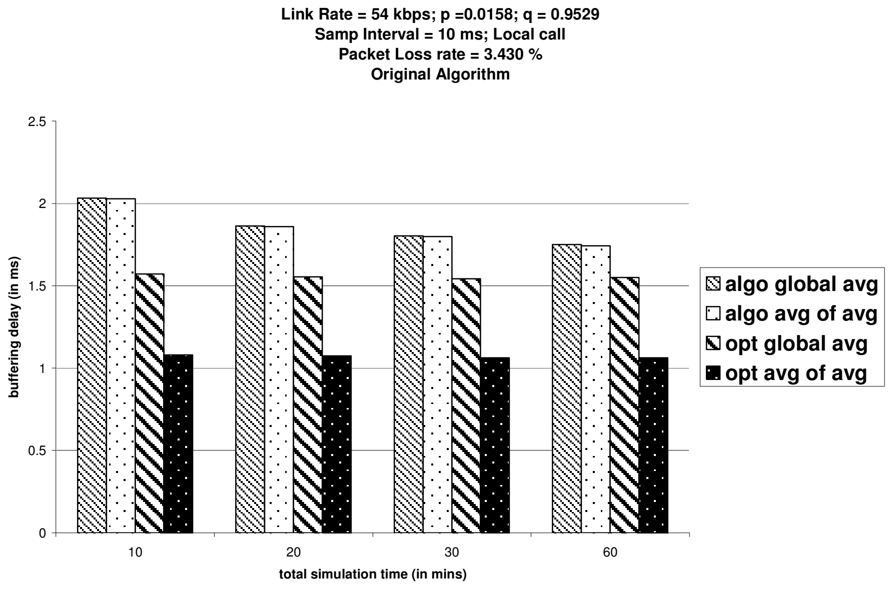
(a)



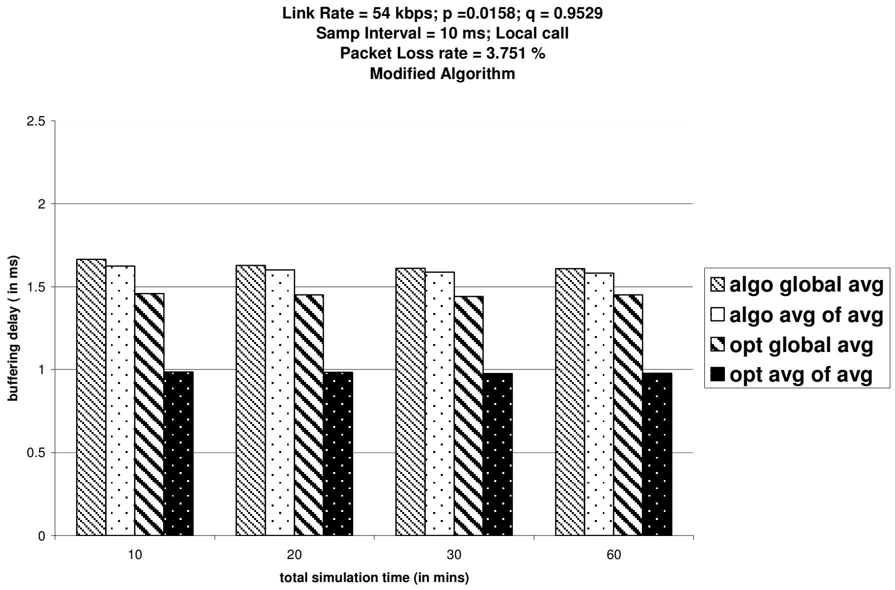
(b)

Figure 6.6: Buffering Delays of (a) Original & (b) Modified Play-out Algorithm in Scenario 6

As the above figures 6.6(a) & (b) show, the modified algorithm shows an improvement of about an order of magnitude over the original algorithm in scenario 6.



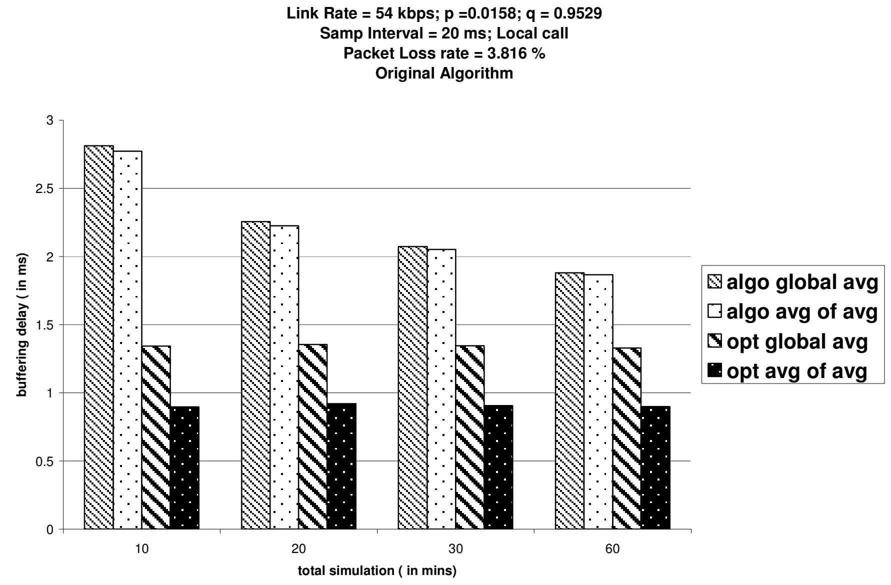
(a)



