Université d'Ottawa · University of Ottawa

Jitter Buffer Management Algorithms for Voice Communication

by
Peng Frank Zhang

A thesis submitted to
the School of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

School of Information Technology and Engineering
University of Ottawa

July 2002

Acknowledgments

I would like to express my sincerely thanks to my supervisor, Professor Oliver W. W. Yang, for his continuous encouragement, professional competence and patience. I would also like to thank Mr. Hao Zhang, for his contribution of the early investigation and coding work on the LMS. This work has been partially supported by a research grant from Mitel Corporation.

For my wife and parents.

Abstract

This thesis studies some jitter management algorithms for real-time applications. These algorithms are executed at a destination node, and assume no knowledge of the source characteristic or the impact of the network path characteristic. The work mainly focuses on prediction algorithms that make use of the information of the packets received in the past, and adjust buffer parameters in order to maintain certain level of quality of service (QoS). Two algorithms are proposed, first, to apply the least mean square method to predict the future packet interarrival time so that the buffer parameters can be dynamically changed in order to adapt to the bursty network traffic; second, to apply the fuzzy logic method on the buffer management to maintain the gap probability within acceptable level while keep the latency as low as possible. These two new algorithms have been evaluated using OPNET simulation and compared with some other algorithms such as the I-policy and the E-policy. We studied and discussed the tradeoff among the gap probability, average display latency, and packet loss probability. Towards the end, we have also made some design recommendations.

Keywords: jitter buffer management, prediction, fuzzy logic, performance evaluation, display latency.

Table of Contents

Acronyms

		Section of the First Appearance
ATM	Asynchronous Transfer Mode	1.1
FIFO	First In First Out	1.1
FLA	Fuzzy Logic Algorithm	5.0
FTP	File Transfer Protocol	1.0
IETF	Internet Engineering Task Force	3.9
IPPM	IP Performance Metrics	3.9
ITU	International Telecommunication Union	1.2
LMS	Least Mean Square	1.3
QoS	Quality of Service	1.1
RSS	Root Summed Square	5.2.3

Symbols and Notations

	Meaning	*Section of the First Appearance*
d	Predicted next packet's inter-arrival time	4.2.2
e(n)	The sequence of the difference between observed value and the estimated value	4.1
p	Number of the previous packet that involved in a prediction	4.1
w(n)	Prediction coefficients	4.1
$x(n)$	The observed values of the interarrival time of the packets	4.1
$y(n)$	The predicted values of the interarrival time of the packets	4.1
$\hat{B}$	Difference between target gap probability and measured gap probability	5.1
B_0	Initial buffer time	4.23
D_0	Inter-arrival time threshold	4.22
G_{prob}	Gap probability	3.4.1
K	The width of "No changing" of the "Buffer size error rate" membership function	5.1
L	Average latency	3.5
L_0	Initial buffer size threshold	4.2.3
L_{prob}	Loss probability	3.6
L(Xi)	Fuzzy set	5.1
M	The width of "OK" of the "Buffer size error " membership function	5.1
N	Packet number of one session	5.3
Q	Current queue size	4.2.2
Q_0	The queue threshold of LMS Algorithm	4.3
R	Membership degree	5.2.3

S_1	The width of "Negative" of the output function	5.1
S_2	The width of "Positive" of the output function	5.1
T	Simulation time	4.3
β	Buffer change factor	5.2.3
δ	The degree of the "Buffer Size Error" membership function	5.1.1
η	The degree of the "Buffer Size Error Rate" membership function	5.1.2
σ	The degree of the "Initial Buffer Size Adjustment" membership function	5.1.3
μ	Step-size or learning rate in the Least Mean Square Algorithm	4.3

List of Figures

List of Tables

Chapter 1

Introduction

Voice and real-time applications have several advantages when implemented on packet-switched networks. Conventionally, they are handled very well in circuit-switched networks. The call is set up first, and the required bandwidth is definitely reserved during the call to ensure the toll quality. The delay and delay variability (known as jitter) are bounded and can be predicted when under control. However, this approach is inefficient and expensive. In comparison, packet-switched networks offer great features. First, data compression can significantly reduce the bandwidth of voice transmission. The silence-suppression can also avoid transmitting samples during silences, and reduce the required bandwidth down to 60%. Second, packet-switched networks have a software-based architecture that eases the development of new services and service configuration and upgrade. Third, packet-switched networks not only bring more efficiency to but also reduce the cost of the enterprises, by converging voice and real-time application with data networks. Thus, enterprises can avoid leasing extra lines, reduce long-distance call, etc. Definitely, due to its capability to efficiently utilize the exiting bandwidth to enhance more unique features, voice over packet networks is proved to be promising and widely accepted in telecommunications market.

Voice and other real-time applications on the networks are time critical. Unfortunately the packet-switched network was originally created for data transmission, not for real-time application. Several challenges must be overcome. Packet-switched networks are packet-centered; each packet is sent from the source end, travels through the

network and reaches its destination. Data packets share the network, including the switches and routers and buffers, etc. The more the packets in a network, i.e. as the load of a network increases, the more contention each packet will encounter with the limited network resources, such as router speed, link capacity, buffer size, etc. As a result, packets may experience large delay and jitter. Packets may take different routes to the destination, and this may cause the packets received at the destination to be out of sequence, or even to be lost during transmission. Some applications, like FTP and E-mail services, can tolerate considerable levels of delay and jitter. They are not time sensitive. They can wait until the destination end receives everything correctly. If errors occur during transmission, they can manage to re-transmit. Voice communications and videoconference, on the other hand, are very sensitive on time. Packets are generated and sent at a scheduled time. When they are received, the original scheduled rhythm must be recovered and displayed. Any extra delay and jitter beyond the acceptable limit will degrade the quality of service.

Jitter is an important factor that has an impact on the quality of the real-time application. Ideally, each voice packet is sent at fixed intervals (for simplicity). They travel trough the same network path, experience the same network delay, and upon reaching the destination. They are immediately being displayed at the original fixed interval. Hence the jitter is zero in an ideal case. However, as a packet network is shared, data packets, and even voice packets from other traffic streams, may compete the same transmission medium, router, switch, etc. As these resources are busy or occupied, packets have to wait in the buffer until the resources are free. The waiting time is often referred as queuing delay. This queuing delay, as the network becomes complicated, will

tend to be random and be difficult to predict. Also the CPU at the source and destination may be busy on other applications, so the voice packet compression and decompression time may be different for each packet. Overall the network environment brings the uncertainty to the voice transmission. When voice or video packets are displayed at the destination end, and even if a packet is late for its due time, a "gap" will occur, result in loss of speech intelligibility.

As jitter is an important factor to the real-time application, the literature is abundant with work on the study of jitter and its control. Existing work on combating jitter is provided below along with other reviews.

1.1. Literature Review

Jitter, defined as the variation of delay, affects the quality of voice. There have been different definitions of jitter: i) Instantaneous jitter [DeCh00], which is the absolute value of the difference between the transfer delays of two adjacent packets. One variation [FuLi98] is without the absolute value, with negative jitter representing a clustering of packets and positive jitter representing a dispersion of packets. ii) Average jitter [ScCa96], which is defined as a function of its previous sample value and the difference of two packets' transfer delays. iii) Peak-to-peak jitter [DeCh00], which is the maximum transfer delay minus the minimum transfer delay during the test period. This is the ITU definition [DeCh00]. iv) Variance jitter [LaYa00], which is the variance of the transfer delays.

Jitter may indicate that packets may arrive late. According to some algorithms [NaKl82], if the late packets arrive later than a tolerant limit, they should be considered

as lost and discarded. So the playout stream will have gaps. Some other algorithms have patience to wait for the late packets, but there are still gaps during waiting. This is because voice has a nature to contain redundant information, and the small gaps may not influence the intelligibility of the voice as perceived by the audience. Certainly the quality of voice will degrade if too much gaps exist. The paper [ClTa99] studied on the effects of jitter on the perceptual quality of video, and it showed that jitter could degrade perceptual quality nearly as much as packet loss did.

The jitter is mainly caused by the random delay occurred in the end-to-end delay. The main factor is the queuing delay. Other factors like encoding delay, packetization delay, processing delay can be neglected [KaTo01]. The literature is abundant with work on the study of jitter and its controls. In general, they can be classified into three base categories:

1. **Source-based approaches:**

   These are usually used to avoid congestions, long end-to-end delays and large jitter. It was recognized [HuPe00] that there was a time delay between the source and the controlled target buffer in the ATM networks. A prediction method is used to predict the ABR buffer future status, and control information is then fed back to the source in order to change the sending rate. So the high link utilization could be maintained, while keeping a fair resource allocation. The approach in [CaCr96] allows the source end to select paths and servers based on the network feedback information.

2. **Network-based approaches:**

   These approaches are used to minimize delay jitter by reserving enough resources

in the network nodes so as (i) to provide guaranteed bounds on delay and delay jitter, (ii) to apply a jitter recovery algorithm on video and voice streams, or (iii) to reserve adequate but separate buffers for each stream at intermediate nodes [ElEl96, ShYa99].

There is another approach [BuVl01] discussed under Differentiated Services networks. Intermediate nodes would use the Weighted Fair Queuing method to ensure the QoS. All voice packets are placed in a single buffer dedicated to the voice traffic. The output rate of the voice buffer was related to the weight in order to bound the queuing delays.

When the voice traffic shares a link with other traffic, non-preemptive priority queues can be used at the intermediate nodes, to gave the voice packets the highest priority [KaTo01]. The authors concluded that this type of priority queue led to the best compromise between the bandwidth utilization and the minimization of delay.

3. **Destination-based approaches:**

To reduce the amount gap during playout, a common but simple operation is to use a playout buffer, i.e. all the coming packets are buffered at the receiver for a while before playing out. However this also introduces an extra buffer delay, called the display latency. Obviously, there is a tradeoff in quality of service, mainly between gap probability and display latency. In general, the longer the packets are buffered, the less the gap probability, but the larger the display latency and the longer the end-to-end delay which beyond certain point becomes unacceptable in real time application.

There are two attractive algorithms [NaKl82] for adjusting the tradeoff: the I-Policy and the E-Policy. In the I-policy, a receiver starts to display at fixed intervals after initial buffering. If a packet does not arrive, a gap results. Even when the late packet arrives later on, it will be discarded *(Ignored)*. On the other hand, the receiver using the E-policy will wait for the late packet, and then continue to display packets. All packets after the late arrival will be delayed accordingly *(Timing is Extended)*. Note that the I–Policy and the E-Policy fall into the extreme ends of the tradeoff. Generally, the I-Policy has a larger gap probability, but a smaller display latency, whereas this is reversed in the E-Policy. There is another algorithm called Q-monitoring algorithm [StJe95], which falls between the I-Policy and the E-Policy. The queue size is monitored when a pre-defined threshold is reached, and the buffer will start discarding the oldest packet in order to prevent the display latency from being too large. The control mechanisms of the I-Policy, E-Policy and Q-monitoring are static. All parameters are set and fixed during operation.

Beyond jitter controls, there have been many methods to control various aspects of traffic. For example, a dynamic bandwidth allocation strategy [Adas98] had used Least Mean Square method to predict the bandwidth requirement of the future frame. The simulation results showed that by using this method, the queue size could be reduced, and the network utilization could be increased between 190%-300%. For real-time VBR traffic [ChSa98], a scheme using Least Mean Square method is presented. By predicting the network traffic, this scheme could reduce the frequency of the bandwidth reallocations and could increase the network utilization. The work [CaFi96] proposed an

application of fuzzy logic in ATM source traffic policing. This approach guaranteed low response time while remaining effective, as well as avoiding complex calculations. A fuzzy logic based congestion avoidance scheme was described in [Qiu97]. Their simulation showed the proposed fuzzy logic predictor outperformed the conventional autoregressive predictor and improved the QoS of high-speed networks. However, there does not appear any work of using fuzzy logic on the jitter control.

The buffer model of jitter management is usually modeled as FIFO (First-In-First-Out) model [Bolo93, MaPa01, PiLa97]. The incoming traffic is sent to the queue where the packets are held, and released to the output link later.

For the source traffic, [AbSo94] investigated Gaussian and Poisson distributions. Recently, wide area traffic was found to be more like self-similar [PaFl95]. An analysis of ATM buffer with self-similar input traffic was presented [LiTs95]. Some evaluations [HuPe00, StJe95] employed actual trace data from a movie as the input of their studies.

1.2. Motivation of Research

A thorough jitter removal mechanism clearly requires control to be exercised at the source node, the intermediate nodes, and the destination node. Ideally a reliable path for the voice stream can be established so that traffic can be controlled when network load is changing. However, in heterogeneous network such as Internet, it is practically impossible or difficult to control the source of an application, not to mention the unknown network environment. Therefore control at the destination end is intuitive and natural. This thesis will focus on the destination end only.

The control mechanisms of I-Policy, E-Policy and Q-monitoring are static. All

parameters are all set and kept no change during operation. We believe this will limit its application on many scenarios when the network load is changing and motivate us to seek more dynamic flavored algorithms in this thesis.

Voice packets are correlated, especially during congestion when several packets suffer from the same large queuing delay. So at the destination end, previous and current information of the packets may provide valuable information for future packets. Therefore, prediction based algorithm is applicable to the destination based jitter buffer management. We will present a prediction-based approach using the Least Mean Square method to the jitter buffer management.

As a different paradigm, we would like to study a fuzzy control scheme. This is because the amount of gaps (and their probability distribution), and the display latency (also their probability distribution) have a fuzzy nature to different audience for different applications. This fuzziness in voice quality leads us intuitively to apply the Fuzzy Logic (FL) method on the jitter buffer management because FL is good at dealing with imprecise but very descriptive language.

The use of jitter buffer will affect both the end-to-end delay and the gap. In order to prevent gaps, a jitter buffer brings an extra delay to the audience, thus increasing the end-to-end delay. Note that, the interactive nature of voice communication may be out of rhythm if the end-to-end delay exceeds 400 milliseconds [ITU93]. Therefore, the trade off of the display latency and the gap probability needs to be studied. We also require them to gauge the performance of our algorithms when compared with some benchmark-algorithms.

1.3. Objectives and Approaches

In this thesis, we are interested in the algorithms that can only be applied at the jitter buffer of the destination end. Specifically, we develop two new algorithms. The first one is prediction-based, using the Least Mean Square method (LMS); the second one is Fuzzy Logic based. We desire these algorithms to have the following characteristics:

1) simple to implement,
2) adaptive to the changing network traffic, and
3) better performance than existing algorithms under most network environments.

To achieve these objectives, we first consider various practical network environments where jitter might occur. Then we apply our algorithm of LMS and Fuzzy Logic. In order to study and compare the results of the jitter buffer management, we abstract the models for various networks and traffic from which we can translate later to simulation models. Two simulation approaches are made. First, we develop programs of the I-Policy, E-Policy and C language, and then use Matlab [Matl01] for the visual demonstration. The second simulation tool is OPNET [Open01]. We will use the I-Policy and E-Policy algorithms as benchmark-algorithms for the performance evaluation of the proposed new algorithms. In addition, instead of measuring jitter, we choose gap probability, average display latency, and loss probability as performance measurements. Reasons are quite obvious. First, jitter only reveals variation of transfer delays of voice stream, not the quality of the recovered voice stream. Since the key point of our algorithms is to adaptively change the buffering to maintain a desirable voice quality, so measuring the gap probability and the loss probability will be more direct and easier.

Second, we think that the average display latency plays an important role in the evaluation of a jitter buffer management algorithm because it reveals how long the extra delay would be incurred if a buffer is introduced. For example, if an algorithm leads to a very low gap probability (or jitter) but a very large latency, which is beyond the acceptable level, we would say this algorithm is not a good algorithm.

We assume we have no knowledge about and control on the source and the network. All our algorithms on jitter buffer management are only applied at the destination node. Instead of using trace data, we have created voice traffic generator to generate voice traffic based on the abstracted on-off model. Network models deal with delays, and our model allows users to select the distribution function of the random delays. We think this is the most effective way to control the scenarios of the network because we can easily investigate the performance under different network environments. Our algorithms are tested under the end-to-end, Ethernet, and tandem nodes network environments. This is because Ethernet is almost ubiquitous, and the skeleton of the model along with application is readily available. Tandem nodes network model is close to the Internet telephony, which is an emerging technology. Simulations are made and the results are compared with existing algorithms. We evaluate the performance of our new algorithms and made comparison among them.

1.4. Contributions

The contributions of this thesis are:

1) The promotion on the use of destination-based approach: As discussed in Section 1.2, there are many scenarios in heterogeneous network environments that this is

the only practical approach.

2) Using LMS as prediction algorithm for the jitter control: Although this method is widely used on network management, its application on the jitter buffer management at the destination only is new.
3) Proposal of Fuzzy Logic algorithm for improving jitter performance: To the best of our limited knowledge on existing approaches, this is the first time to apply the Fuzzy Logic method on the jitter buffer management at the destination end.
4) Formulation and coding of OPNET, C, and Matlab simulation programs: We have done extensive work including debugging to accomplish our objectives and approaches.
5) Comparison among the several algorithms in terms of performance in the gap probability, the average display latency, and the loss probability: We have run extensive simulations to test their performance mentioned on different network environments, to be sure that the results are universal and applicable in the real application.

1.5. Thesis Organization

Thesis is organized as follows: Chapter 2 discusses in details our network models, simulations models and OPNET models. Assumptions are given as well. Chapter 3 defines two benchmark-algorithms, the I-Policy and the E-Policy in order to evaluate our two algorithms. It also defines metrics on the performance evaluation of the jitter buffer management in our study. Chapter 4 presents the Least Mean Square method in details, including mathematical definition, the algorithm and the performance. Chapter 5 presents

the Fuzzy Logic approach in details, including the mathematical definition, the algorithm and the performance, and the procedures on how this algorithm manages the optimal buffer size adaptively. Chapter 6 compares all the algorithms under different network environments. Chapter 7 summarizes our contributions and discusses some future work.

Chapter 2

Network Operation, Modeling and Definitions

In this chapter we present the models that will be used in later chapters. These include the mathematical models, network models and node models, and the traffic models. We also make several assumptions to simplify our study in order to focus our research on the key issues in jitter buffer management.

2.1. End-to-end Model

An end-to-end model for voice stream can be represented by the components in Figure 2.1, the source node, the network, and the destination node. Each in turn can be modeled differently as follows.

Figure 2.1. An end-to-end model.

1) Source node

In digital communication, analog voice is digitized using a vocoder. We shall use the on-off mathematical model [JiSc00] for the voice traffic generated from such vocoder. In this model, one talk session of a speaker alternates between two types of periods: the active period and the idle period, as shown in Figure 2.2. During an active period, talk

spurts (voice streams) are generated and chopped into fixed-length packets, before being sent to the network. While in a silence period, there is no traffic will be generated, therefore, no packet is sent to the network.

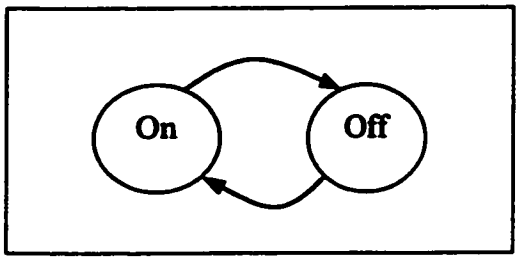

Figure 2.2. An on-off model for the voice traffic generator.

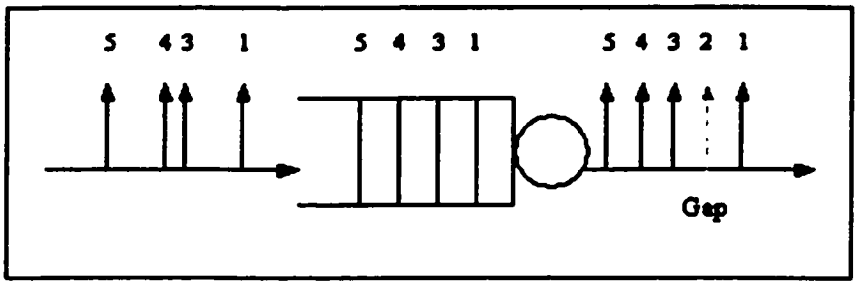

Figure 2.3. A buffer at the destination.

2) Destination Node

We apply the jitter buffer management only at the destination nodes. As shown in Figure 2.3, the destination node can be represented by a simple buffer at which the buffer algorithms are implemented, including existing algorithms and our new algorithms. When arriving at the destination, all the packets are first stored in the buffer momentarily for two purposes: to compensate for the jitter and to reorder any out-of-order packets. Either after a pre-defined initial time or when the buffer size reaches a pre-defined initial buffer size, whichever comes first, the station will start to display (play out) the packet

according to the original timing from the source. If a packet is later than the scheduled display time, a gap occurs, as exampled by packet 2 which has not been shown up in Figure 2.3. (How to fill the gap is beyond the scope of this thesis.) After the last packet, the node stops displaying and goes into the silence period. In real application, the node may play some pre-defined noise to convince the audience that this is silence and not the line is dropped. The jitter buffer uses a FIFO discipline.