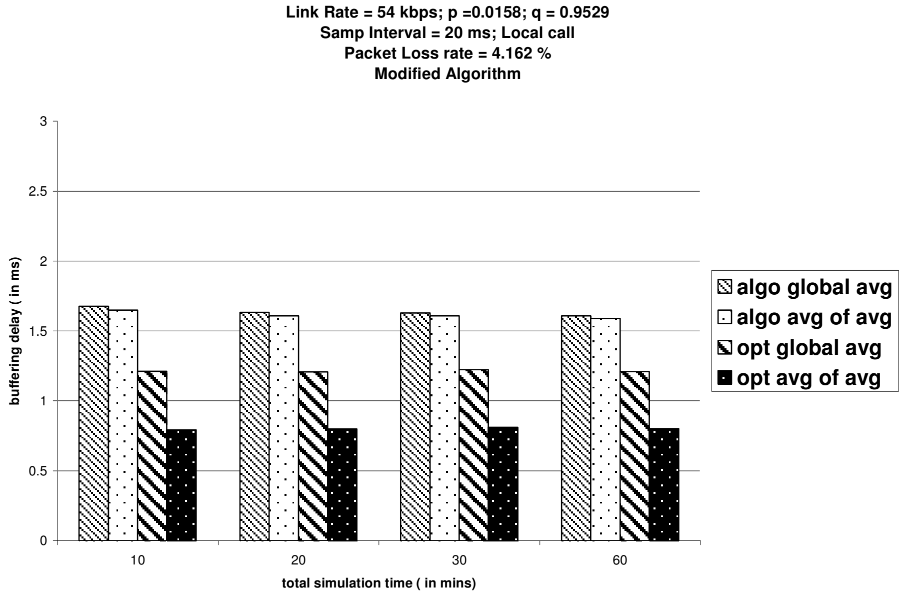
(b)

Figure 6.7: Buffering Delays of (a) Original & (b) Modified Play-out Algorithm in Scenario 7

Figures 6.7(a) & (b) compare performance of the algorithms when the input parameters to the simulation are same as in scenario 1 except that a local call is placed which means that the average network delay is 7.5 ms.



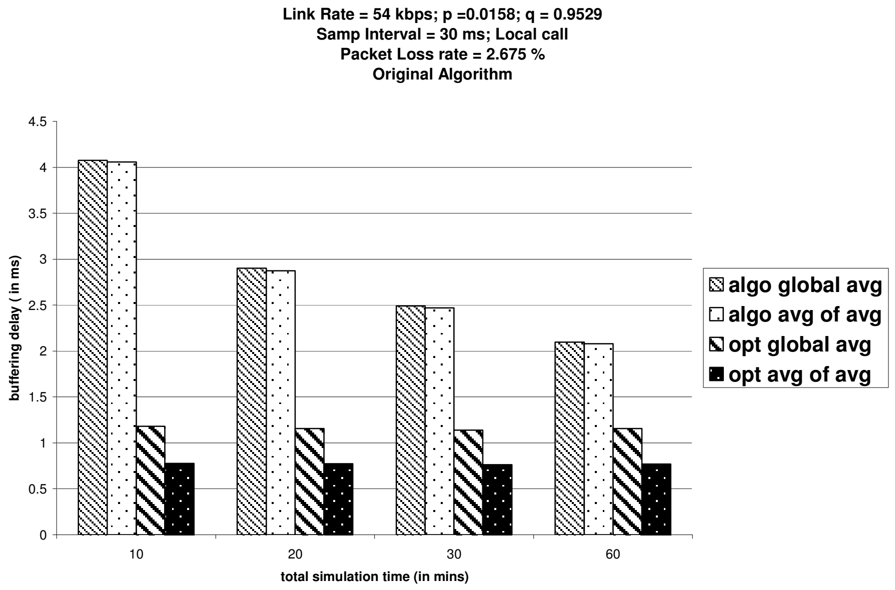
(a)



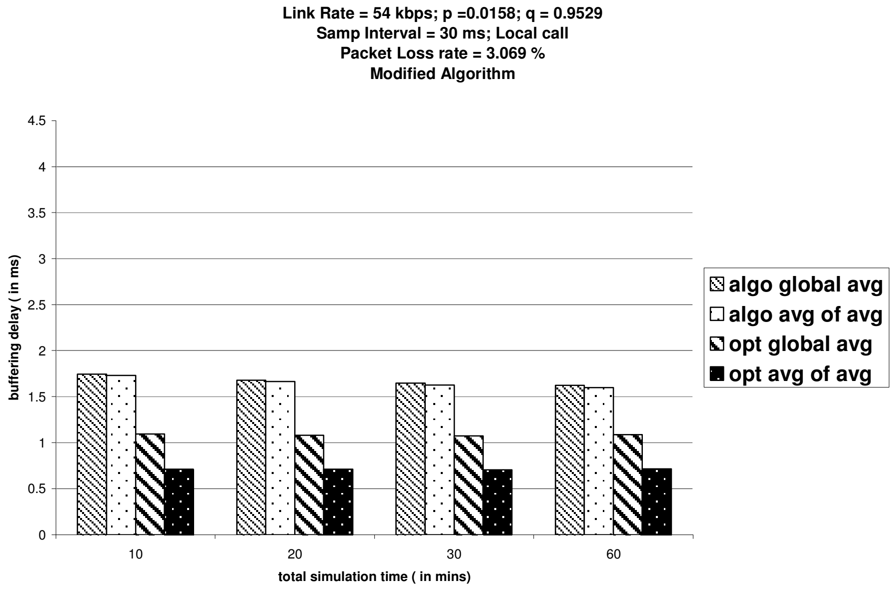
(b)

Figure 6.8: Buffering Delays of (a) Original & (b) Modified Play-out Algorithm in Scenario 8

Figures 6.8(a) & (b) compare performance of the algorithms when the input parameters are same as in scenario 7 while packets are generated every 20 ms.



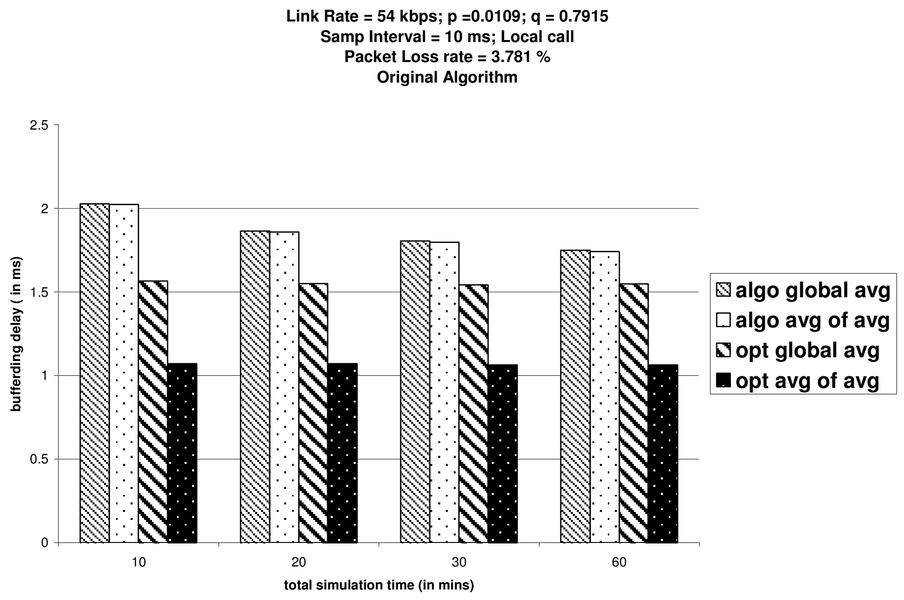
(a)



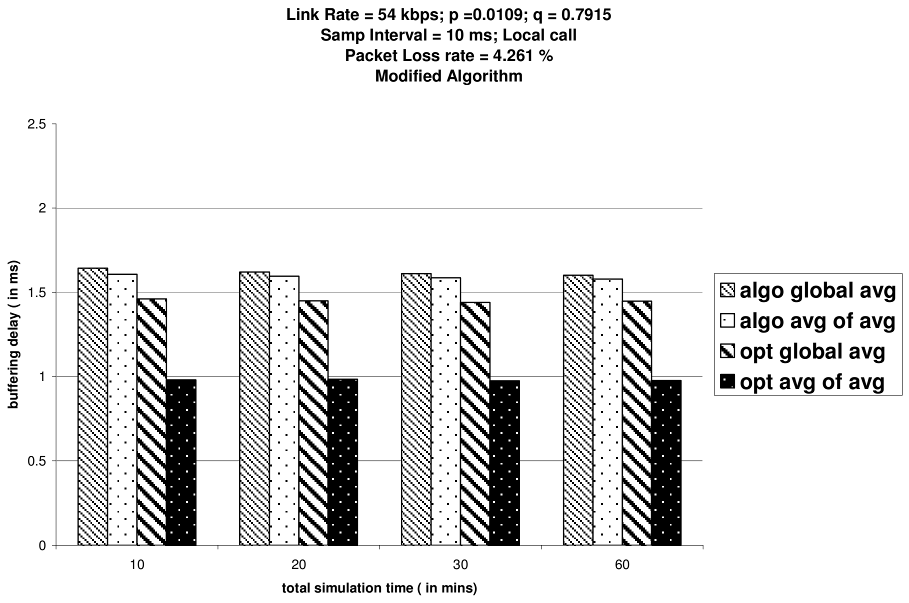
(b)