3) Network

The network cloud in Figure 2.1 may represent different environments that may produce different jitter statistics. In our study, we will use different models for evaluation.

a) Mathematical model: Here we ignore the implementation details of the network cloud and represent mathematically the random delay experienced by a packet as it traverses the network cloud. The quantity of the random delay can be described by a distribution function. For example, we will use the exponential distribution and the Normal distribution because they are commonly used.

b) Point-to-point model: Under the end-to-end network environment, random delays come from the queuing time at various routers and switches along the path to the destination. Here we simply use a point-to-point connection with random delays generated and added to the arrival time of each packet to represent the network delay. Of course, we also include the constant parts of the network delays.

c) Ethernet model: In this model, the random delays from the source node come from the random access delays when a packet suffers a collision with other packets and has to be retransmitted. Also there is waiting time until the transmission medium is idle. We designate a source and a destination node in the network, and define reference traffic

between them. Other nodes would just generate background traffic that come from various applications like email, FTP, web browsing etc. The background nodes only transmit data to each other (not to the reference nodes). An example is given in Section 2.2.1.3.

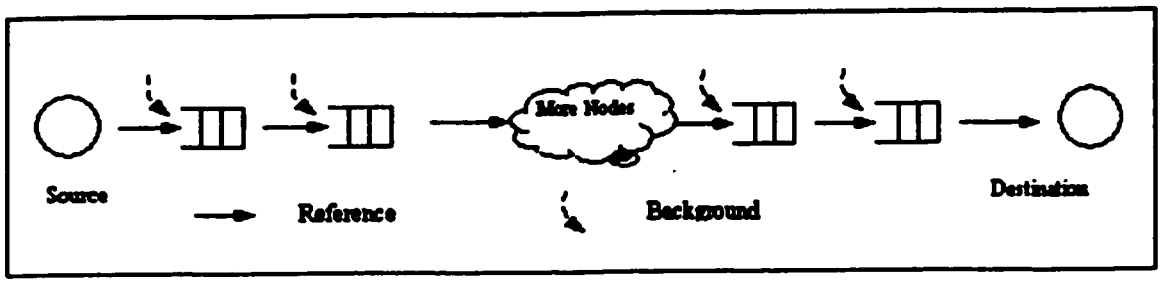

Figure 2.4. Tandem nodes network model.

d) Tandem node model: In this model given in Figure 2.4, each node can represent a real node or a subnet. A voice packet would traverse on its way to the destination. The source and the destination nodes construct reference voice traffic. Each intermediate tandem node has its own background traffic, sharing the same buffer as the reference traffic. Thus the reference voice packets will experience random queuing delays at each tandem nodes, representing the interference effect from the cross traffic in different parts of the Internet. Up to 12 nodes are used, as it is believed that this is the maximum one would go through [LaYa00]. The buffers at each node conform to FIFO discipline.

2.2. Simulation Models

Due to the intractability of theoretical analysis of performance on the network model, we have resorted to simulations as our analysis and evaluation tools. Two simulation tools are used: the OPNET simulator, and the simulator based on C and MATLAB. OPNET Modeler [Opne01] is widely used in network performance evaluation and analysis. It has

comprehensive understanding of networking technologies and enables users to more effectively design and deploy networks, provision services, diagnose network and application performance problems, and predict the impact of network changes. It has hierarchical network model structures that manage complex network topologies with sub-network nesting. It has the capability to represent modeling details at different levels, from the behavior of individual objects at the "Process Level" to "Node Level", "Network Level" and "Project Level" [Opne01]. We have also used Matlab mainly because it can integrate mathematical computing and visualization to provide a flexible environment for technical computing. The open architecture makes it easy to use MATLAB to explore data, visualize algorithms that provide early insights of the object [Matl01].

2.2.1 OPNET Models

There are three network models implemented by our OPNET simulation in order to accomplish different network delay statistics:

A) End-to-end: this is a simple and general network model.

B) Ethernet: this can be used in situations like office building, campus, residential area, etc.

C) Tandem nodes: this has application to the Internet telephony.

All the algorithms are implemented under the same three network models.

2.2.1.1. Common Elements Used in OPNET Network Models

According to OPNET Modeler, a network model consists of node model(s). A node model consists of process models. Before we give the details of the three network

models, we first describe these common elements.

a) The Sender Node Process Model

Each sender node such as that in Figure 2.10 contains a traffic generator that generates voice traffic and sends packets to the link. As shown in Figure 2.5, this generator generates voice talk spurts based on the on-off model. Talk spurts are chopped into fixed-size packets. It does not generate anything during silence period.

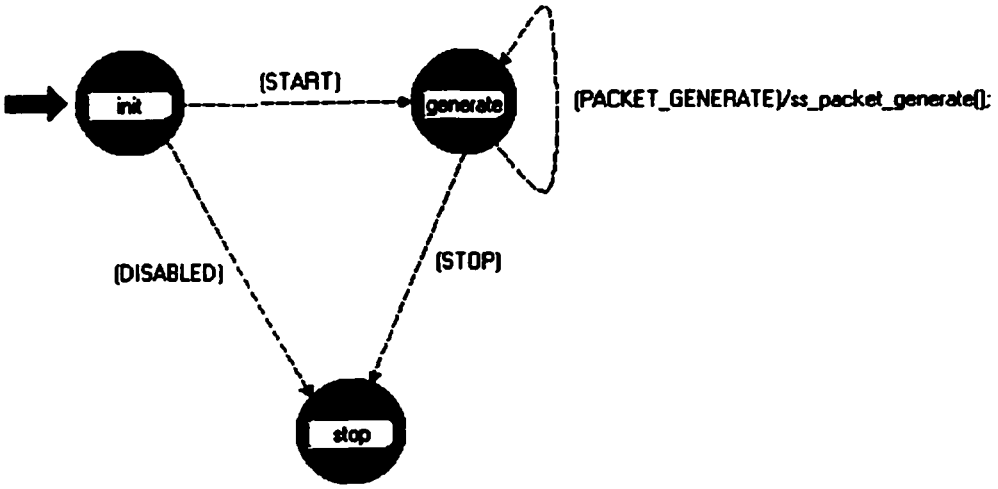

Figure 2.5. OPNET process model for the traffic generator.

For simulation purpose, each packet has an extra header to indicate whether this packet is the first or the last packet of the talk spurt, or something in between. So the destination would trigger the playout mechanism if a packet were the first packet of the talk spurt, and would stop displaying after the last packet. This extra header will not take any processing time for both the sender and the receiver, as it is created only for the simulation purpose.

By default, each packet has a creation time. So the destination end will use it to

calculate and recover the original silence period between two talk spurts. This is determined from the difference between the creation time of the first packet of a talk spurt and the creation time of the last packet from the previous talk spurt.

b) The Network Delay Node Model

Figure 2.6. OPNET node model for transfer delay.

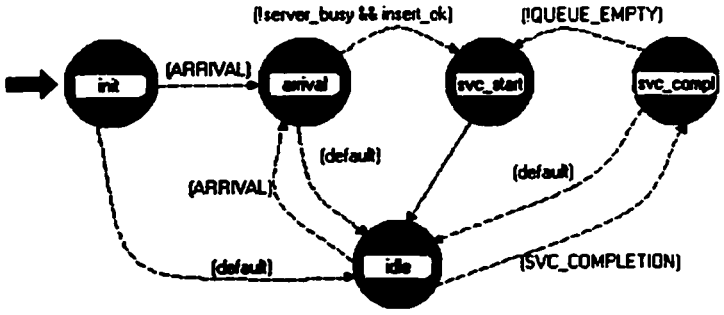

Figure 2.7. OPNET process model for transfer delay.

Figure 2.6 depicts a node model that can add network transfer delays on each incoming packet, before sending it towards the destination. Only the end-to-end and the tandem nodes network models utilize this node. In the Ethernet network model, the delay comes from random access delay.

Figure 2.7 depicts the process model used within this node model. This is an extension of the G/G/1 model program in the standard OPNET library. The added delay

includes a constant delay and a random delay. The buffer size is usually set to a large value so that no buffer overflow will occur, and FIFO (First-In-first-Out) discipline is used. The link capacity is also set to be infinity to simplify the scenarios.

c) The Receiver Node Model

Figure 2.8. OPNET node model for receiver.

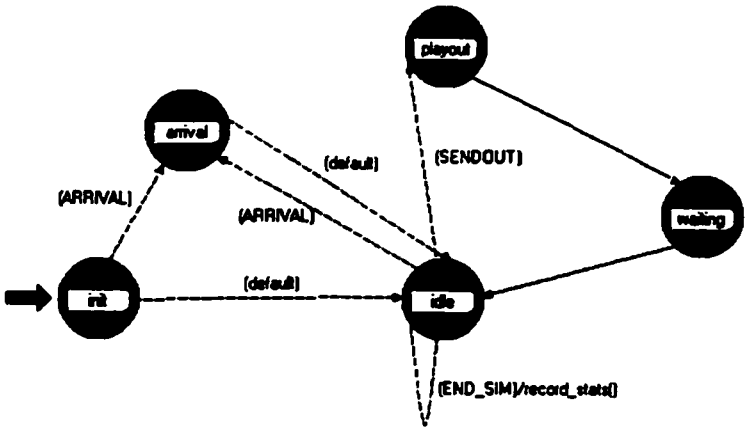

Figure 2.9. OPNET process model for the I-Policy.

Figure 2.8 depicts that the node model buffers the incoming packets, displays packets and destroys packets. The process model in the "buffer" node is where all the key algorithms studied in this thesis. It buffers and displays the incoming packets according to different algorithms. After collecting the statistics, it sends the packets to the sink model for destruction.

Four different process models are designed and coded corresponding to the four

algorithms (the I-Policy, E-Policy, the LMS, and the FLA) in this thesis. Figure 2.9 shows the process model for the I-Policy as one example.

During simulation, we timestamp each packet after it arrives at the destination buffer in order to calculate the extra time that a packet stays in the buffer before displaying. The display latency is simply equal to the difference between the display time and the arrival time of a packet.

Note that in the I-Policy algorithm, packets will be destroyed if it arrives later than its due time. In the LMS algorithm, packets may also be destroyed before displaying in order to reduce the buffer size request. This will keep the latency under an acceptable level while maintaining the gaps under a tolerable level.

2.2.1.2. The End-to-end Network Model

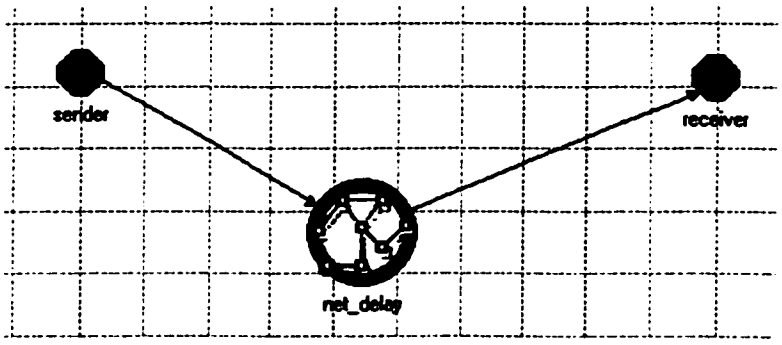

Figure 2.10. OPNET network model.

Figure 2.10 depicts a network in which a "sender" node sends voice traffic in one direction to the "receiver node". The "net_delay" node model provides both a random and a constant delay of the networks, before a packet reaches the "receiver" node. This is implemented by the network delay node model mentioned in Section 2.2.1.1.b. After being displayed, packets are destroyed at the "receiver" nodes. The only difference of

each algorithm is at the receiver's "buffer" process model. A receiver model in Section 2.2.1.1 is implemented in the "receiver".

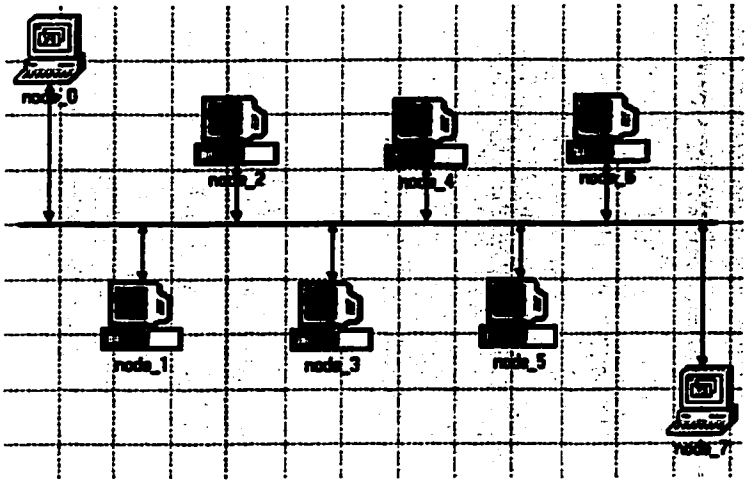

Figure 2.11. OPNET network model for Ethernet network model.

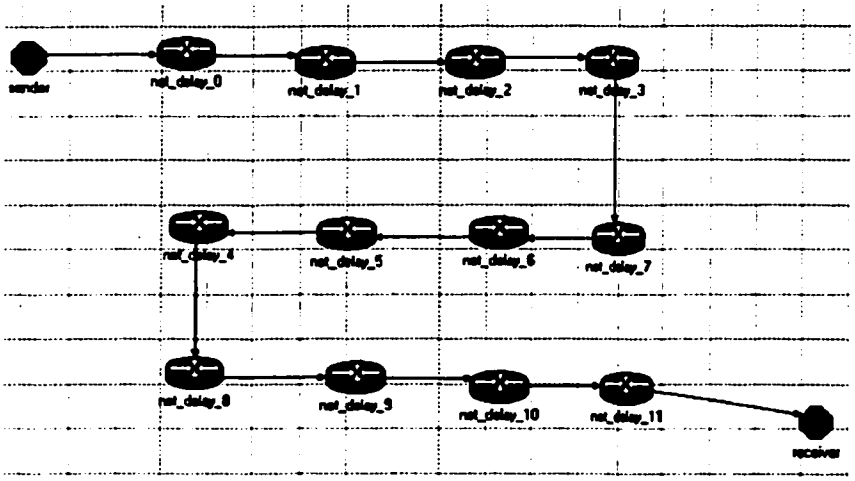

Figure 2.12. OPNET network model for Tandem nodes Scenario.

2.2.1.3. The Ethernet Network Model

Figure 2.11 depicts the Ethernet network model in our simulation. All the algorithms have the same Ethernet network model. "node_0" and "node_7" are terminal stations set

to be the source and destination nodes; they only send voice traffic (defined as *reference traffic*) to each other, not to other workstations. Other workstations send traffic (defined as *background* traffic) to each other only, neither to the source nor to the destination nodes.

2.2.1.4. The Tandem Nodes Network Model

Figure 2.12 depicts the Tandem Nodes Model. There are 12 intermediate nodes connected as series between two terminal nodes, the source node and the destination node. Each intermediate node, net_delay_i, i = 0, 1, 2, 3, ...11, is implemented by a node model mentioned in Section 2.2.1.1.b, and it provides both a random (due to the cross traffic which shares the same buffer and server with the reference traffic) and a constant delay of the networks, before a packet reaches the "receiver" node. After being displayed, packets are destroyed.

2.2.2. C and Matlab Model

In order to capture the behavior of the jitter buffer management, we have also developed a C-program that generates voice packets at a source and sends them to a buffer at the destination. Voice packets have constant packet size and inter-arrival time, and are subjected to random transfer delays before arrival at the destination. At the destination, three jitter buffer algorithms (the I-Policy, the E-Policy, and the Queue-Monitoring) are implanted independently. During simulation, the program records the relevant information of a certain talk spurt, which can be specified by users before simulation. After simulation, users can use Matlab [Matl01] to plot a graph that contains the network transfer delay, display latency, and gaps of each packet. Details are explained in the

graph in Section 3.3.1.

2.3. Assumptions

We need assumptions to focus our investigations and evaluations. Generally, we assume that we have no knowledge about the source and the network. All the algorithms are implemented only at the destination nodes. Specifically, we further take the following assumptions:

1. There is no packet loss when the packets travel through a network. However, at the destination node, if the packet is later than a time limit, it will be considered as lost.

2. The data rates of all links are infinite. This is because OPNET allows all delays to be summarized in the node process.

3. The buffer size at the destination is infinite because we want to simplify the scenarios. The exception is the buffer size of the I-Policy, which is finite, and can be decided by user.

4. In all the simulations of the thesis, it takes one time slot for the node to display one packet.

Chapter 3

Definitions and Benchmark

There is no well-acceptable method or standard, as far as this author knows, that would allow one to determine which algorithm is better than others. A common practice to solve this problem is to define a benchmark-algorithm as the reference standard, and then compare our new algorithm against it. In this thesis, we have adopted the I-Policy and the E-Policy as the benchmark-algorithms, and we shall give a brief description of them in this chapter.

3.1. The I–Policy

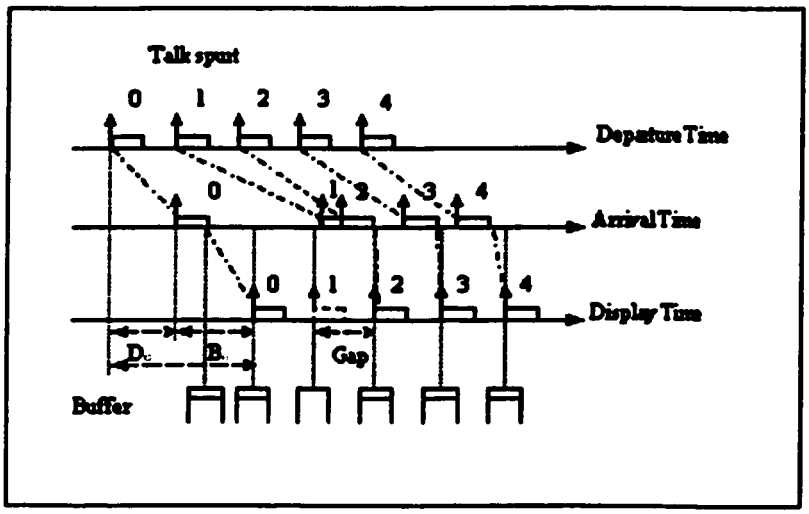

Figure 3.1. The I-Policy.

In the I-Policy, the initial buffer time is pre-defined and fixed for each talk spurts. After the initial buffering, the destination node starts to display the voice packets with the original timing. If a packet is late for the due display time, it will be considered as lost and be discarded (Ignored), as shown by packet # 1 in Figure 3.1. Therefore, there is a

gap at the time of display when that packet is absent. This algorithm is quite simple, and it tries to maintain a constant end-to-end delay for all the packets within a talk spurt, this is shown in Figure 3.3.

3.2. The E–Policy

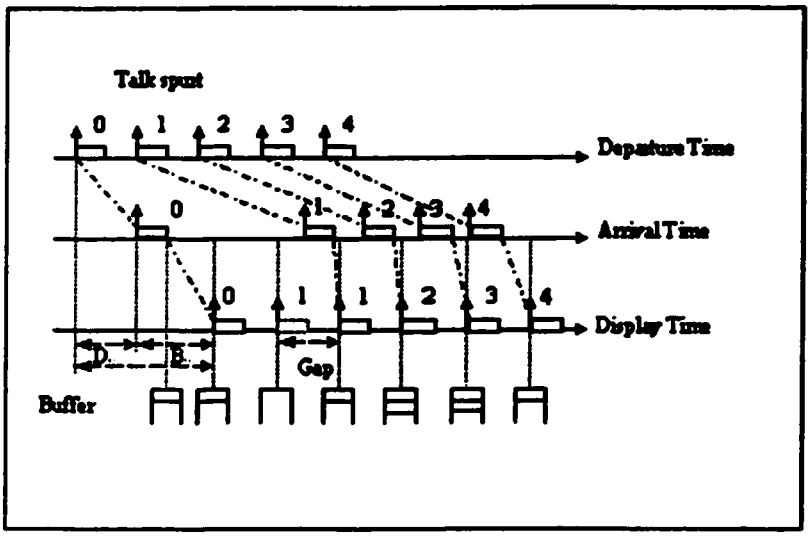

Figure 3.2. The E-Policy.

Like the I-Policy, the destination node starts to display the voice packets with the original timing after the initial buffering. However, if a packet is late for the due display time, the node will wait until it arrives. The following packets, if they come earlier will be delayed as well. So the display timing is delayed (Extended). This is illustrated in packet # 1 of Figure 3.2. Notice that there are gaps during the waiting time (i.e. time difference between the actual display time and the scheduled display time of a voice packet). This algorithm tries to maintain the information as much as possible. The drawback is that waiting for a late packet will delay all the display time for subsequent packets. Especially as more late packets it experienced, the larger display latency it will accumulate, and

there is no mechanism to reduce the latency. As a result, the display latency is normally larger than other two algorithms(detail comparison can be found in Section 3.4). The good-point of this algorithm is that as the initial buffer time increases, there is less probability it will experience gaps, as shown in Figure 3.3.

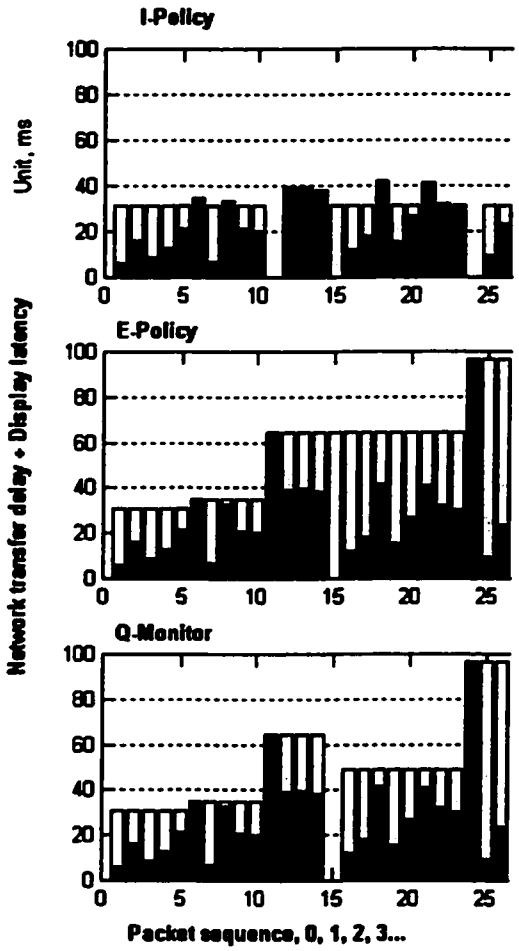

Figure 3.3. Transfer Delays and Latency in the I-Policy, E-Policy and Queue Monitoring.

3.3. The Queue Monitoring Algorithm

In the pursuit of better buffer algorithm, we have experimented with the Queue Monitoring algorithm as an extension of the E-Policy. In this algorithm, if a packet is

late, the node will wait until it arrives (as in the E-Policy). In order to prevent the accumulation effect of the latency, this algorithm monitors the queue size of the buffer. The idea is that when the queue size reaches a pre-defined threshold, the node starts to discard the oldest packet in the buffer. In this way, it is able to maintain the level of the queue size to be under the threshold. As a result, it prevents the display latency (accumulated from the previous packets) from becoming too high, as shown in Figure 3.3. On the other hand, it may experience more gaps than the E-Policy.

3.4. Performance Comparison

We have implemented the I-Policy, E-Policy, Queue Monitoring and a display program in C-language in order to compare their performance. In this experiment, a sender node sends voice traffic to a receiver node and each voice packet experiences a network delay during transmission. The probability density function of the network delay has a Normal distribution; we have chosen a mean of 20.0 ms, a standard deviation of 20.0 ms, an initial buffer time of 20.0 ms. The buffer size of the I-Policy is 4800 bits, and the buffer size of both the E-Policy and the Queue Monitoring are set to infinity. Finally, the queue threshold in Queue Monitoring is 3 packets. The experiment's visual results are shown in Figure 3.3, and the numerical results are presented in Table 3.1.

The graphs of Figure 3.3 capture each packet's network delay and display latency, as well as the gap occurrences. The black bars represent network delays and the yellow (shown as gray in black and white print) bars represent the display latencies. If a bar is absent (blank), it means a packet is missing (discarded) and a gap has occurred. Notice that sometimes a black bar is taller than its previous bar (black and yellow together), it

also indicates that there is a gap, and the length of gap (measured in millisecond) equals to the difference between the taller one and the shorter one.

Table 3.1. Performance Comparison of the I-Policy, E-Policy and Queue Monitoring Algorithms in term of Gap probability, Average Display Latency and Loss Probability.

(* Packets destroyed in the E-Policy are 0 because we assume an infinite buffer size and we will receive all packets sooner or later.)

	GAP PROBABILITY	AVERAGE DISPLAY LATENCY	LOSS PROBABILITY
I-Policy	8.08%	27.46 ms	3.82%
E-Policy	2.07%	38.28 ms	0 *
Queue Monitoring	2.60%	33.32 ms	1.39%

The simulation results reflect the fact that the I-Policy and the E-Policy fall the extreme ends of the performance in term of gap probability, average display latency and loss probability. The Queue Monitoring Algorithm falls between the I-Policy and the E-Policy. Because Queue Monitoring starts discarding packets when the queue size reaches the threshold, the queue size and the display latency are reduced. However, as a tradeoff, the loss probability and the gap probability increase. This experiment demonstrates that the I-Policy and the E-Policy are suitable as benchmark for our further evaluations of new algorithms.

3.5. Gaps

We define gap here to be the omission of information at a time when it is expected. As we study the causes of gaps, we found that there exist three types of gaps during the display operation of voice packets. These are shown in Figure 3.4. They have different properties and natures, and therefore, worth to be understood as they all contribute to the intelligibility of the recovered information (i.e. voice).

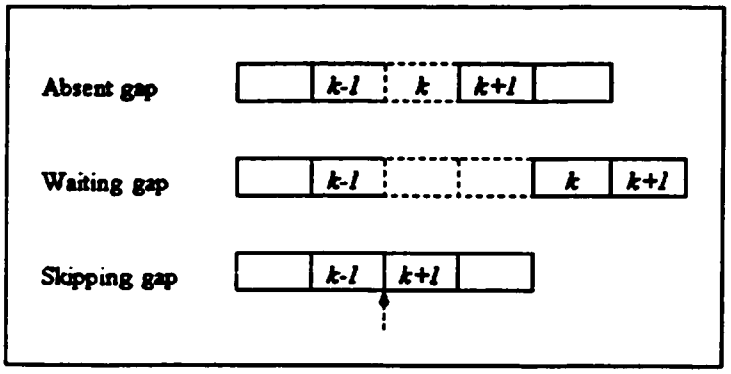

Figure 3.4. Three types of gaps are defined in this thesis.

1) Absent gaps: This can be found in the I-Policy. As shown in Figure 3.4, when packets $k-1$, k, $k+1$ are supposed to be displayed. However, packet k is late for its due display time and this late packet is discarded. So when the node starts to display packet k, a gap occurs. The length of this gap is T (e.g. 20 ms), which is the scheduled displaying interval in the de-codec algorithm. If the time is slotted, and each unit is T, then one late packet will introduce only one absent gap. Conversely, one absent gap represents only one lost packet.

2) Waiting gaps: If we use the same example above in the E-Policy, the node will wait for the late packet k until it arrives. During this waiting time, which may be longer than T (depending on how late it is), there may be gap(s). So one waiting gap will span at least

one time slot. If one packet takes one slot, then the total number of slots actually used for displaying voice packets may be larger than the total number of packets. These differences represent the number of waiting gaps, because the E-Policy has extended the display timing. For example, there are 2 slots of waiting gaps occurred as the node waits for late coming packet k in Figure 3.4.

3) Skipping gaps: Again, we use the same example above in the Queue Monitoring algorithm. Suppose the node is displaying packet $k-1$, and at this time the queue size has reached the threshold. Be definition, one packet has to be discarded, which is the oldest packet k in the buffer. Then after finishing displaying $k-1$, the node will display $k+1$, while skipping k.

In [StJe95], the authors claimed that the audience would not notice the difference, and they therefore did not count skipping gaps as gaps. This is only true when skipping gaps are few, but when the skipping gaps are many, we must account for them in some way. The more information is lost, the more the quality of voice will be degraded. Our thesis therefore introduces the packet loss probability as a performance measure, and we evaluate it in all our simulations.

3.5.1. Gap Measurement

Since we assume one time slot is required to display one packet (Section 2.3), we can define gap probability G_{prob} to be:

$Gap\ probability(G_{prob}) = \frac{Total\ number\ of\ gap\ slots}{Total\ \ number\ of\ displaying\ \ slots}$. We shall express all gap probabilities in percentage. Take Figure 3.3 as an example, the gap probabilities of the I-

Policy and the E-Policy are 8.08%, and 2.07% respectively. Note that only absent gaps and waiting gaps are counted as gaps. Under the same environment, if algorithm A has a lower gap probability than B, we say A outperforms B in gap probability.

3.6. Average Display Latency

We shall define the average display latency L_k of packet k to be $L_k = D_k - A_k$, where D_k is departure time of packet k from, and A_k is its arrival time to the buffer. So the average display latency can be computed as: $L = \frac{\sum_m L_k}{m}$, where m is the total number of packets that have been successfully displayed. $(m \neq 0)$. Under the same environment, if algorithm A has a lower average latency than B, we say A outperforms B on average latency.

3.7. Loss Probability

We shall define the loss probability of a session to be $L_{prob} = \frac{Total\ discarded\ \ packets}{Total\ \ packets}$.

Under the same environment, if algorithm A has a lower loss probability than B, we say A outperforms B on loss probability.

3.8. Initial Buffer Time/Size Thresholds

We shall define the initial buffer time threshold B_0 to be $B_0 = T_1 - T_0$, where T_0 is the time that the node receives the first incoming packet (may not be packet-1), and T_1 is the time that the node starts to display packet-1. Similarly the initial buffer size threshold is

defined as the queue size of the buffer before the node would start to display packet-1. When the initial buffer time or the initial buffer size reaches its predefined threshold, whichever comes first, the node immediately starts to display the packets.

Chapter 4

The Least Mean Square Method

The performance of both the E- and I-Policies suffer from using fixed buffering during the entire talk session. To address the above problems, we need a mechanism that can dynamically adjust the buffer scheme, not only to keep the gap probability to an acceptable level, but also to maintain the latency under a maximum bound as required by real time applications. In this chapter, we shall study the Least Mean Square (LMS) method because LMS prediction can be used as an on-line algorithm for forecasting time intervals. We shall give the mathematical formulation and then present the procedures and flowchart of how the LMS predictor works. Finally, we evaluate the LMS algorithm using OPNET simulations.

4.1 Mathematical Formulation

In both the I-Policy and the E-Policy, the initial buffering (either in unit packet size or time millisecond) is fixed during a talk session. This has led to some undesirable subsequences. Under the I-Policy, if the initial buffered packets are not enough, and if the packets' incoming rate is lower than their outgoing rate, the playout buffer will become empty and therefore a large gap probability will occur. Under the E-Policy, although the buffer size may be increased for late packets, there is no mechanism to reduce the queue size. As a result, the packets that follow will suffer unnecessarily large latency. The larger buffer size and longer end-to-end delay are both undesirable to real time

application.

The LMS (Least Mean Square) method has been widely used in prediction because of its simplicity and effectiveness. A *k*-step and *p*th-order least mean square linear prediction can be implemented by the equation [Hayk01]: $y(n+k)=\sum_{l=0}^{p-1} w(l)x(n-l)$. In this thesis *x(n)* are observed values of the interarrival time of the packets, *y(n)* are predicted values of the interarrival time of future packets, and *w(l)*, for $l = 0, 1, 2,\ldots,$ p-1, are the prediction filter coefficients. In our study, we use one-step prediction (i.e. $k = 1$), and *w(l)* are the time varying prediction coefficients for the interarrival time of each packet. The optimal linear predictor in the mean square sense is the one that minimizes the mean square error $\xi = E\{e(n)^2\}$, where *e(n)* is the difference between an observed value and its predicted value. Each *e(n)* sample will be fed back to adapt the filter coefficients in order to decrease the mean square error.

4.2. The Queuing Model and Assumption

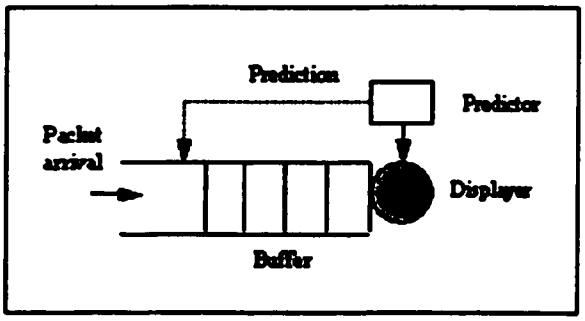

Figure 4.1. The prediction mechanism in LMS.

To apply this mechanism, we first assume that the network traffic is predictable, i.e. future traffic values can be correlated to current and past values. Secondly, due to the

nature of most real time application, user can tolerate a small packet loss percentage. Therefore, some packets that are already in the buffer might be skipped and discarded in order to reduce the queue size and the display latency of subsequent packets. Here we apply the LMS algorithm at the destination end only as mentioned in the Introduction. We assume we are not able to control or get the information about the network except that about the receiver. The schematic diagram of this prediction mechanism implemented on a queue is depicted in Figure 4.1.

The model consists of one queue (buffer), one server (displayer), and one controller (predictor). An incoming packet is held in the buffer when the displayer is displaying a packet (i.e. the server is busy). Otherwise, it will enter the server to be displayed directly. The queue capacity is assumed infinite, and FIFO queuing discipline is used. When the displayer enters its displaying mode, the service rate is fixed due to the fixed packets size and a constant processing rate. If the queue size reaches a predefined threshold, the predictor is triggered to predict the next packet interarrival time. If this prediction value were smaller than a threshold value (i.e. the packet is ahead of schedule, and the queue size will still increase), then the displayer will discard the oldest packet in the buffer. As a result, the displayer should skip the discarded packet and display the next available packet in the buffer.

4.3. The LMS Algorithm

The LMS algorithm consists of four basic processes:

1. A *triggering process*, which triggers the LMS algorithm. The queue size Q is monitored when the packets arrive and be sent at the buffer, if the buffer size reaches

the threshold Q_0, then this will trigger the LMS algorithm to make prediction and action.

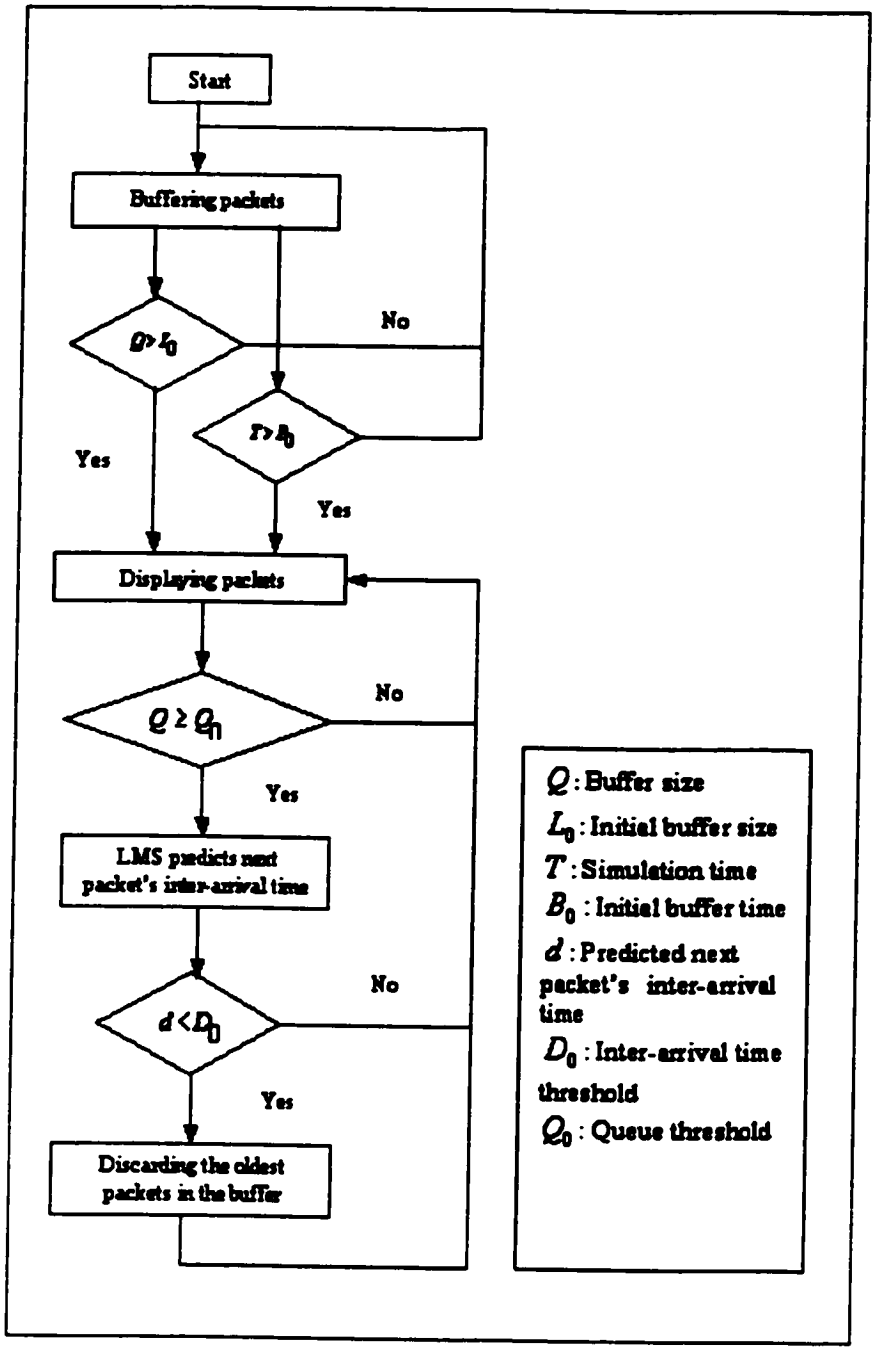

Figure 4.2. Flowchart of LMS algorithm for a talk spurt.

2. A *prediction process,* which involves (i) computing the next packet's interarrival time d, from the current and the previous packets' interarrival time, and (ii) comparing the computed value with a pre-defined threshold D_0.
3. A *packet-discarding process*, which discards the oldest packet in the buffer if the result predicted is less than the threshold (D_0), i.e. the prediction shows that the next packet will come earlier than expected. So discarding one packet in the buffer will keep the queue size under the threshold. Whereas if the predicted result is larger than the threshold, the coming packet is buffered and no packet is discard.
4. An adaptive process, which involves updating the coefficients $w(n)$ automatically in accordance with the estimation error.

The flow chart of the algorithm is given in Figure 4.2, and more mathematical details of the steps are the following:

1) Start with an initial estimate of the filter (prediction) coefficients $w(0)$.
2) For each new data point of the queue length, compute $\nabla\xi = -2E\{e(n)x(n)\}$, where $x(n)$ is a sequence of observed values. In simulation, these statistics are not known and may change with time. Therefore, the expectation operation E is replaced by an estimate. In our investigation, we use the simplest estimate of one point sample average of $e(n)x(n)$, i.e. $\frac{1}{p}\sum_{i=1}^{p} e(i)x(i)$.
3) Update $w(n)$ by taking a step of size 0.5μ, (μ is a constant, normally called as step-size or learning rate, it determines how fast the iterations reach the bottom of the performance surface) in the negative gradient direction. This will point the prediction

to the bottom of the performance surface. Mathematically [Hayk01],

$$w(n+1) = w(n) + 0.5\mu\nabla\xi$$

$$w(n+1) = w(n) + \mu e(n)x(n).$$

4.4. Performance Evaluation

Simulation program were written to evaluate our LMS algorithm. They were run on a PC with a Pentium III 1GHz CPU and 1 G byte memory. The operation system is Windows 2000 Professional. The release version of OPNET modeler is 7.0.B (Build 1109). A typical simulation run collects the statistics of about 66800 packets, and it takes about 10 minutes to run.

In our experiments, the packet interarrival time is constant at 0.02 second, and the packet size is constant at 1280 bits. The talk spurt interarrival time (i.e. silence period) is exponentially distributed with a mean of 0.65s, and the talk spurt size is exponentially distributed with a mean of 22400 bits. The initial buffer time B_0 has a default value of 0.15 s, but it can be promoted (i.e. changed before each simulation). We systematically vary the initial buffer sizes L_0 from 2 to 9 packets in steps of 1 in most experiments. The interval threshold D_0 is set to 0.02 second and is fixed in all simulations because we have designed it to be equal to the packet interarrival time of the source node. The queue threshold Q_0 has a default value of 5 packets, but can also be promoted. The step-size μ is set to be constant at 0.05, and the order parameter p is set to be 5.

In the end-to-end scenario, the constant network delay component is 6 ms and the random network delay component is exponentially distributed with a mean of 15ms. In the tandem nodes scenario, the delay of each tandem node has a constant component of

0.1 ms and a random component that is exponentially distributed with a mean of 1ms.