Figure 6.9: Buffering Delays of (a) Original & (b) Modified Play-out Algorithm in Scenario 9

Figures 6.9(a) & (b) demonstrate that in scenario 9, the modified algorithm is closer to the optimum value than the original version.



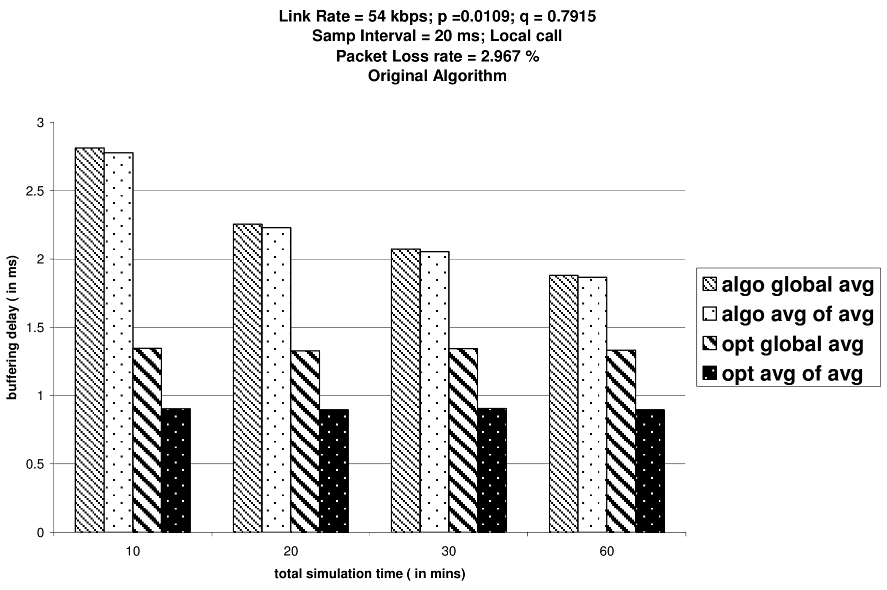
(a)



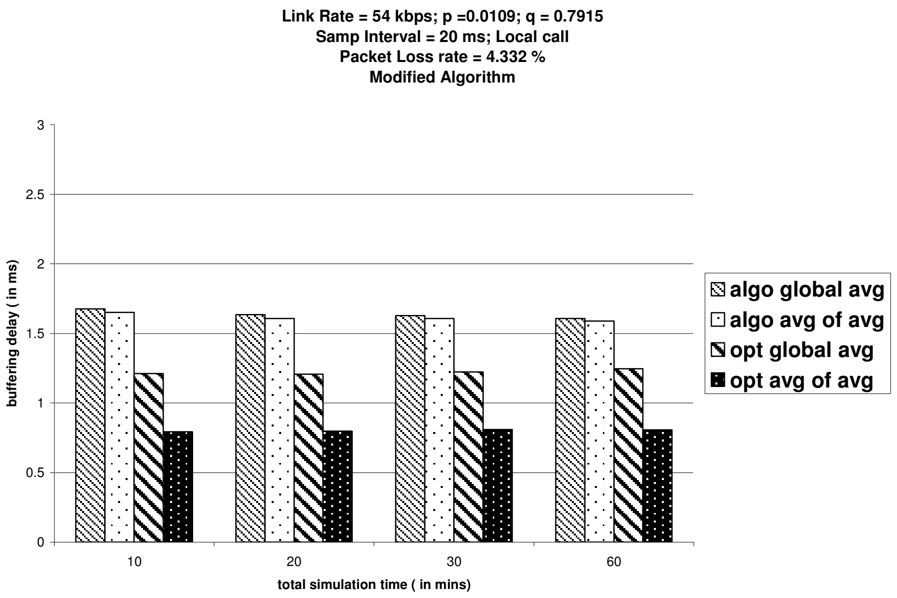
(b)

Figure 6.10: Buffering Delays of (a) Original & (b) Modified Play-out Algorithm in Scenario 10

Figures 6.10(a) & (b) show that when the input parameters are as that given in scenario 10, the modified algorithm performs better than the original algorithm.