We have also obtained 95% confidence intervals to verify the sanity of our simulations. But due to the amount of simulations, we did this test (Alpha = 0.05) on a few scenarios. A typical 95% confidence intervals performance curve with 8 points (each with 11 samples) takes about 15 hours. Figure 4.3 is an example of the loss probability versus initial buffer size. Each simulation point is obtained as an average of 11 simulation runs. As one can see, the 95% confidence intervals bounded by the dotted lines (----) are small compared with the mean, thus indicating the accuracy of our simulations.

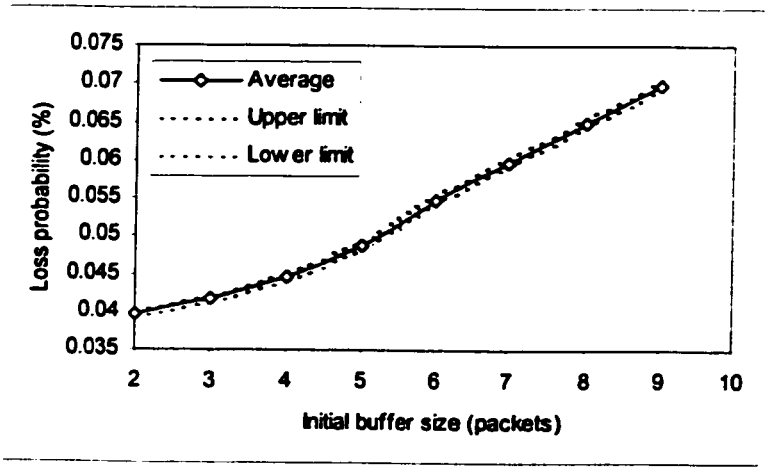

Figure 4.3. Loss probability performance and the 95% confidence interval.

4.4.1. Gap Probability Performance

In this section, we present the gap probability performance (defined in Section 3.5.1) of the end-to-end model, the Ethernet model and the tandem nodes model. We have observed that in all these models the gap probability of LMS is a decreasing function of the initial buffer size, e.g. the $Q_0 = 3$ curve in Figure 4.4. This result is expected because the more packets are buffered, the less chance the receivers will run out of packets,

hence, the smaller the gap probability. In the extreme when the initial buffer size is very large and all packets are buffered, there is no chance for the displayer to experience a gap.

4.4.1.1. The End-to-end Model

The OPNET model of this end-to-end model has been discussed in the Section 2.2.1.2.

A. Effect of the Queue Threshold Q_0

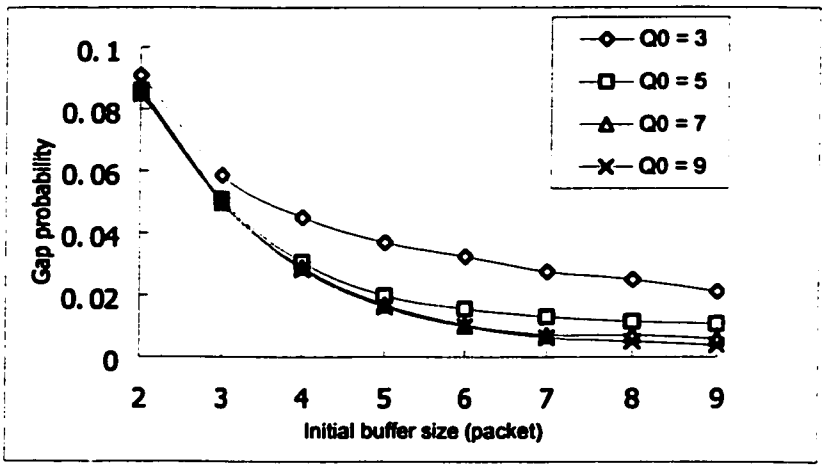

Figure 4.4. Gap probability vs. initial buffer size with different queue threshold values in the end-to-end scenario of LMS.

We have run simulations to reveal the dependency of the LMS performance on the queue threshold Q_0. Figure 4.4 depicts a system with B_0 = 0.15s. Under the end-to-end scenario, one can see that the queue threshold has impact on the gap probability. For a fixed initial buffer size L_0, as the queue threshold values get larger, the gap probabilities become smaller. This is because when the threshold value Q_0 becomes larger, more packets can be buffered before discarding would occur. Therefore there is less probability for a gap.

We also notice that when the threshold values are very large, like Q_0 = 7 and 9 packets, their gap probability performance are very close because the actual queue sizes seldom reach such queue thresholds.

B. Effect of the Initial Buffer Time B_0

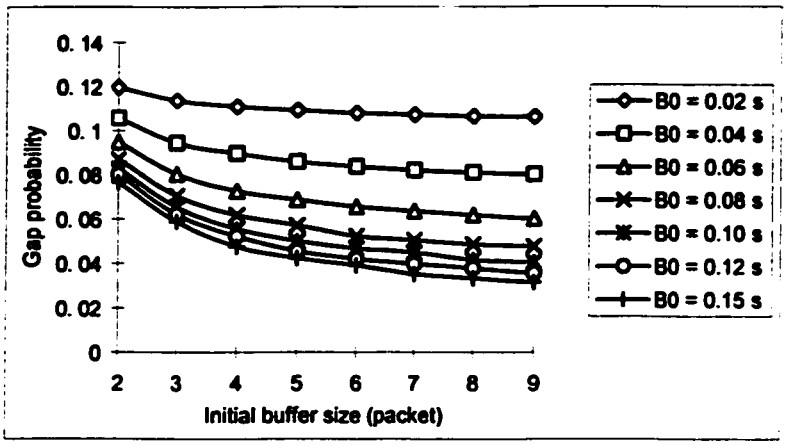

Figure 4.5. Gap probability vs. initial buffer size with different initial buffer time values in the end-to-end scenario of LMS.

We have run simulations to reveal the dependency of the LMS performance on the initial buffer time B_0. Figure 4.5 depicts that under the end-to-end scenario, the gap probability is a decreasing function of the initial buffer size. It also depicts the trend that when the initial buffer time increases, the gap probability of LMS decreases. This is as expected because when the initial buffer time is large, more packet are buffered, so less probability the node will experience a gap. We also observe that as the initial buffer time is small (e.g. B_0 = 0.02s), the gap probability drops slightly only, and remains constant even when the initial buffer size increases. This is because the initial buffer time is reached first and the node starts to playout packet. In this case, increasing the initial buffer size has no

impact on the gap probability any more.

4.4.1.2. The Ethernet Model

The OPNET model of this Ethernet model has been discussed in the Section 2.2.1.3.

A. Effect of the Queue Threshold Q_0

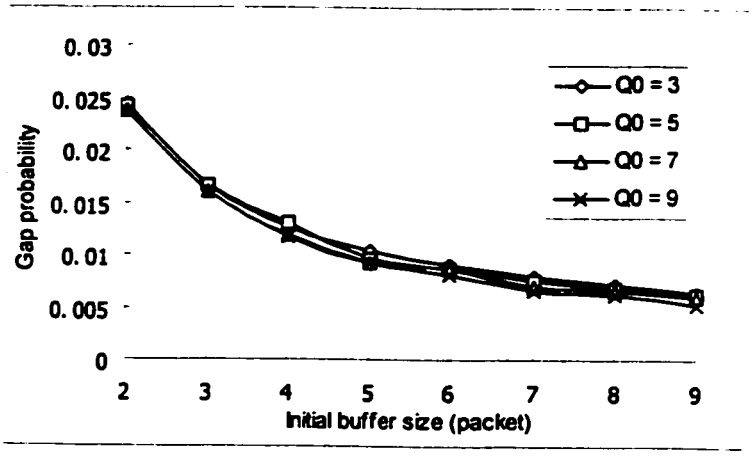

Figure 4.6. Gap probability vs. initial buffer size with different queue threshold values in the Ethernet scenario of the LMS.

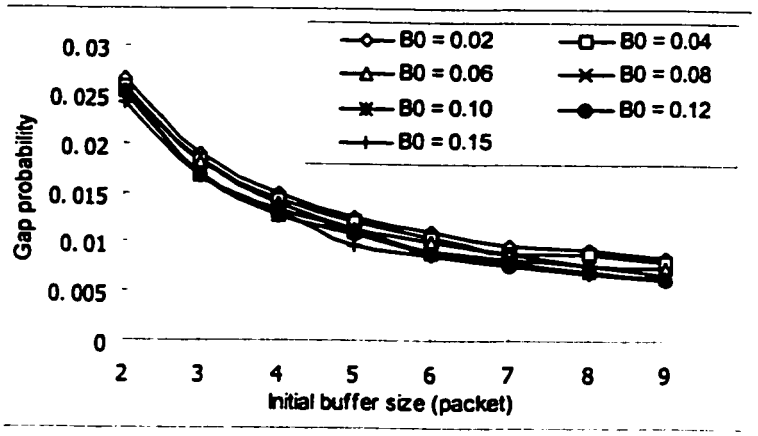

Figure 4.7. Gap probability vs. initial buffer size with different initial buffer time values in the Ethernet scenario of the LMS.

Figure 4.6 considers a system with B_0 = 0.15 s. Since all B_0 performance curves all

close to each other, we conclude that the queue threshold has less impact on the gap probability under the Ethernet scenario. This is because the actual queue size is small and seldom reaches the queue threshold.

B. Effect of the Initial Buffer Time B_0

Figure 4.7 considers a system with $Q_0 = 5$ packets under the Ethernet model. Although it depicts the trend that when the initial buffer time increases the gap probability of LMS decreases, however, simulations show that this impact is limited when compared with the one in the end-to-end model. This is because the actual queue size is small and B_0 is seldom reached.

4.4.1.3. The Tandem Nodes Model

The OPNET model of this tandem nodes model has been discussed in the Section 2.2.1.4.

A. Effect of the Queue Threshold Q_0

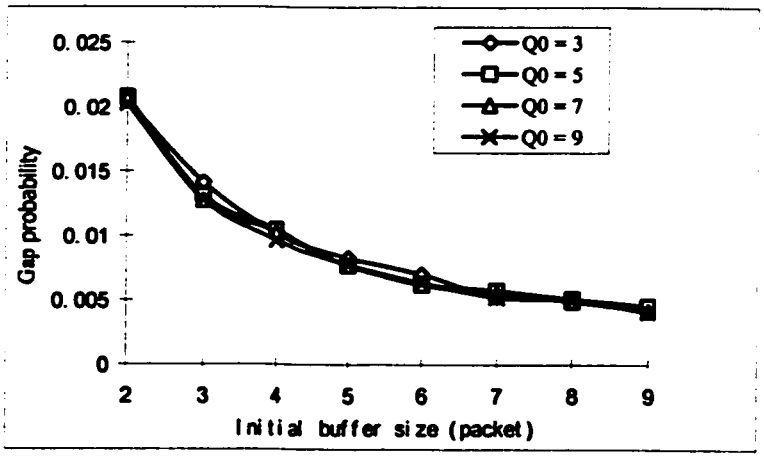

Figure 4.8. Gap probability vs. initial buffer size with different queue threshold values in tandem nodes scenario of the LMS.

Figure 4.8 considers a system with $B_0 = 0.15$ s. We observe that, the queue threshold in

this model has less impact on the gap probability performance when compared with the end-to-end model. This is because the actual queue size is small and seldom reaches the queue threshold.

B. Effect of the Initial Buffer Time B_0

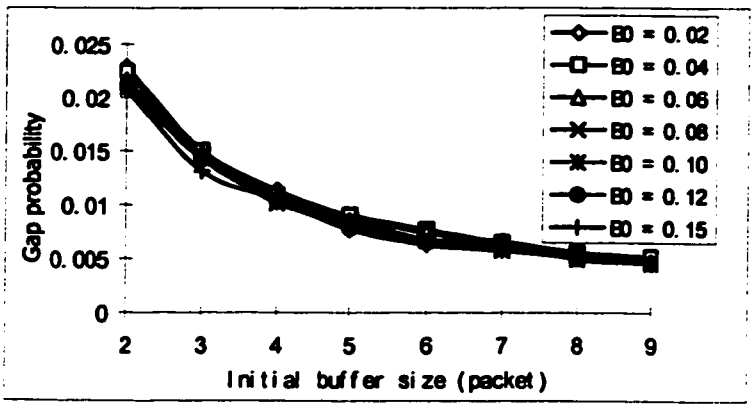

Figure 4. 9. Gap probability vs. initial buffer size with different initial buffer time values of LMS under tandem nodes scenario.

Figure 4.9 considers a system with $Q_0 = 5$ packets under the tandem nodes model. It depicts that the initial buffer time has a less impact on the gap probability under the tandem nodes model contrasted with the one under the end-to-end model. This is because the actual queue size is small and B_0 is seldom reached.

4.4.2. Average Display Latency

In this section, we present the average display latency performance (defined in Section 3.6) of the end-to-end model, the Ethernet model and the tandem nodes model. We have observed that in all these models the average display latency of LMS is an increasing function of the initial buffer size, e.g. the $Q_0 = 9$ curve in Figure 4.10. This is as expected

because as more packets are buffered, on average, packets will stay longer in the buffer before displaying.

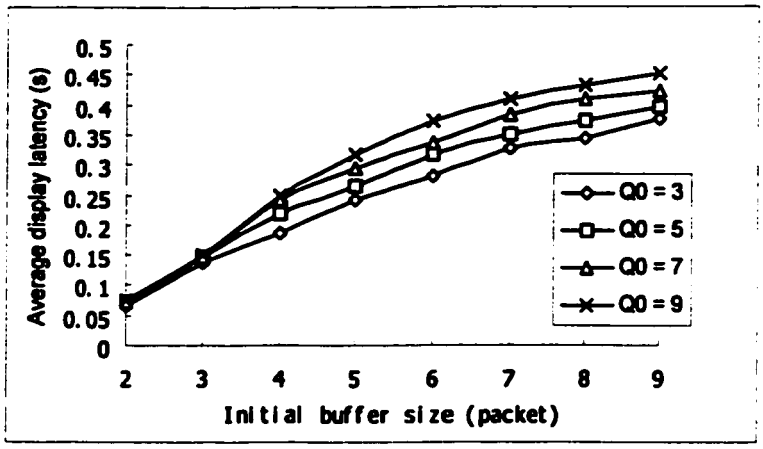

Figure 4.10. Average display latency vs. initial buffer size with different queue threshold values in the end-to-end scenario of LMS.

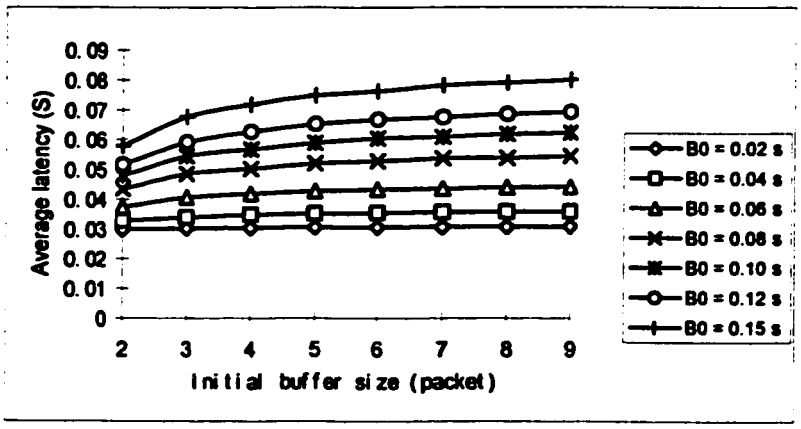

Figure 4.11. Average display latency vs. initial buffer size with different initial buffer time values in the end-to-end scenario of LMS.

4.4.2.1. The End-to-end Model

The OPNET model of this end-to-end model has been discussed in the Section 2.2.1.2.

A. Effect of the Queue Threshold Q_0

Figure 4.10 considers a system with B_0 = 0.15s. It depicts that under the end-to-end scenario, as the queue threshold increases (for a fixed L_0), the average latency also increases. This is because when the queue threshold is large, the actual queue size is also allowed to be large, so the latency is large. However, as the initial buffer size is small (e.g. L_0 = 2, and 3 in Figure 4.10), this effect is not obvious because the actual queue size seldom reaches the threshold. For this case, increasing the queue threshold will not affect the average display latency. This is clearly illustrated in Figure 4.10.

B. Effect of the Initial Buffer Time B_0

Figure 4.11 considers a system with $Q_0 = 5$ packets in the end-to-end scenario. It depicts the trend that when the initial buffer time increases, the average display latency of LMS increases. This is as expected because when the initial buffer time is large, more packet are buffered, and on average, the packets will stay longer in the buffer. We also observe that as the initial buffer time is small (e.g. B_0 = 0.02 and 0.04 seconds), the average display latency remains almost constant even when the initial buffer size increases. This is because the initial buffer time is reached first and the node starts to playout packet. So in this case, to increasing the initial buffer size has no impact on the average display latency.

4.4.2.2. The Ethernet Model

The OPNET model of this Ethernet model has been discussed in the Section 2.2.1.3.

A. Effect of the Queue Threshold Q_0

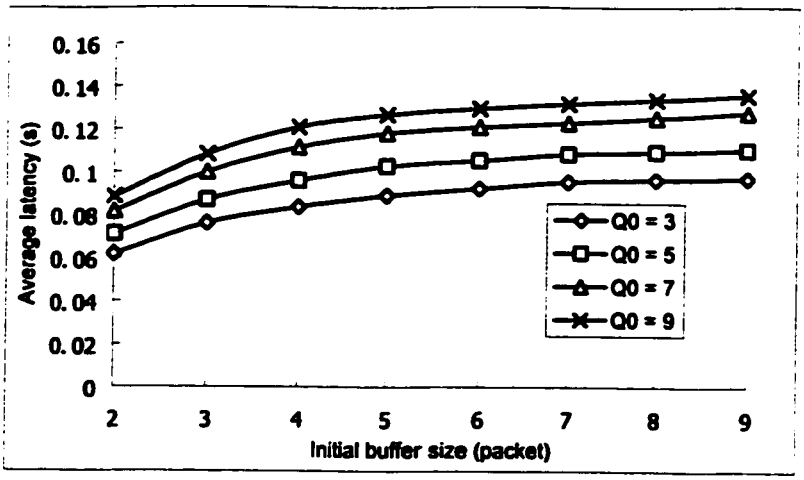

Figure 4.12. Average display latency vs. initial buffer size with different queue threshold values in Ethernet scenario of the LMS.

Figure 4.12 considers a system with B_0 = 0.15s. It depicts that under Ethernet scenario, the queue threshold of LMS has an impact on the average display latency. When the queue threshold is large, the latency is also large. This is as expected because as the threshold value is large, the queue size of the buffer is allowed to be large. Therefore, on average each packet will stay longer in the buffer before displaying.

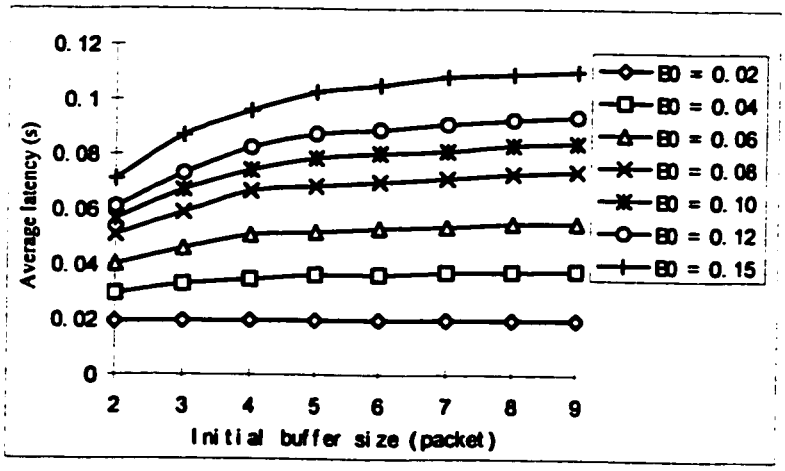

Figure 4.13. Average display latency vs. initial buffer size with different initial buffer time values in the Ethernet scenario of the LMS.

B. Effect of the Initial Buffer Time B_0

Figure 4.13 considers a system with $Q_0 = 5$ packets under the Ethernet model. It depicts the trend that when the initial buffer time increases, the average display latency of LMS increases. We also observe that as the initial buffer time is small (e.g. $B_0 = 0.02$ seconds), the average display latency remains almost constant even when the initial buffer size increases. This is because the initial buffer time is reached first and the displayer starts to playout packet.

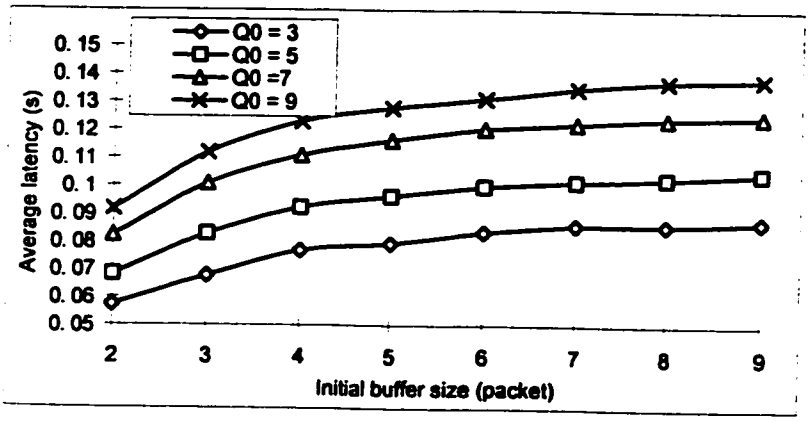

Figure 4.14. Average latency vs. initial buffer size with different queue threshold values under the tandem nodes scenario of the LMS.

4.4.2.3. The Tandem Nodes Model

The OPNET model of this Ethernet model has been discussed in the Section 2.2.1.4.

A. Effect of the Queue threshold Q_0

Figure 4.14 considers a system with $Q_0 = 5$ packets under the tandem nodes scenario. It depicts that the threshold value Q_0 of LMS has an impact on the average display latency. When the threshold value is large, the latency is also large. This is also as expected

because as the threshold value is large, the queue size of the buffer is allowed to be large. Therefore, on average each packet will stay longer in the buffer before displaying.

B. Effect of the Initial Buffer Time B_0

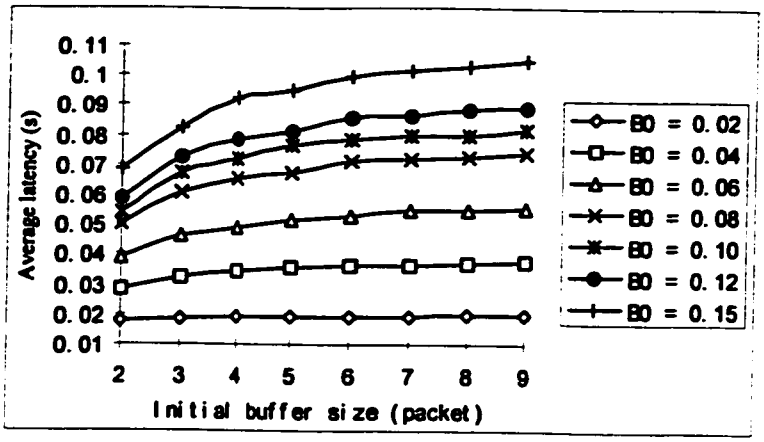

Figure 4.15. Average display latency vs. initial buffer size with different initial buffer time values under the tandem nodes scenario of the LMS.

Figure 4.15 considers a system with $Q_0 = 5$ packets. It depicts the trend that when the initial buffer time increases, the average display latency of LMS increases. We also observe that as the initial buffer time is small (e.g. B_0 = 0.02 seconds), the average display latency remains almost constant even when the initial buffer size increases. This is because the initial buffer time is reached first and the node starts to playout packet.

4.4.3. Loss Probability

In this section, we present the loss probability performance (defined in Section 3.7) of the end-to-end model, the Ethernet model and the tandem nodes model. We have observed that in all these models the loss probability of LMS is an increasing function of the initial buffer size, e.g. $Q_0 = 3$ curve in Figure 4.16. This is as expected because as more packets

are buffered, more probabilities that the queue size reaches the queue threshold. Therefore, a displayer has to discard more packets by the algorithm, as a result, increasing the loss probability.

4.4.3.1. The End-to-end Model

The OPNET model of this end-to-end model has been discussed in the Section 2.2.1.2.

A. Effect of the Queue Threshold Q_0

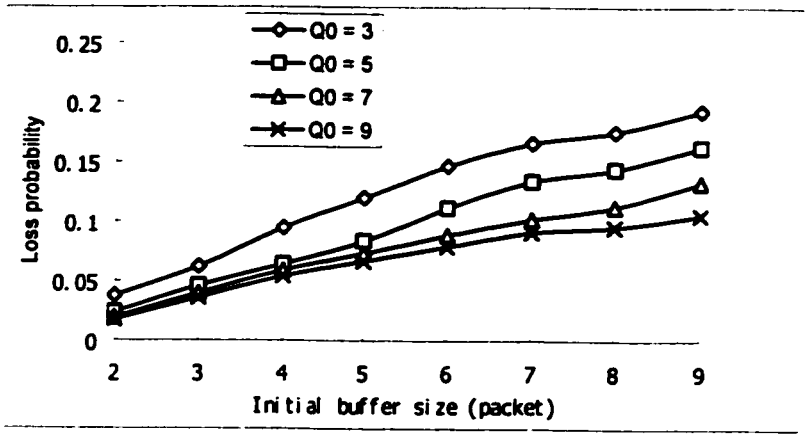

Figure 4.16. Loss probability vs. initial buffer size with different queue threshold values in the end-to-end scenario of the LMS.

Figure 4.16 considers a system with B_0 = 0.15 s. It depicts that under the end-to-end scenario, the queue threshold values have an impact on the loss probability. Generally, when the threshold value is large, the loss probability is small. This is as expected because for a given initial buffer size, as the queue threshold value is larger, a larger queue size is permitted. As a result, the node needs to discard fewer packets to keep the queue size below the threshold.

B. Effect of the Initial Buffer Time B_0

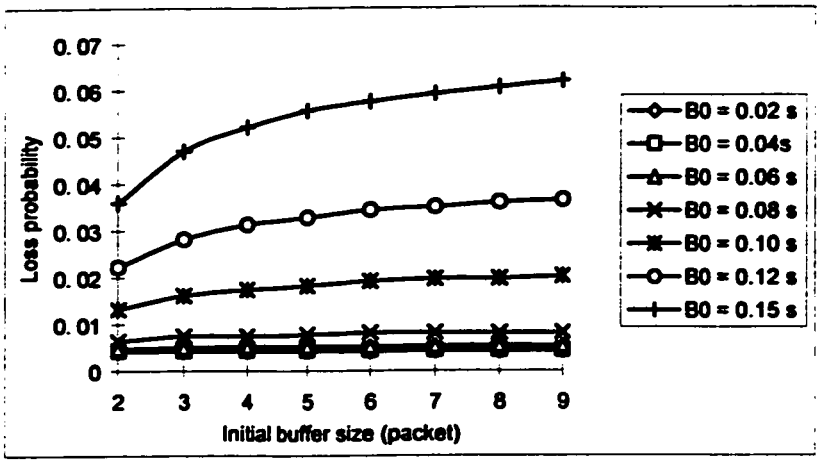

Figure 4.17. Loss probability vs. initial buffer size with different initial buffer time values in the end-to-end scenario of the LMS.

Figure 4.17 considers a system with $Q_0 = 5$ packets. It depicts the trend that when the initial buffer time increases, the loss probability of the LMS increases. This is as expected because when the initial buffer time is large, more packet are buffered, so more probability that the queue size reaches the threshold. Therefore, the node will discard more packets. We also observe that as the initial buffer time is small (e.g. $B_0 = 0.02$, 0.04, 0.06, and 0.08 seconds), the loss probability remains constant even when the initial buffer size increases. This is because the initial buffer time is reached first and the node starts to playout packet. In this case, to increase the initial buffer size has no impact on the loss probability.

4.4.3.2. The Ethernet Model

The OPNET model of this Ethernet model has been discussed in the Section 2.2.1.3.

A. Effect of the Queue Threshold Q_0

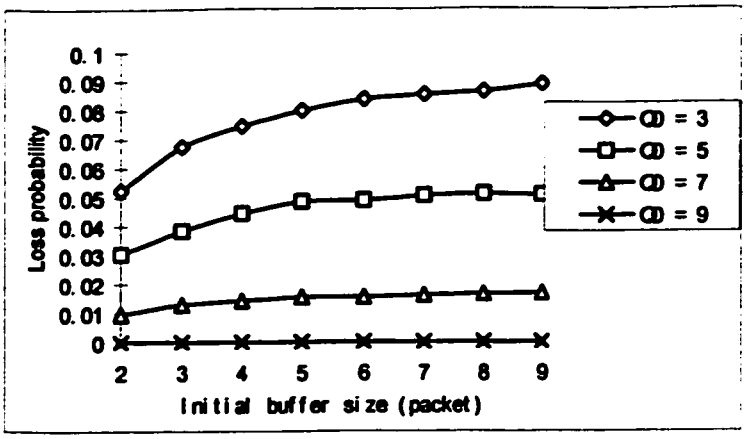

Figure 4.18. Loss probability vs. initial buffer size with different queue threshold values under the Ethernet scenario of the LMS.

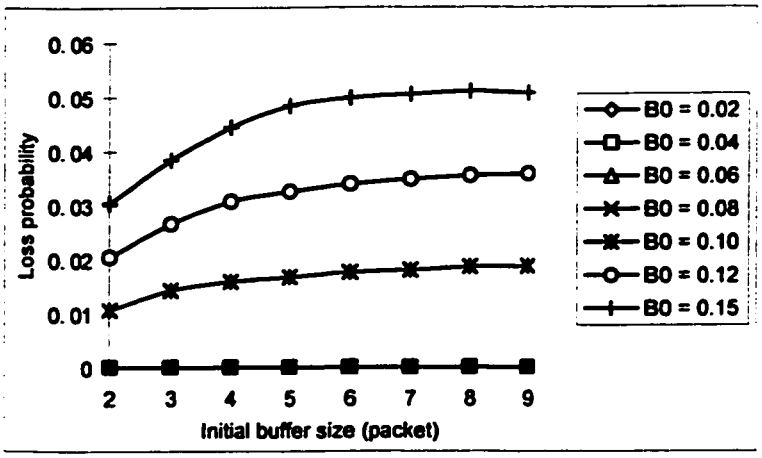

Figure 4.19. Loss probability vs. initial buffer size with different initial buffer time values under the Ethernet scenario of the LMS.

Figure 4.18 considers a system with B_0 = 0.15 s. It depicts that under the Ethernet scenario, the queue threshold values have impact on the loss probability. Generally, when the threshold value is large, the loss probability is small. This is as expected because for a given initial buffer size, the larger the queue threshold value, the larger is the queue size permitted, so the node needs to discard fewer packets (even no packet, e.g. Q_0 = 9 packets

in Figure 4.18) to keep the queue size below the threshold.

B. Effect of the Initial Buffer Time B_0

Figure 4.19 considers a system with $Q_0 = 5$ packets under the Ethernet scenario. It depicts the trend that when the initial buffer time increases, the loss probability of LMS increases. The reasons are the same as described under end-to-end scenario. We also observe that as the initial buffer time is small (e.g. B_0 = 0.02, 0.04, 0.06, and 0.08 seconds), the loss probability remains zero as the initial buffer size increases. This is because the initial buffer time is reached first and the queue sizes never reach the thresholds ($Q_0 = 5$ in this case) so the node never discards any packet.

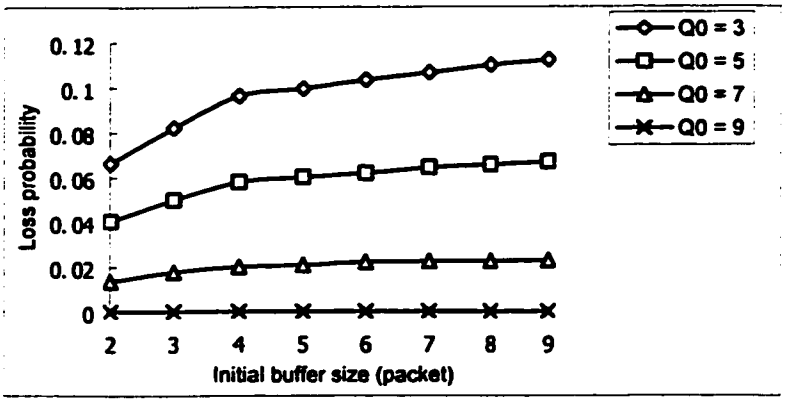

Figure 4.20. Loss probability vs. initial buffer size with different queue threshold values under the tandem nodes scenario of the LMS.

4.4.3.3. The Tandem Nodes Model

The OPNET model of this Ethernet model has been discussed in the Section 2.2.1.4.

A. Effect of the Queue threshold Q_0

Figure 4.20 considers a system with B_0 = 0.15 s. It depicts that under the tandem nodes scenario, the queue threshold values have impact on the loss probability. Generally, when

the queue threshold value is large, the loss probability is small. This is as expected because for a given initial buffer size, the larger the queue threshold value, the larger is the queue size permitted, so the node needs to discard fewer packets (even no packet as $Q_0 = 9$ packets in Figure 4.20) to keep the queue size below the threshold.

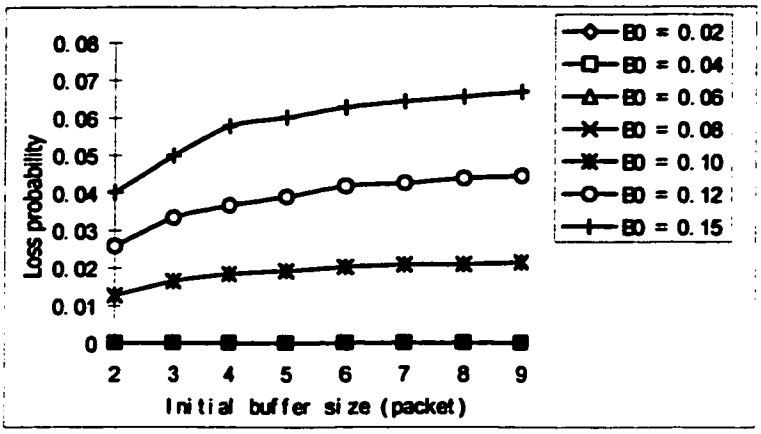

Figure 4.21. Loss probability vs. initial buffer size with different initial buffer time values in the tandem nodes scenario of the LMS.

B. Effect of the Initial Buffer Time B_0

Figure 4.21 considers a system with $Q_0 = 5$ packets under tandem nodes scenario. It depicts the trend that when the initial buffer time increases, the loss probability of LMS increases. The reasons are the same as described under end-to-end scenario. We also observe that as the initial buffer time is small (e.g. B_0 = 0.02, 0.04, 0.06, and 0.08 seconds), the loss probability remains zero as the initial buffer size increases. This is because the initial buffer time is reached first and the queue sizes never reach the thresholds ($Q_0 = 5$ in this case) so the node never discards any packet.

4.5. Concluding Remarks

Based on the simulation results, we conclude that the gap probability of LMS is a decreasing function of the initial buffer size. The queue threshold Q_0 has impact on the gap probability. As the queue threshold values are large, the gap probabilities are small. The initial buffer time B_0 also has impact on the gap probability. As it increases, the gap probability of LMS decreases. Also as the initial buffer time is small, the gap probability drops slowly and remains constant even when the initial buffer size increases. This is because the initial buffer time is reached first and the node starts to playout packet, so in this case, to increasing the initial buffer size has no impact on the gap probability. Note that the gap probability of the LMS in the end-to-end model is larger than the ones in the Ethernet and tandem nodes models. This is because in the end-to-end model, the network delay is set larger than the ones in the other two models. As a result, there is a larger end-to-end delay variation in the end-to-end model, and therefore a larger gap probability.

We can also conclude that the average display latency is an increasing function of the initial buffer size. Increasing the queue threshold or the initial buffer time will lead to the increasing the average display latency. We also observe that as the initial buffer time is small (e.g. B_0 = 0.02 seconds), the average display latency remains almost constant even when the initial buffer size increases. This is because the initial buffer time is reached first and the node starts to playout packet. Note that the average display latency of the LMS in the end-to-end model is larger than the ones in the Ethernet and tandem nodes models. This is because in the end-to-end model, the network delay is set larger than the ones in the other two models. As a result, the receiver node in the end-to-end

model has to wait longer to buffer enough packets, and therefore there is a larger average display latency.

Finally, we conclude that as either the initial buffer size or the initial buffer time increases, the loss probability of LMS increases. As the initial buffer time is small, the loss probability remains zero even as the initial buffer size increases. This is because the initial buffer time is reached first and the queue sizes never reach the thresholds, so the node never discards any packet. The queue threshold values also have impact on the loss probability that as it increases, the loss probability decreases (the minimum is zero).

Chapter 5

The Fuzzy Logic Method

We would like to investigate the fuzzy logic method here as a different paradigm approach. Fuzzy logic is widely used in the industry since it is very similar to the human control logic. We apply the Fuzzy Logic Algorithm (FLA) at the destination end to control the level of the buffer size at its optimum level. In this chapter, we shall give the mathematical definitions of the method to solve our problem. Then we explain the procedures and the flowchart of the operation of the fuzzy logic controller. Finally, we use OPNET simulations to evaluate our fuzzy logic algorithm against benchmark-algorithms.

5.1. General Operation and Definitions

Our Fuzzy logic controller consists of three processes:

1. A *Fuzzification* process, which is a mapping process that converts the measured input values (e.g. gap probability) to the fuzzy set of linguistic input values (e.g. "small").
2. An *inference* process, which is a rule-based structure that defines the fuzzy output (e.g. "Positive") in response to the linguistic fuzzy inputs (e.g. "small").
3. A *defuzzification* process, which calculates the crisp output (e.g. to increase initial buffer time by 34.52%) in response to the outputs from the inference process (e.g. "Positive").

We shall define three membership functions to be used later. The first two are

input membership functions. They are called the "buffer size error", and the "buffer size error rate", and as shown in Figure 5.1 and Figure 5.2 respectively. The third one is an output membership function called the "Initial buffer size adjustment" as shown in Figure 5.3. Associated with each function is a set of "linguistic values" called the fuzzy variables. "For simplicity, we shall use the triangles to represent the membership functions, and give the details of each one of them below.

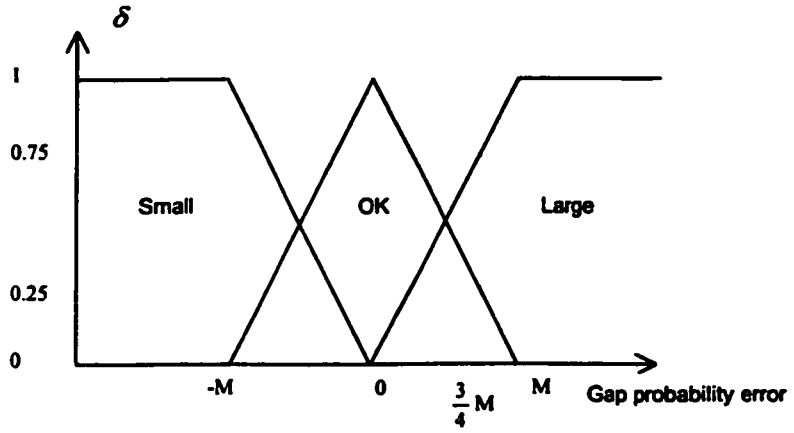

Figure 5.1. The "buffer size error" membership function.

5.1.1. The "Buffer Size Error" Membership Function

Let X1 be a fuzzy variable, and L(X1) be the set of fuzzy variables. We also define the output value $\delta \in [0,1]$ along the ordinate to be the degree of membership of a certain element in the fuzzy set L(X1). For example, the fuzzy set L(X1) in Figure 5.1 contains 3 fuzzy variables, i.e. L(X1) = { Small (S),Ok (O), Large (L) }. The fuzzy variable "Small" indicates that the initial buffer size is too small. "OK" indicates that the initial buffer size is good and there is no need to change it while "Large" indicates that the initial buffer size is large. In Figure 5.1, the horizontal axis is the input parameter $\hat{B}$, which is the gap

probability error and defined as the difference between the target gap probability and the measured gap probability. The parameter M is the pre-defined positive threshold. The function defines the degree of membership ($0 \le \delta \le 1$) of each member for a certain $\hat{B}$. For example, as shown in Figure 5.1, when $\hat{B} = \frac{3}{4}M$, we can get $\delta = 0$, 0.25, and 0.75 as the degrees of the membership of "Small", "OK", and "Large", respectively.