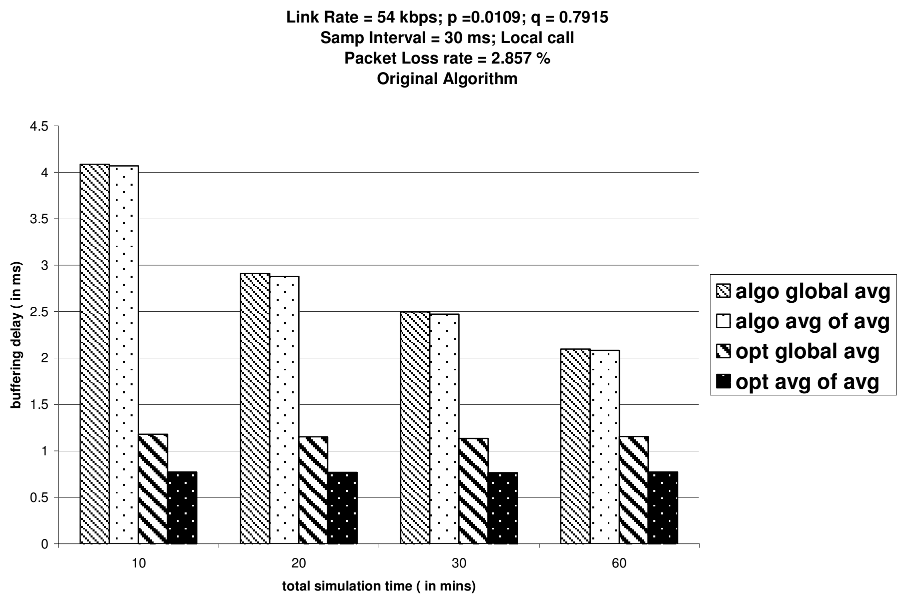
(a)



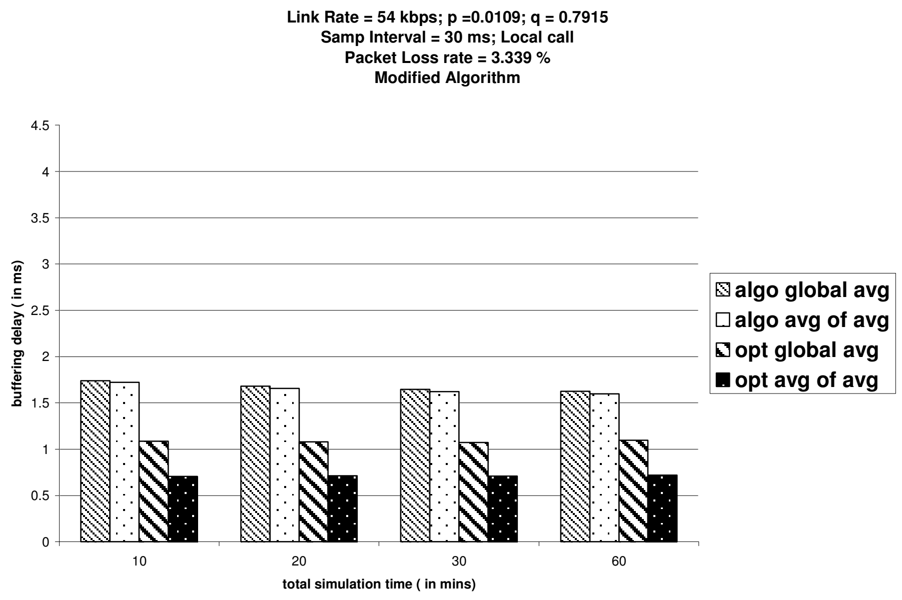
(b)

Figure 6.11: Buffering Delays of (a) Original & (b) Modified Play-out Algorithm in Scenario 11

Figures 6.11(a) & (b) demonstrate that when the input parameters are as that given in scenario 11 with sampling interval of 20 ms, the modified algorithm shows an improvement over the original algorithm.