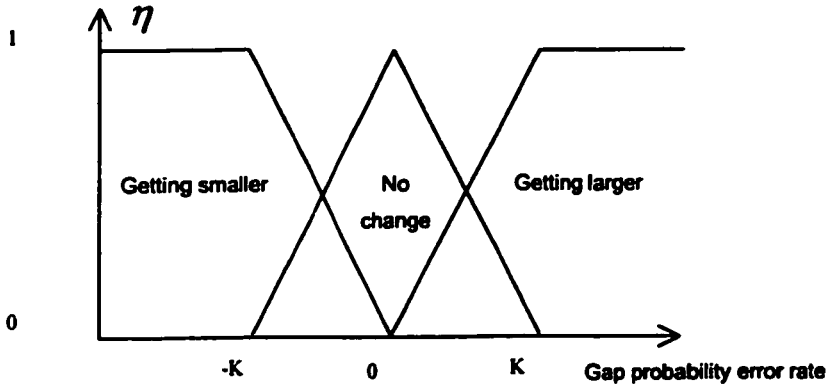

Figure 5.2. The "buffer size error rate" membership.

5.1.2. The "Buffer Size Error Rate" Membership Function

Let X2 be a fuzzy variable, and L(X2) be the set of fuzzy variables. We also define the output value $\eta \in [0,1]$ along the ordinate to be the degree of membership of a certain element in the fuzzy set L(X2). For example, the fuzzy set L(X2) in Figure 5.2 contains 3 fuzzy variables, i.e. L(X2) = { Getting smaller (GS), No changing (NC), Getting larger (GL) }. The fuzzy variable "Getting smaller" indicates that the initial buffer size is getting smaller. "No change" indicates that the initial buffer size is not changing while "Getting larger" indicates that the initial buffer size is getting larger. In Figure 5.2, the horizontal axis is the input parameter "gap probability error rate" and defined as the

changing rate of $\hat{B}$, i.e. $\frac{\hat{B}}{\Delta t}$, where Δt is the duration of one test session (i.e. the time for displaying every N packets). The parameter K is the pre-defined positive threshold. Like Figure 5.2, the degree of membership ($0 \le \eta \le 1$) can be readily determined by various membership function as a function of $\frac{\hat{B}}{\Delta t}$.

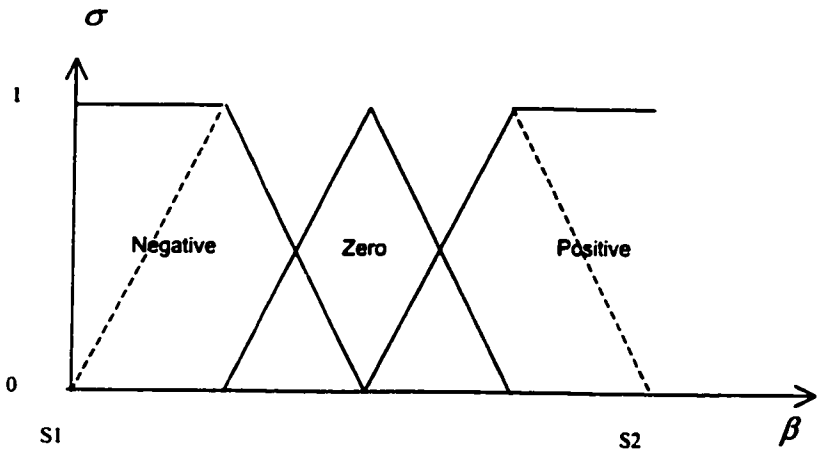

Figure 5.3. Output Membership function.

5.1.3. "Initial Buffer Size Adjustment" Output Membership Function

Let X3 be a fuzzy variable, and L(X3) be the set of fuzzy variables. We also define the output (horizontal) value β as the buffer change factor to be the centeriod of the area constructed by the degree of membership of a certain element in the fuzzy set L(X3). For example the fuzzy set L(X3) in Figure 5.3 contains 3 fuzzy variables: "Negative", "Zero", and "Positive", i.e. L(X3) = { Negative (N), Zero (Z), Positive(P) }.The fuzzy variable "Negative" stands for calling for decreasing the initial buffer size. "Zero" stands for no change on the buffer size, while "Positive" stands for calling for increasing the

initial buffer size. We also define the input (vertical) value σ, $\sigma \in [0,1]$ is from the result of the inference of the two inputs' degree of membership. The parameters S1 and S2 are the pre-defined minimum and maximum response limits, respectively.

5.2. Fuzzification, Inference and Defuzzification

Having explained the membership functions, we now provide the operation of a few important processes required in the FLA.

5.2.1 Fuzzification

Fuzzification is the process of determining the membership degree from measured values. Fuzzification at the destination end involves measuring the gap probability and comparing it with the reference. The node calculates $\hat{B}$ (i.e. the difference between target gap probability and measured gap probability) and $\frac{\hat{B}}{\Delta t}$, and uses these two values as the inputs for the two membership functions (the "buffer size error" and the "buffer size error rate") in order to map and obtain the membership degree of the fuzzy variables of the fuzzy set L(X1) and L(X2).

5.2.2 Inference

Inference allows certain decisions to be made based on the inputs. Since we have three fuzzy variables in each input fuzzy set L(X1) and L(X2), we can define nine rules corresponding to the nine combinations of these two sets of variables. A typical rule is

IF *A* AND *B* THEN *C*,

where $A \in$ L(X1), $B \in$ L(X2), $C \in$ L(X3), and AND is the logical product. For example, A = "Small", B = "Getting smaller", C = "Positive", in other words, if the buffer size error $\hat{B}$ is small and the buffer size changing rate $\frac{\hat{B}}{\Delta t}$ is getting smaller, then we infer that we should increase the initial buffer size.

5.2.3 Defuzzification

Defuzzification allows us to get the crisp output for the fuzzy controller by combining the results of the inference process and determining the weights of each input contributed to the final crisp output. In our study, we shall use the Root Sum Square (RSS) method to combine the effects of all rules and calculate the membership degree R of the output fuzzy set members. For example, the result R corresponding to "Positive" can be expressed as $R_{Positive} = \sqrt{\sum_{i=1}^{k} \omega^2 i}$, $\omega_i \in L(X3)$. The parameter ω_i is the result associated with "Positive" from the inference rules, and k is the number of rule results associated with "Positive". The $R_{Negative}$ and R_{Zero} are likewise defined for $X1$ and $X2$ separately. We choose RSS because there are few member functions contributed to the inputs and output. After we get the degree of each membership R_i, $i \in \{Positive, Negative, Zero\}$, we use the "Fuzzy Centroid" method to calculate the "centroid" (i.e. the center of gravity) of the membership function for the fuzzy variables. Mathematically, the centroid is computed as $\beta = \frac{S_1 \times R_{Negative} + 0 \times R_{zero} + S_2 \times R_{Positive}}{R_{Negative} + R_{zero} + R_{Positive}}$. The buffer change factor β will be utilized in out algorithm in the next section.

5.3. Adaptive Adjustment and Algorithm Flowchart

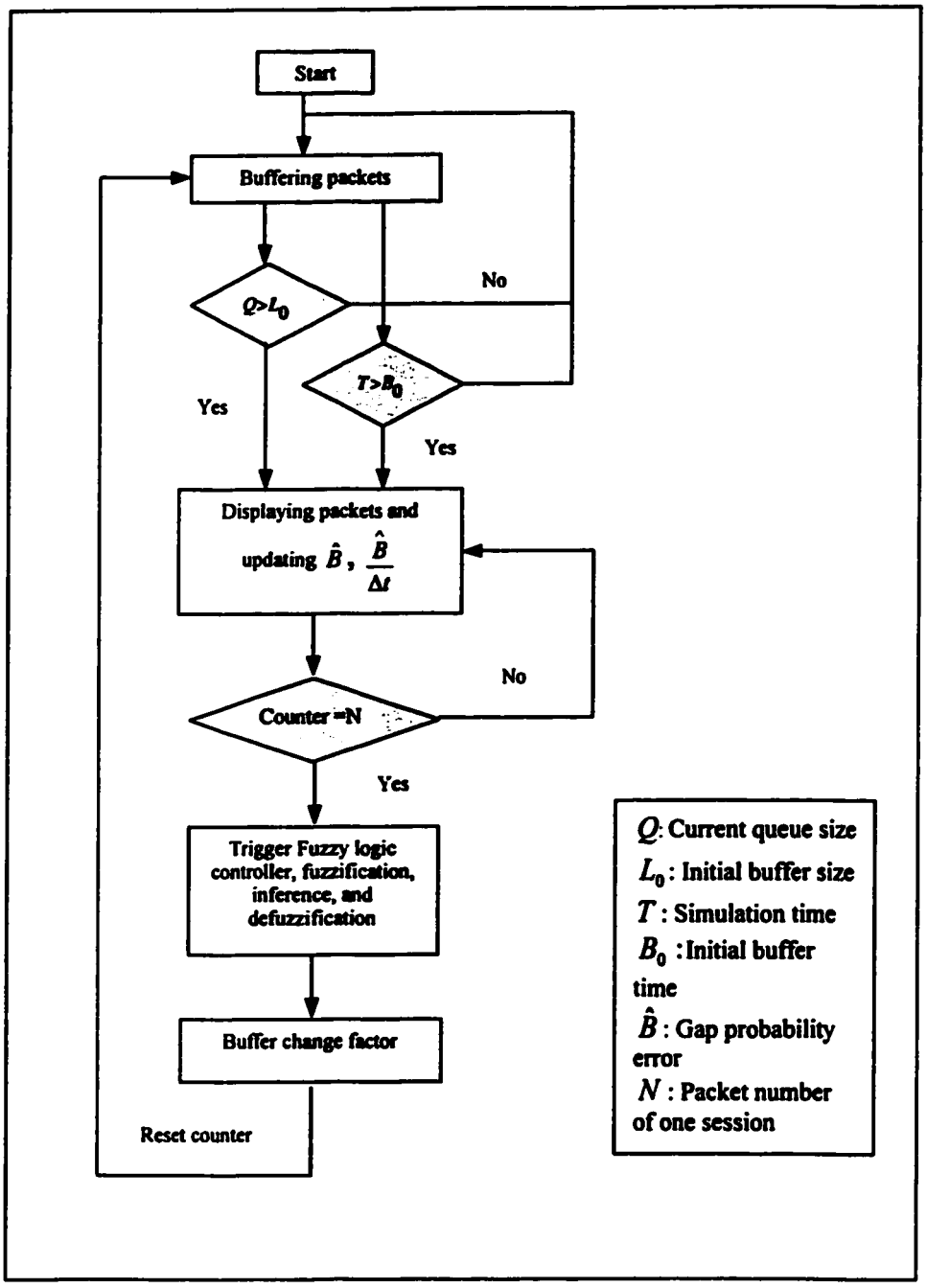

Figure 5.4. Flowchart of Fuzzy Logic Algorithm.

Consider a test session, consisting of several talk spurts. During each test session, the

node will wait for the late coming packets, like the E-Policy, and the gap probability is measured and compared with the pre-defined reference gap probability. After the fuzzification process, the inference process and the defuzzification process, we obtain the output of the fuzzy controller as response. The response is crisp because it gives a clear answer on how much of the initial buffer time to increase, or to reduce. In this way, the destination node is able to decide when and how much to self-adjust the initial buffer time. As a result, the gap probability is maintained around the level of the pre-defined reference level, which can be chosen and defined by an expert, while being quite tolerable to the audience. On the other hand, given a certain level of gap probability, the node will keep the display latency as small as possible.

The flowchart of the algorithm is shown in Figure 5.4. At the beginning, when the initial buffer size threshold (L_0) is reached or when the initial buffer time threshold (B_0) is reached, whichever comes first, the node starts to display the packets. In the first test session, after the buffer change factor is calculated, it is used to modify the initial buffer time of next test session. In the mean time, the counter is reset to zero. Since then, the node only buffers packet according to the initial buffer time, not the initial buffer size any more. For example, suppose $L_0 = 3$ packets is reached before $B_0 = 29.00$ ms. Then the node would start displaying at t = 25 ms. Then in the first test session, the node would change B_0 from 29 ms to 25 ms. Now suppose that after the first test session, the node has obtained the buffer change factor $\beta = 0.37$, then the node would update the next initial time to $B_0 = (1+\beta)\hat{B}_0 = (1+ 0.37) * 25 = 34.25$ ms from its old value $\hat{B}_0 = 25\ ms$, and the 34.25 ms would be the initial buffer time for all talk spurts to come in that

session.

5.4. Performance Evaluation

Simulations program were written to evaluate our FLA algorithm. They were run on a PC with a Pentium III 1GHz CPU and 1 G byte memory. The operation system is Windows 2000 Professional. The release version of OPNET modeler is 7.0.B (Build 1109). A typical simulation run collects the statistics of about 66800 packets and takes about 10 minutes.

We use the same parameters as our LMS experiment. That is, we set the packet interarrival time to be constant at 0.02 second. The packet size equals to 1280 bits, and the talk spurt interarrival time (i.e. silence period) is exponentially distributed with a mean of 0.65s, and the talk spurt size is exponentially distributed with a mean of 22400 bits. The initial buffer time threshold B_0 has a default value of 0.15s, and the initial buffer sizes L_0 is set to 5 packets. The target (reference) gap probability is set to be 4%. Note that it is possible to set it lower, e.g. 2%, however, as a tradeoff, the average display latency is larger, we therefore set it to be 4%, which is quite acceptable while keeping the latency low. The session packet number N equals to 200. The width of the "OK" of the "Buffer size error " membership function M has a default value of 0.02, but can be promoted (i.e. changed before each simulation). Mathematically, we desire $|\textit{measured gap probability error} - 0.04| \leq 0.02$ for the "OK" region in Figure 5.1. The width of the "No change" of the "Buffer size error rate" membership function K has a default value of 0.03, but can also be promoted. The width of the "Positive" of the output

function S_2 is set to 100, and the width of the "Negative" of the output function S_1 equals to −100. In the end-to-end scenario, the network constant delay component is set to be 6 ms and the network random delay component is exponentially distributed with a mean of 15ms. In the tandem nodes scenario, for each tandem node, the network constant delay component is set to be 0.1 ms and the network random delay component is exponentially distributed with a mean of 1ms.

During simulation, we found that most of the experiment curves were sensitive to the seed of the simulation. Therefore, we have to obtain the 95% confidence interval of these simulations. Each simulation point is obtained as an average of 11 simulation runs (i.e. 11 different seeds). A typical performance curve with 5 points takes about 8 hours.

We have run simulations under the three different network scenarios. In the following sections, we will present the results and reveal the relations among the width parameters M, K, the gap probability and the average display latency. The purpose of these simulations is to confirm good ranges for M and K, so that in the later chapter when we compare the performance of the FLA with other algorithms, we shall be able to sure to pick up the right parameters from the right ranges. Other parameters like N, S_1, and S_2 are not important compared with M and K. They mainly decide the "strength" and "how often" the node will control the buffer. A systematical study of their impacts on the performance is omitted due to our limited resource. Notice that FLA always has zero loss probability because FLA waits for the late coming packets, as the E-Policy does.

5.4.1. Gap Probability

In this section, we present the gap probability performance (defined in Section 3.5.1) of the end-to-end model, the Ethernet model and the tandem nodes model.

5.4.1.1. The End-to-end Model

The OPNET model of this end-to-end model has been discussed in the Section 2.2.1.2.

A. The Effect of the Width Parameter M

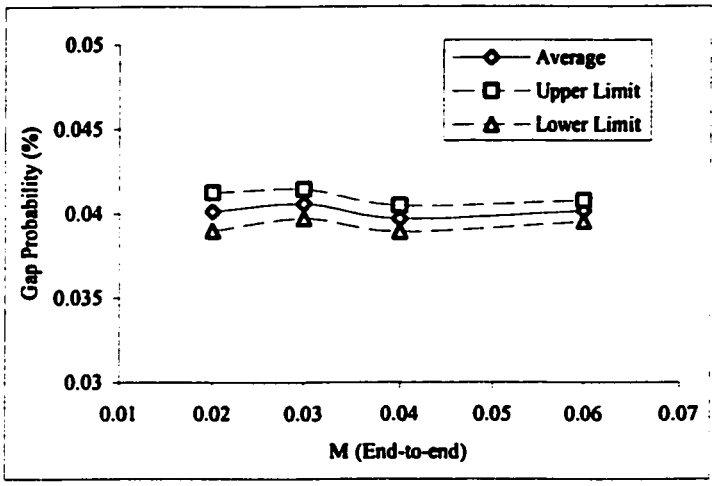

Figure 5.5. The gap probability versus M in the end-to-end model of the FLA.

As shown in Figure 5.1, the parameter M is the width of "OK" of the "Buffer size error" membership function. In order to find its impact on the FLA performance, we construct simulations based on different values of M. Figure 5.5 considers a system with K = 0.03. It depicts the 95% confidence interval of the gap probabilities of the FLA fluctuates slightly around 0.04 as M increases. However, this fluctuation is not significant and we can therefore regard it as constant. This result is expected because the FLA controller controls the buffer to maintain the gap probability around the reference gap probability, which is pre-defined as 4% before simulations.

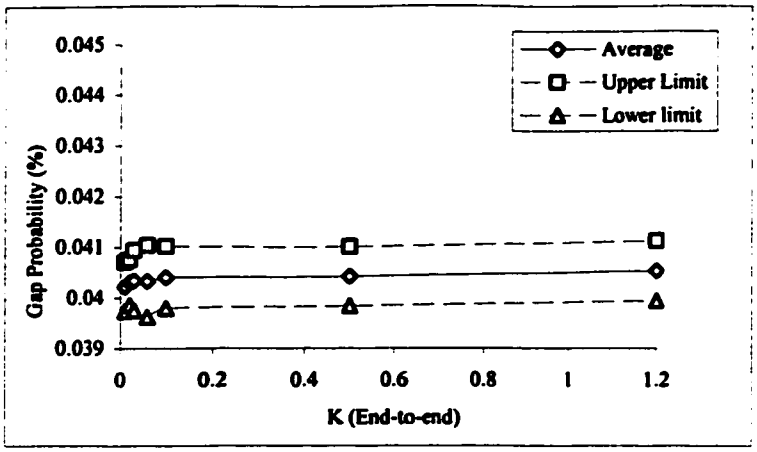

Figure 5.6. The gap probability versus K in the end-to-end scenario of the FLA.

B. The Effect of the Width Parameter K

As shown in Figure 5.2, the parameter K is the width of "No change" in the "Buffer size error rate" membership function. We have also run simulations in order to find optimum range of K. Figure 5.6 considers a system with M = 0.02. It depicts that the gap probability sometimes fluctuates as K increases. However, the 95% confidence interval shows that the average value of K is almost constant around 0.04.

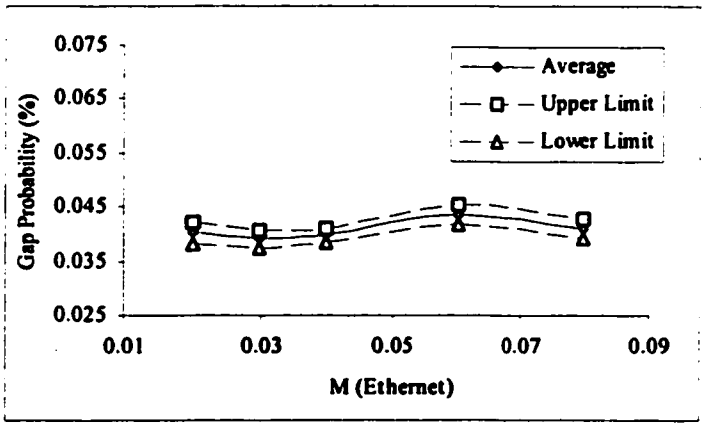

Figure 5.7. The gap probability versus M in the Ethernet scenario of the FLA.

5.4.1.2. The Ethernet Model

The OPNET model of this Ethernet model has been discussed in the Section 2.2.1.3.

A. The Effect of the Width Parameter M

Figure 5.7 considers a system with K = 0.03. The 95% confidence interval depicts that under Ethernet scenario, the gap probabilities of FLA fluctuate around 0.04 and can be regarded as constant as M increases. This is similar to the result in the end-to-end model.

B. The Effect of the Width Parameter K

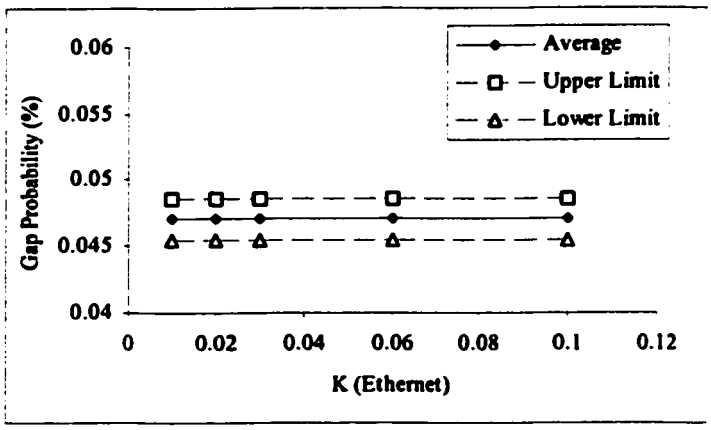

Figure 5.8. The gap probability versus K in the Ethernet scenario of the FLA.

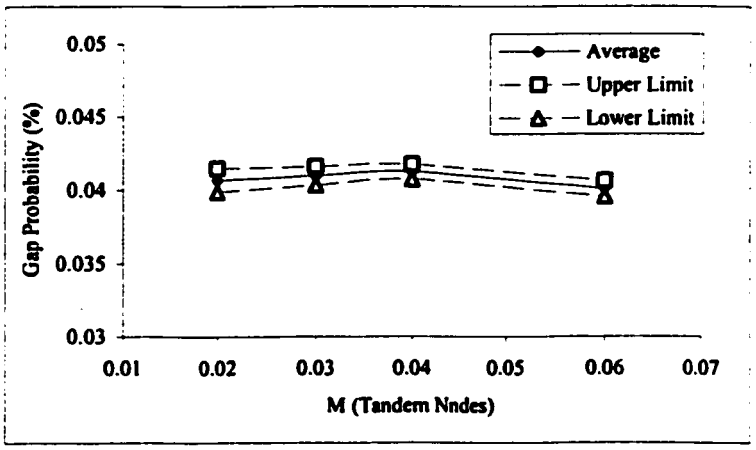

Figure 5. 9. The gap probability versus M in the tandem nodes scenario of the FLA

Figure 5.8 considers a system with M = 0.03. The 95% confidence interval depicts that the gap probability remain constant as K increases in the Ethernet model.

5.4.1.3. The Tandem Nodes Model

The OPNET model of this tandem nodes model has been discussed in the Section 2.2.1.4.

A. The Effect of the Width Parameter M

Figure 5. 9 considers a system with K = 0.03, The 95% confidence interval depicts that in the tandem nodes scenario, the gap probabilities of FLA fluctuates around 0.04 as M increases. However, the fluctuation is not significant so that we can also regard it as constant.

B. The Effect of the Width Parameter K

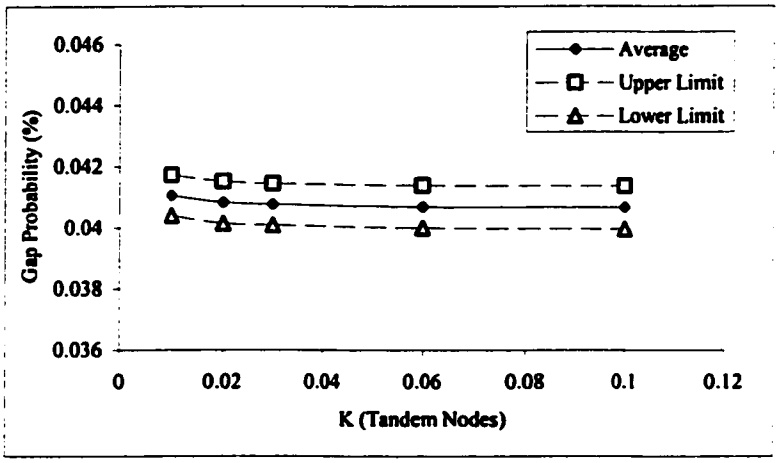

Figure 5.10. The gap probability versus K in the tandem nodes scenario of the FLA.

Figure 5.10 considers a system with M = 0.02. The 95% confidence interval depicts that in tandem nodes scenario, the gap probability decreases slightly as the K increases. However, the gap probability can be regarded as constant (around 0.0407) since the decreasing is not significant.

5.4.2. Average Display Latency

In this section, we present the average display latency performance (defined in Section 3.6) of the end-to-end model, the Ethernet model and the tandem nodes model.

5.4.2.1. The End-to-end Model

The OPNET model of this end-to-end model has been discussed in the Section 2.2.1.2.

A. The Effect of the Width Parameter M

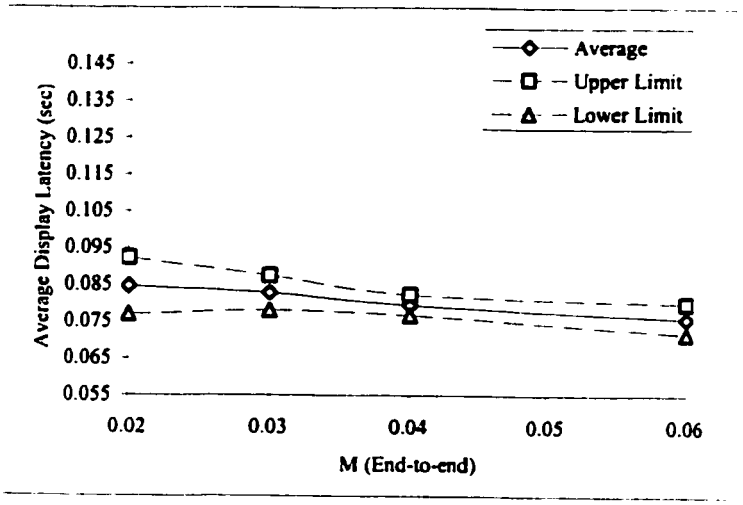

Figure 5.11. The average latency versus M in the end-to-end scenario of the FLA.

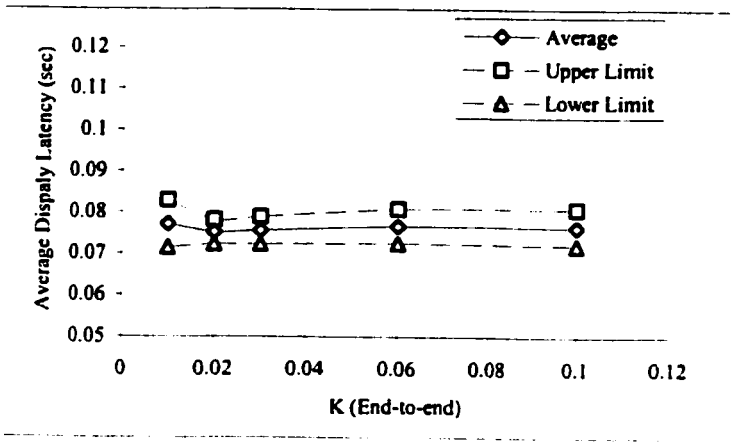

Figure 5.12. The average latency versus K in the end-to-end scenario of the FLA.

Figure 5.11 considers a system with K = 0.03. The 95% confidence interval depicts that the average display latency of FLA in the end-to-end model decreases slightly as M increases. However, this decreasing is not significant and we can regard it as constant.

B. The Effect of the Width Parameter K

Figure 5.12 considers a system with M = 0.02. The 95% confidence interval depicts that the average display latency of FLA in the end-to-end model fluctuates slightly as K increases. However, this fluctuation is not significant and we can regard it as constant.

5.4.2.2. The Ethernet Model

The OPNET model of this Ethernet model has been discussed in the Section 2.2.1.3.

A. The Effect of the Width Parameter M

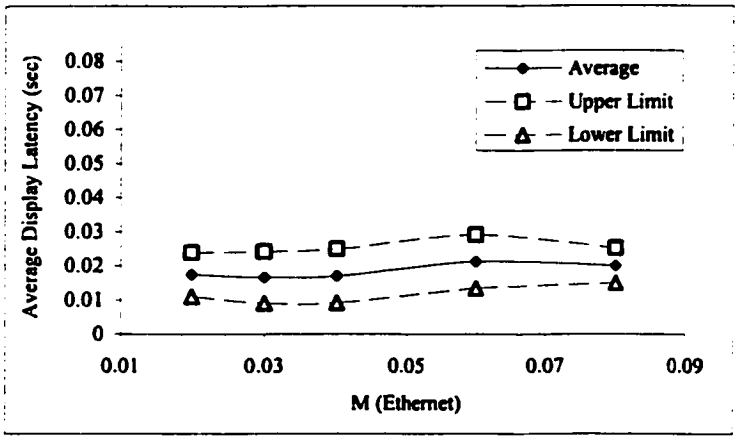

Figure 5.13. The average latency versus M in the Ethernet scenario of the FLA.

Figure 5.13 considers a system with K = 0.03. The 95% confidence interval depicts that under Ethernet scenario, as M increases, the average display latency of FLA fluctuates slightly and can also be regarded as constant.

B. The Effect of the Width Parameter K

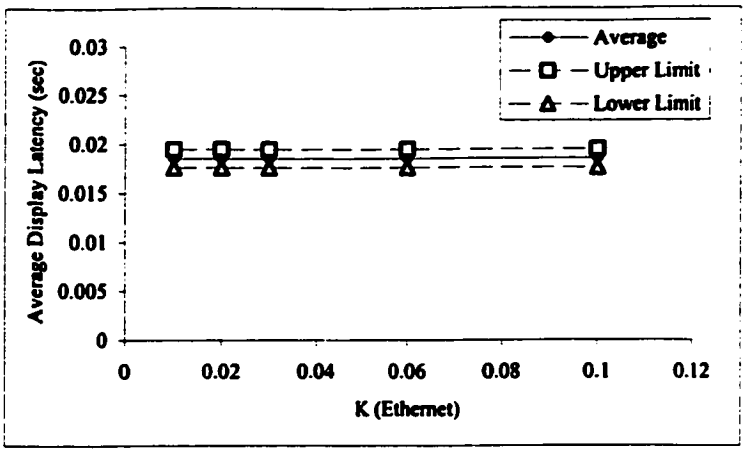

Figure 5.14. The average latency versus K in the Ethernet scenario of the FLA.

Figure 5.14 considers a system with M = 0.02. The 95% confidence interval depicts that under the Ethernet scenario, the average latency remains constant as K increases.

5.4.2.3. The Tandem Nodes Model

The OPNET model of this tandem nodes model has been discussed in the Section 2.2.1.4.

A. The Effect of the Width Parameter M

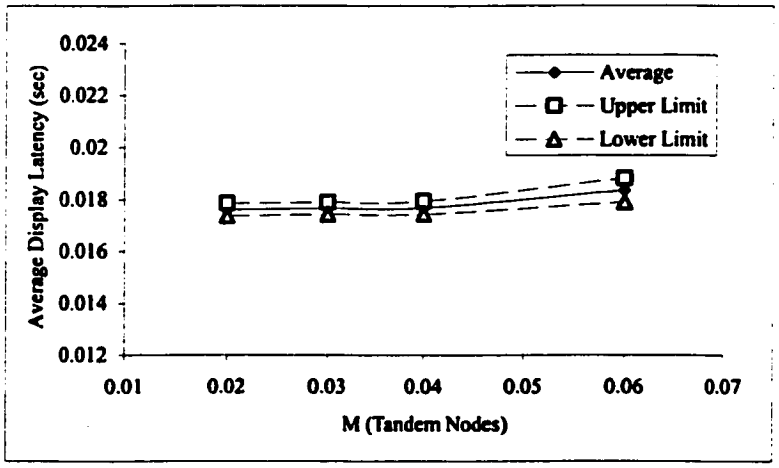

Figure 5.15. The latency versus M in the tandem nodes scenario of the FLA.

Figure 5.15 considers a system with K = 0.03. The 95% confidence interval depicts that

in tandem nodes scenario, the average display latency of FLA increases slightly as M increases. However, this increasing is not significant that we can regard it as constant.

B. The Effect of the Width Parameter K

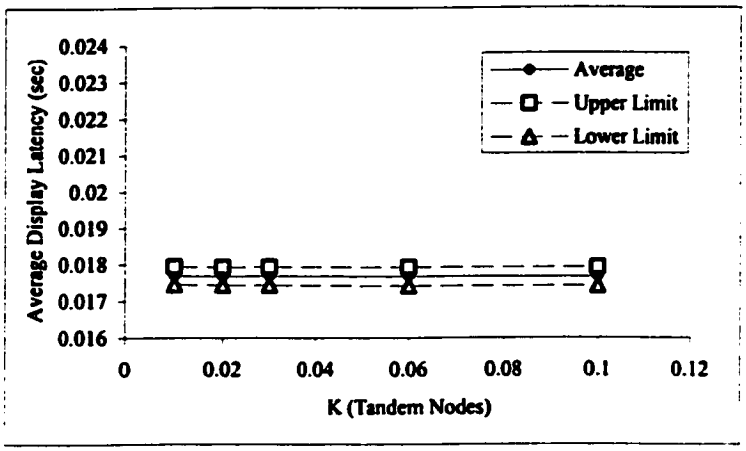

Figure 5.16. The average latency versus K in the tandem scenario of the FLA.

Figure 5.16 consider a system with M = 0.02. The 95% confidence interval depicts that the average latency remains constant as K increases.

5.5. Concluding Remarks

Our simulations reveal that M and K appear to have minimal impacts on the performance for the regions we have investigated. On the other hand, we notice that their impacts on the gap probability and the average display latency are sensitive on the simulation seeds. The 95% confidence interval reveals that when they are within certain range, the FLA will have the desired performance, i.e. both the gap probability and the average display latency are almost constant within the acceptable levels. The simulation results confirm that for the three network models, the good range for M is from 0.02 to 0.06, and the good range for K is from 0.02 to 0.1. The results show that if M and K are picked up

values from these ranges, the gap probability and the average display latency will remain constant, or fluctuate slightly around constant values. This is expected because the purpose of the FLA is to control the gap probability around the desired reference value, which is a constant pre-defined value. These simulations will lead us to find desired parameters of FLA for its implementation.

Chapter 6

Performance Comparison and Design Guideline

Since we have proposed and evaluated two new different algorithms of controlling jitter at the destination end, it would be of interest to compare their performance under different scenarios with other benchmark-algorithms. Towards the end of the chapter, we would also offer some design guideline based on our experience.

We have run the simulations for different algorithms under the three network models with the same parameters as we have used in the performance evaluations on the LMS and the FLA. Those parameters are either fixed throughout the simulations of the thesis or set to the default values, which are described in the Section 4.4 and 5.4.

6.1. Gap Probability Performance Comparison

6.1.1. The End-to-end Model

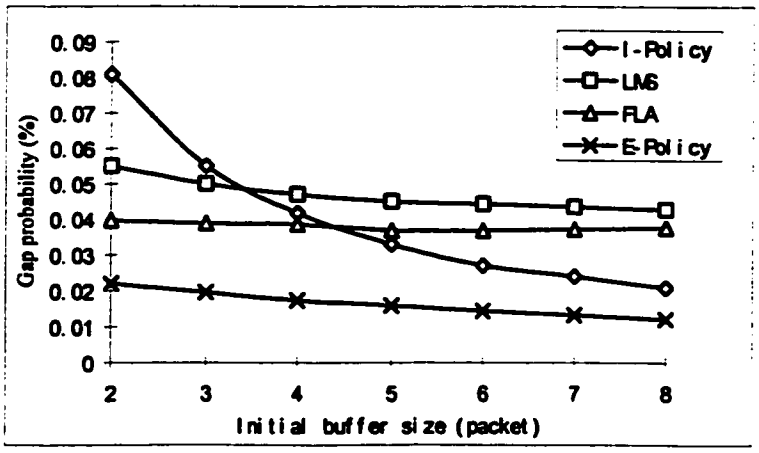

Figure 6.1. Gap probability vs. initial buffer size in the end-to-end scenario.

Figure 6.1 depicts that under end-to-end scenario the gap probability of the I-Policy, E-Policy and LMS algorithms decreases when the initial buffer size increases. This result is expected because when more packets are buffered, the less chance that the receiver will run out of packets, therefore, the less chance of the gap probability. The FLA has performed desirably to achieve an almost constant gap probability around the pre-defined reference gap probability (4%). The E-Policy is the lowest among the four algorithms. We also observe that the gap probability of the I-Policy is much more sensitive on the initial buffer size than the E-Policy, the LMS and the FLA. This is because the E-Policy, LMS and FLA algorithms have features to wait for the late coming packets. That means when the initial buffering is insufficient, the three algorithms may extend the initial buffering by waiting for the late packets.

6.1.2. The Ethernet Model

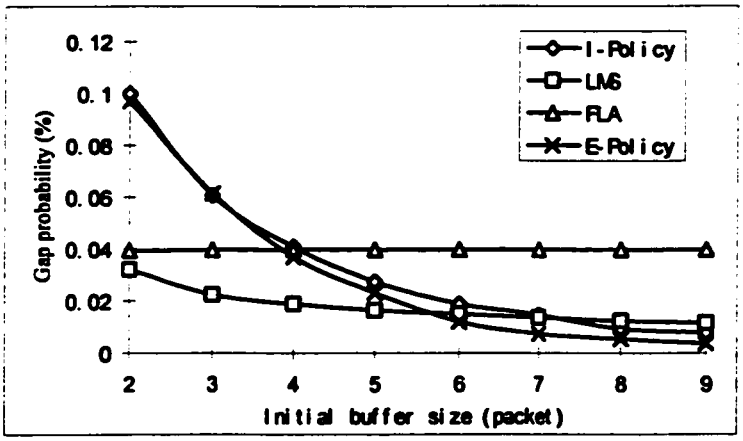

Figure 6.2. Gap probability vs. initial buffer size in the Ethernet scenario.

Figure 6.2 depicts that under the Ethernet scenario, similar to the end-to-end scenario, the gap probability decreases as the initial buffer size increases of the I-Policy, E-Policy

and LMS. The LMS has the lowest gap probability when the initial buffer size is less than 5 packets. The FLA tries to maintain a constant gap probability. Therefore, even though the FLA has the largest gap probability for initial buffer size above 4 packets, its level around 4% is quite acceptable. (In fact, it can be lowered.) When the initial buffer size is large (e.g. L_0 = 8 and 9 packets), the I-Policy and the E-Policy have lower gap probability than the LMS and FLA. However, as a tradeoff, they will suffer from larger average display latencies (beyond acceptable level), which will be shown in Section 6.2.2.

6.1.3. The Tandem Nodes Model

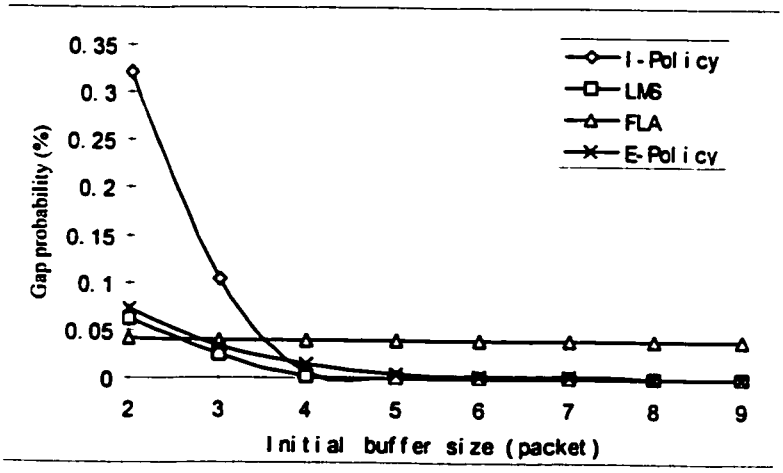

Figure 6.3. Gap probability vs. initial buffer size in Tandem node scenario.

Figure 6.3 depicts that under tandem nodes scenario, the gap probabilities of the I-Policy, E-Policy and LMS algorithms drop quickly as the initial buffer size increases and the values tend to zero quickly. The LMS outperform the other benchmark-algorithms by having lowest gap probabilities. The FLA again has a constant value within the

acceptable level.

Figures 6.1, 6.2 and 6.3 show that when the initial buffer size is small and moderate, the gap probability reduces significantly as initial buffer size increases. The gap probability of the I-Policy is the highest because of inadequate buffering and no waiting for the late packets. The gap probability of the FLA is constant in the three models because it adjusts the gap probability around the optimum level (4% in this case) whether the start-up initial buffer size is not enough or too much. Definitely we can set the target gap probability lower than 4% if the application requires. Although the E-Policy sometimes has a low gap probability, it will suffer from larger latency because it waits for the late packets, and has no mechanism to drop the queue size accumulated. The LMS outperforms other algorithms in the Ethernet and the tandem nodes models and is between the I-Policy and E-Policy in the end-to-end model, because it has the mechanism to reduce queue size based on prediction results. However, when the initial buffer size increases to relatively large, most packets are buffered in the I-Policy, E-Policy, and LMS algorithms, so there is less difference among them, and it approaches to zero. This is very obvious under tandem nodes scenario. Overall we think that the FLA is the best choice among other algorithm because its gap probability performance is quite stable against the changing network load under different environments.

6.2. Average Display Latency Performance Comparison

6.2.1. The End-to-end Model

Figure 6.4 shows that in the end-to-end model, the average display latencies of the I-

Policy and the E-Policy increase when the initial buffer size increases. The reason is that when the receiver holds more arrived packets, more extra delay will be introduced. Notice that the LMS has the lowest and nearly constant average display latency. It outperforms all other algorithms. This is because the LMS discards packet(s) when the queue size reaches the queue threshold. The E-Policy has the largest latency because of its "accumulation" phenomenon by waiting for the late packets. The FLA has the second lowest constant values and outperforms the two benchmark-algorithms.