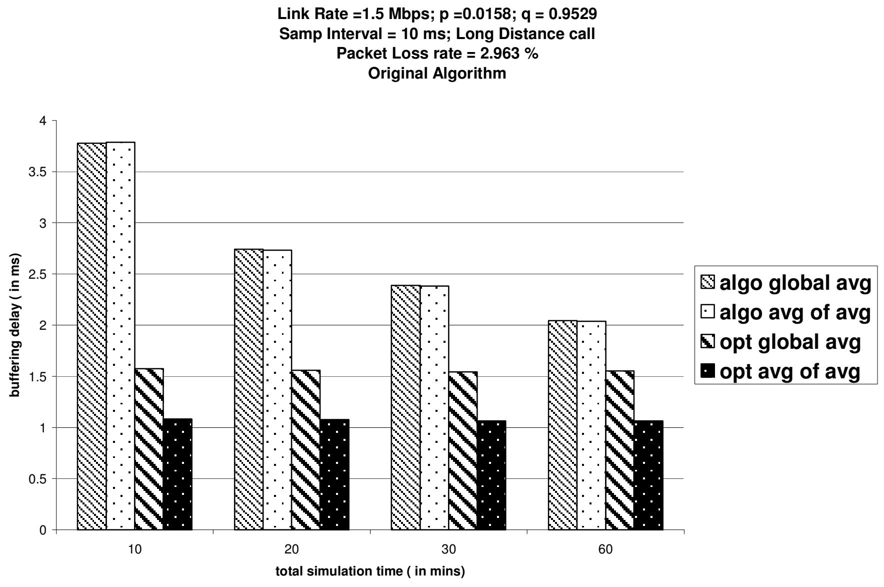
(a)



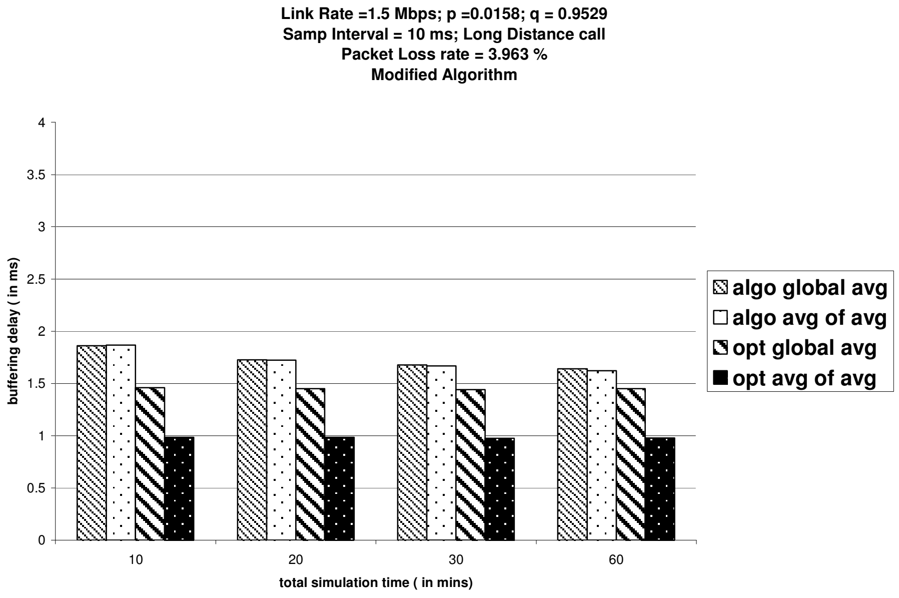
(b)

Figure 6.12: Buffering Delays of (a) Original & (b) Modified Play-out Algorithm in Scenario 12

Figures 6.12 (a) & (b) shows that the modified algorithm operates at lower buffering delay values which are closer to the optimum values than the original one.



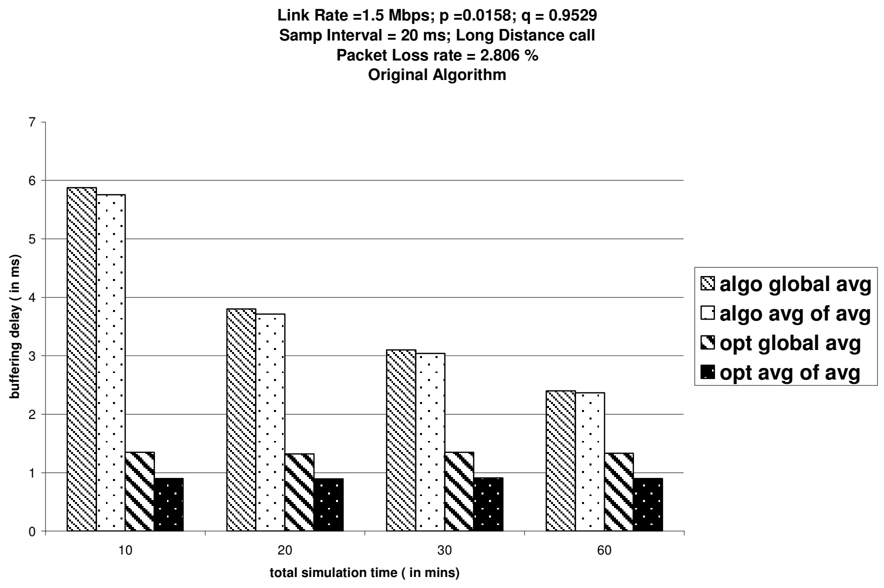
(a)



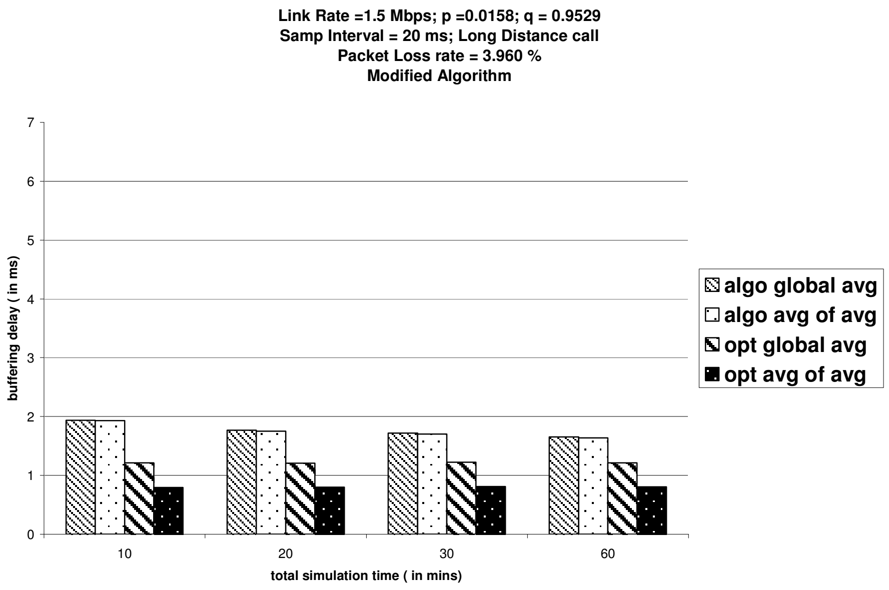
(b)

Figure 6.13: Buffering Delays of (a) Original & (b) Modified Play-out Algorithm in Scenario 13

Figures 6.13 (a) & (b) show the performances of the original and modified algorithms in scenario 13 where the link rate at both ends are increased to 1.5 Mbps.



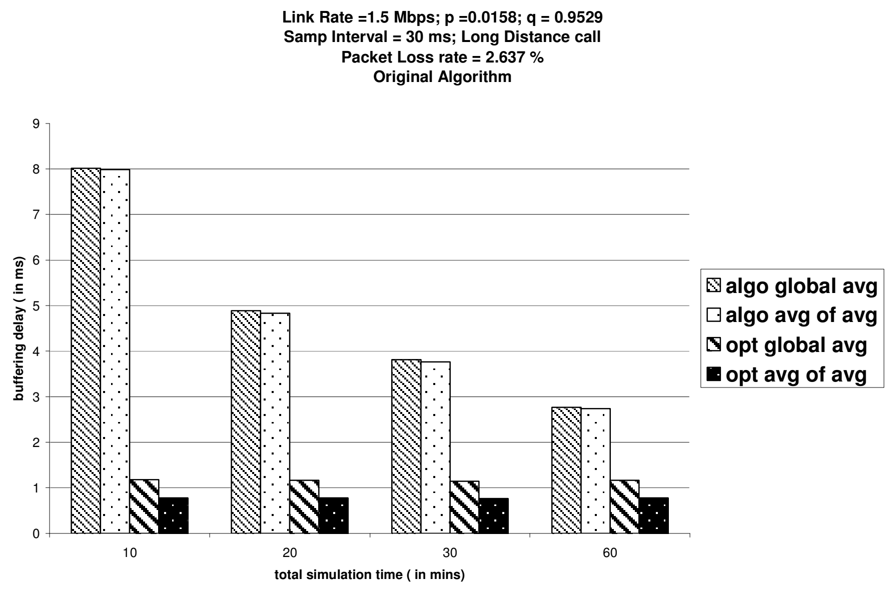
(a)



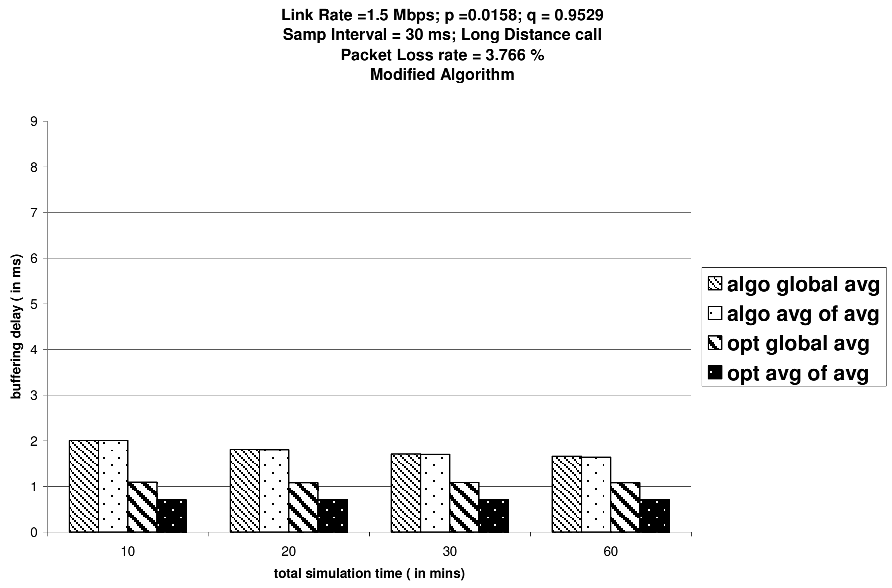
(b)

Figure 6.14: Buffering Delays of (a) Original & (b) Modified Play-out Algorithm in Scenario 14

Figures 6.14 (a) & (b) show the performances of the original and modified algorithms in scenario 14. The modified algorithm significantly lowers the buffering delay values as compared to those provided by the original algorithm.



(a)



(b)

Figure 6.15: Buffering Delays of (a) Original & (b) Modified Play-out Algorithm in Scenario 15

Figures 6.15 (a) & (b) show the performances of the original and modified algorithms in scenario 15. Here, the global buffering delay for the modified algorithm is about 2 ms while that of the original algorithm is 8 ms.

# CHAPTER 7

# CONCLUSION

In this thesis, we simulated a VoIP call over the Internet in different scenarios. The sender generates packets for the receiver and transmits them over the Internet. The voice packets are subjected to delay and loss patterns representative of those experienced by packets in the Internet. The receiver employs a generic adaptive play-out algorithm that estimates the network delay for the next talk-spurt based on the delays experienced by packets it has received. The buffering delay of the packets at the receiver as a result of the algorithm is computed. The optimum buffering delay for the VoIP call session for the same packet loss rate is determined. The algorithm with the proposed modifications is then subjected to the same simulation environment and its performance is compared to the original algorithm. The simulation results demonstrate that the modified algorithm performs closer to the optimum value than the original algorithm in all scenarios. Hence the reduced buffering delay introduced by the modified algorithm will lead to an overall reduction in the end-to-end network delay. Since network delay plays a significant role in determining the quality of a time-sensitive application such as VoIP, the modified algorithm at the receiver will improve the call quality as perceived by the end-user.

## REFERENCES

[1] Rajendran R., Ganguly S., Izmailov R. & Rubenstein, D. (2006). Performance Optimization of VoIP using an Overlay Network.

[2] Rosenberg, J., Lili Qiu, & Schulzrinne, H. (2000). Integrating packet FEC into adaptive voice playout buffer algorithms on the Internet. INFOCOM Page(s):1705 - 1714 vol.3.

[3] Hassan M. & Nayandoro A. (2000). Internet Telephony: Services, Technical Challenges and Products. IEEE Communications Magazine.

[4] www.internettrafficreport.com

[5] Valdes R. How VoIP works. http://computer.howstuffworks.com/ip-telephony1.htm

[6] Internet Telephony. http://www.homepagez.com/chaffee009/ECE494/intro.htm

[7] International Engineering Consortium (IEC). Voice over Internet Protocol http://www.iec.org/online/tutorials/int_tele/topic01.html

[8] Black U. (1999).VOICE OVER IP. First Edition. Prentice Hall series in Advanced Communication Technologies. ISBN#: 0130224634

[9] Vlaovic, B. & Brezocnik, Z. (2001). Packet based telephony. EUROCON'2001, Trends in Communications, International Conference on. Volume 1, 4-7 Page(s):210 - 213 vol.1

[10] DeLeon P. & Sreenan C. J. (1999). An Adaptive Predictor for Media Play-out Buffering. IEEE Conf on Acoustics, Speech and Signal Processing (ICASSP), vol. 6, pp. 3097—3100

[11] Ramjee R., Kurose J., Towsley D., & Schulzrinne, H. (1994). Adaptive play-out mechanisms for Packetized Audio Applications in Wide-Area Networks. Proceedings of IEEE INFOCOM '94.

[12] ITU-T Recommendation P.59. (1993) Telephone Transmission Quality Objective Measuring Apparatus: Artificial Conversational Speech.

[13] Corlett A., D.I. Pullin D.I., & Sargood S. (2002). Statistics of One-Way Internet Packet Delays. 53rd IETF, Minneapolis.

[14] Yajnik M., Moon S., Kurose J. & Towsley D. (1999). Measurement and Modeling of the Temporal Dependence in Packet Loss; IEEE INFOCOM '99

## BIOGRAPHICAL INFORMATION

Ruchir Pramod Shende received his Bachelor of Engineering degree in Computers from Veermata Jijabai Technological Institute, Mumbai, India in 2004. He joined the University of Texas at Arlington as a Master's student in Computer Science in August 2004. He received his Master of Science degree in Computer Science from the University of Texas at Arlington in 2006.