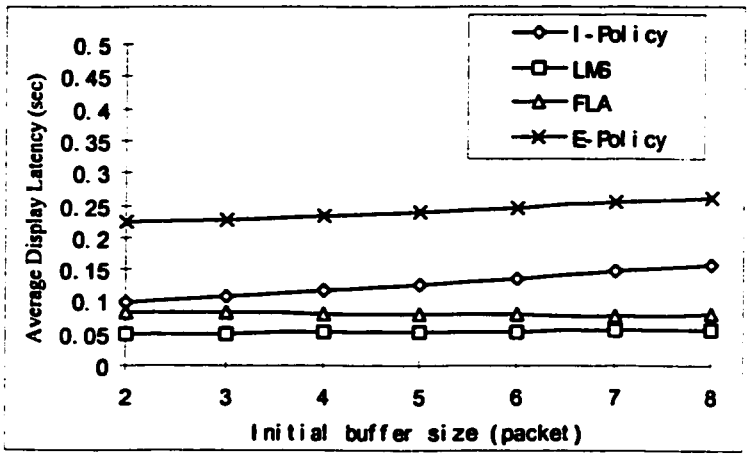

Figure 6.4. Average display latency vs. initial buffer size in the end-to-end model.

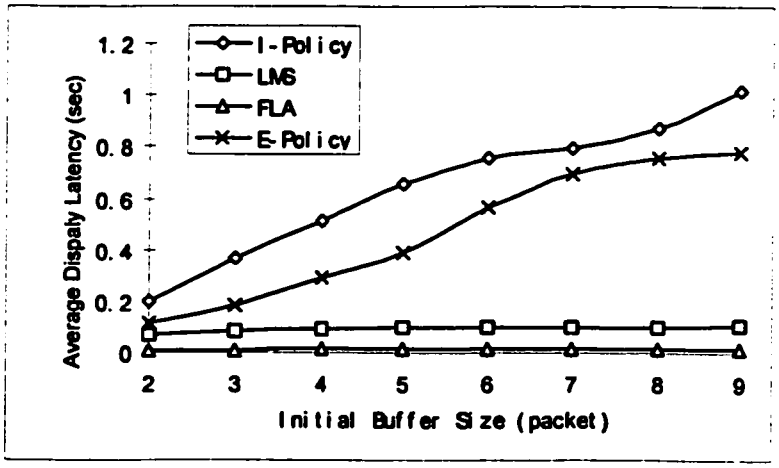

Figure 6.5. Average display latency vs. initial buffer size in the Ethernet model.

6.2.2. The Ethernet Model

Figure 6.5 shows that under Ethernet environment, the average display latency of the LMS does not change too much (under 100ms) and has the second lowest values. The FLA has the lowest almost constant values and surpasses others in performance. The I-Policy and the E-Policy have large latencies since they buffer too many packets and no way to drop the queue sizes.

6.2.3. The Tandem Nodes Model

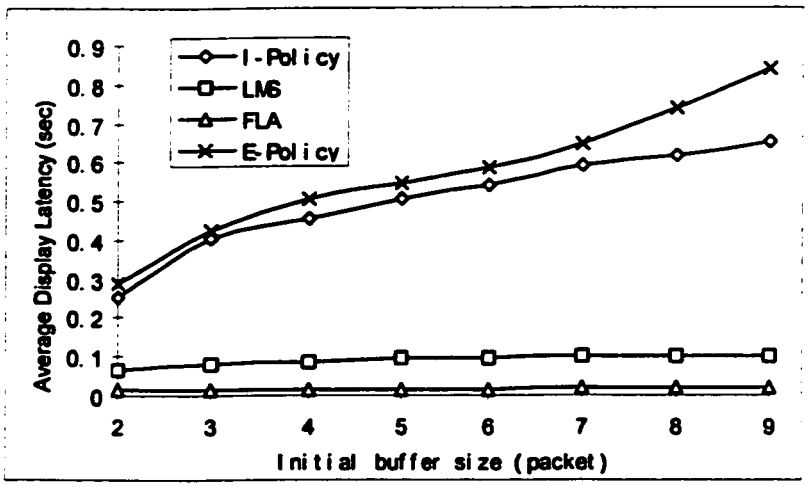

Figure 6.6. Average display latency vs. initial buffer size in the tandem nodes scenario.

Figure 6.6 depicts that under tandem nodes scenario, the average display latency of the I-Policy and the E-Policy increases as the initial buffer size increases. The FLA has the lowest and almost constant latency and E-Policy has the largest. The LMS has the second lowest latency and tends to be constant because the buffer size is kept around the threshold. The FLA outperforms others.

Figures 6.4, 6.5, and 6.6 show that as the initial buffer size increases, the node

buffers more packets, so the average display latencies of the I-Policy and the E-Policy increase. The E-Policy often has the largest value since it must wait for late packets and there is no mechanism to reduce the accumulated queue size. The FLA always has constant latency and outperforms others in the Ethernet and the tandem nodes models by having the lowest latency. The LMS has the lowest latency in the end-to-end model and has the second lowest latencies in the other two models. Generally, the FLA outperforms the other algorithms on the performance of the average display latency.

6.3. Loss probability Performance Comparison

6.3.1. The End-to-end Model

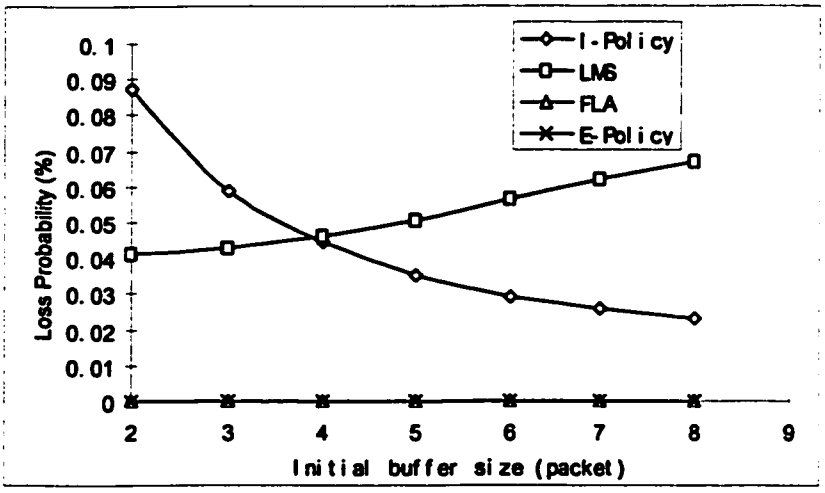

Figure 6.7. Loss probability vs. initial buffer size in the end-to-end scenario.

Figure 6.7 depicts that in the end-to-end model, the loss probability of the I-Policy decreases as the initial buffer size increases. This is because the I-Policy does not wait for the late packets, so when the initial buffer size threshold is big, fewer packets are considered as late, and fewer packets will be discarded when they arrive. On the

contrary, the loss probability of the LMS increases as the initial buffer size increases. This is because when the node buffers more packets, more chances that the queue size reaches the queue threshold. Therefore, by its definition, it will discard more packets. The E-Policy and the FLA have zero loss probability and outperform the other two, because by definitions they wait for the late coming packets and they do not discard received packets (unlike the LMS)

6.3.2. The Ethernet Model

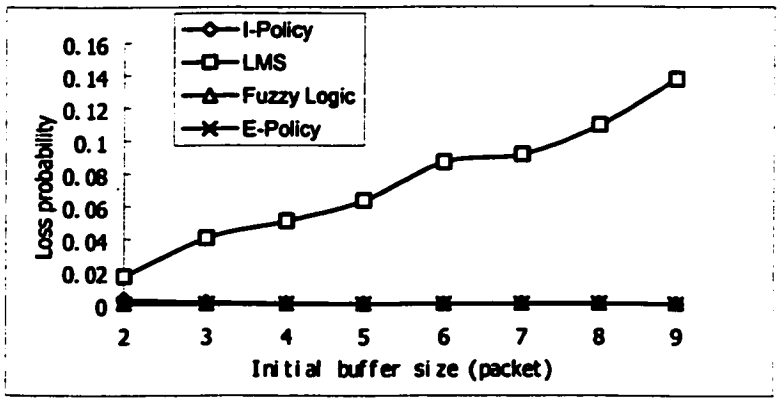

Figure 6.8. Loss probability vs. initial buffer size in the Ethernet scenario.

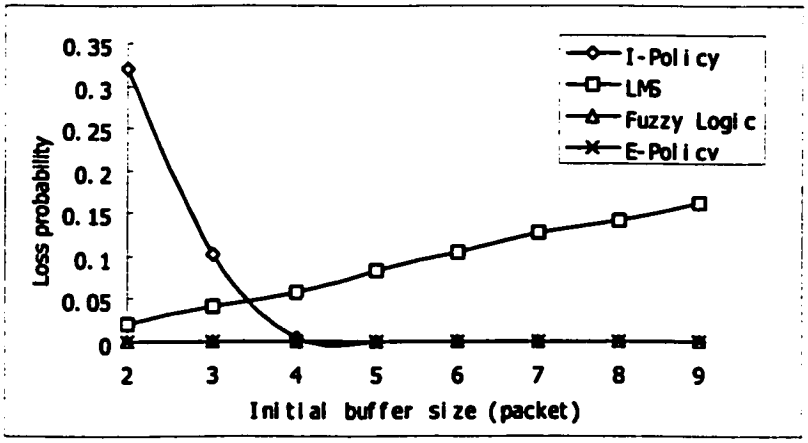

Figure 6.9. Loss probability vs. initial buffer size in the tandem nodes scenario.

Figure 6.8 shows clearly that under the Ethernet environment, the loss probability of the LMS increases as the initial buffer size increases. Again the E-Policy and the FLA have zero loss probabilities by definition and outperform the other two. Due to the large latency (shown in Figure 6.5), the I-Policy has very low loss probability in this case.

6.3.3. The Tandem Nodes Model

Figure 6.9 depicts that under tandem nodes model, the loss probability of LMS increases as the initial buffer size increases because more packets are need to be discarded if to keep the queue size below the threshold level. Whereas in the I-Policy, the loss probability decreases when the initial buffer size increases, since less packets are regarded as late and need to be rejected when they come. Again, the E-Policy and the FLA outperform the other two by having zero values since they wait for all the packets.

Figures 6.7, 6.8, and 6.9 show that when the initial buffer size increases, the node buffers more packets, and the loss probability reduces greatly in the I-Policy. On the other hand, the loss probability of LMS increases as it discards more packets in order to keep the buffer size below the threshold level. The E-Policy and the FLA have zero loss probabilities by definition and display the most complete information of the original voice content.

6.4. Comparison as the Network Random Delays are Normally Distributed

All the above simulations are based on the network random delays that are set as exponentially distributed. In order to verify the good performance of these two new

algorithms, we have run simulations and made performance evaluations based on the network random delays that have a Normal distribution. In the end-to-end scenario, the mean of the random network delay is 0.01 second, and the standard deviation is 0.005 second. In the tandem nodes scenario, the mean is 0.001 second per node, and the standard deviation is 0.0005 second per node. Note that distribution of the network random delays can be changed only in the end-to-end and the tandem nodes scenarios. Under Ethernet, it is determined by the medium random access time. All other parameters have the same values as those in the case of exponential distribution and they are either fixed throughout the simulations of the thesis or set to the default values, which are described in the Section 4.4 and 5.4.

6.4.1. The End-to-end Model

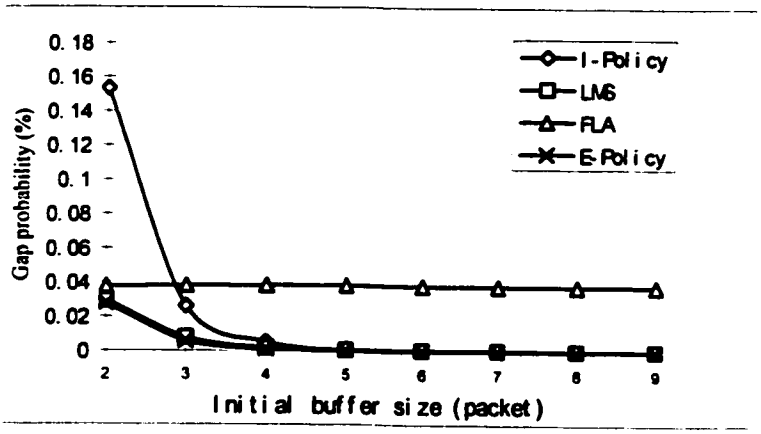

Figure 6.10. Gap probability vs. initial buffer size with Normally distributed network random delays in end-to-end scenario.

Figure 6.10 depicts that when the network random delay is Normally distributed in end-to-end scenario, the gap probabilities of the four algorithms behave similarly with the ones as the network random delays are exponentially distributed. The gap probabilities of

the I-Policy, the E-Policy and the LMS decrease very fast as the initial buffer size increases, and they tend to zero when the initial buffer size is larger than 5 packets. The E-Policy has the lowest gap probability and the LMS has very close values to the E-Policy. Although the FLA has larger gap probability than other algorithm, we think the values (around 3.8%) are quite acceptable low and very close to the target gap probability (4%).

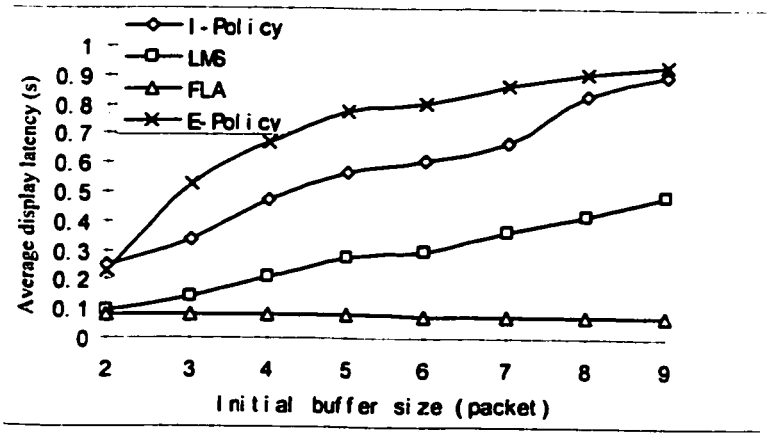

Figure 6.11. Average latency vs. initial buffer size with Normally distributed network random delays in the end-to-end scenario.

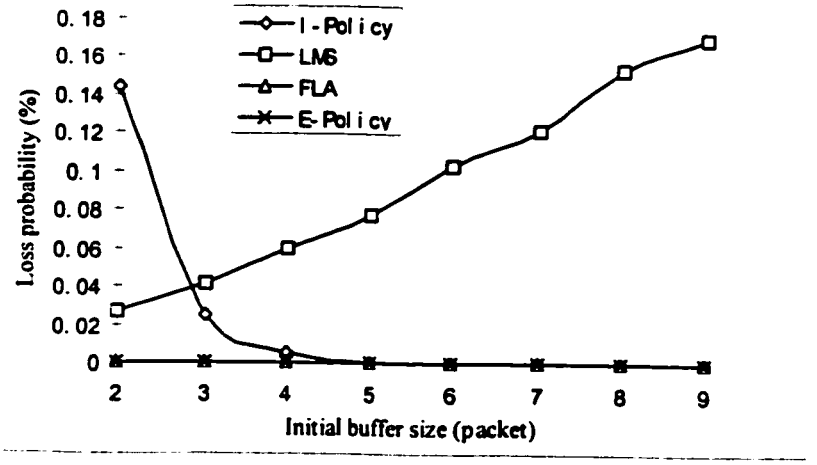

Figure 6.12. Loss probability vs. initial buffer size with Normally distributed network random delays in the end-to-end scenario.

Figure 6.11 depicts that in the end-to-end scenario, the average display latencies of the I-Policy, the E-Policy and the LMS increase as the initial buffer size increases. The FLA has the lowest almost constant average display latency and the E-Policy has the largest one. LMS and I-Policy have values in between. The FLA outperforms other algorithms.

Figure 6.12 depicts that in the end-to-end scenario, the loss probability of the LMS increases as the initial buffer size increases. On the other hand, the loss probability of the I-Policy decreases quickly as the initial buffer size increases. The E-Policy and the FLA have zero values because of their definitions. When the initial buffer size is larger than 5 packets, there is hardly any difference among the I-Policy, the LMS and the FLA since they are all zeros.

6.4.2. The Tandem Nodes Model

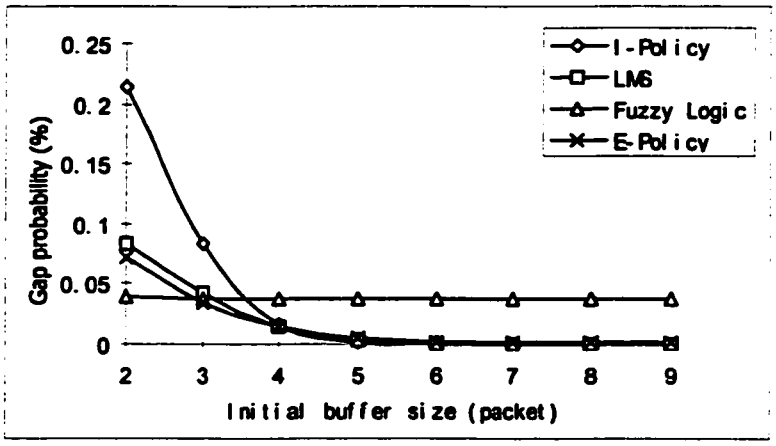

Figure 6.13. Gap probability vs. initial buffer size with Normally distributed network random delays in the tandem nodes scenario.

Figure 6.13 depicts that in the tandem nodes scenario, the gap probabilities of the I-

Policy, E-Policy and LMS also decreases very fast as the initial buffer size increases and they all tend to zero when the initial buffer size is larger than 6 packets. The FLA has a constant gap probability within the desired range. The I-Policy has the largest gap probability, and the E-Policy has the lowest one, while the LMS have values in between.

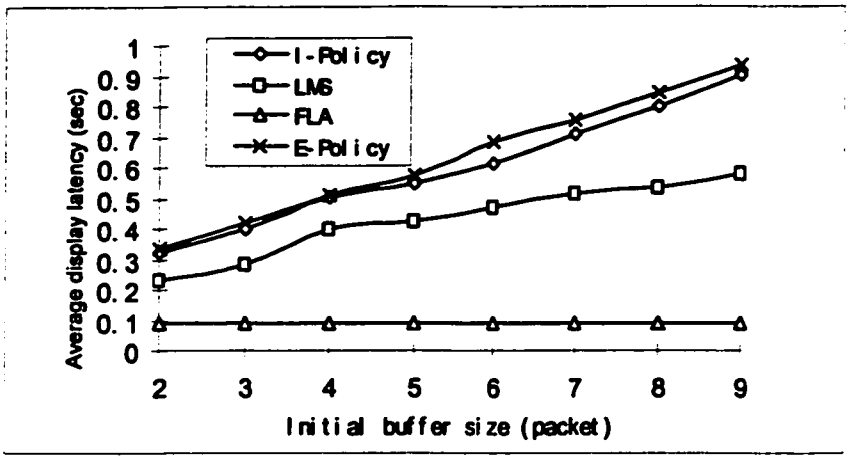

Figure 6.14. Average latency vs. initial buffer size with Normally distributed network random delays in the tandem nodes scenario.

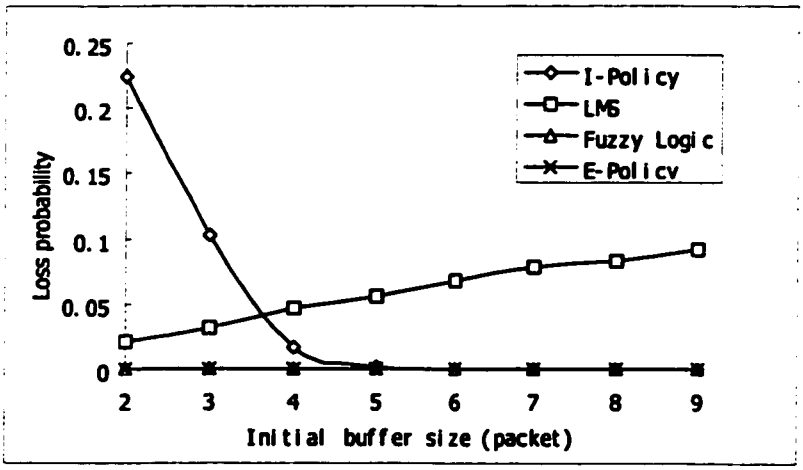

Figure 6.15. Loss probability vs. initial buffer size with Normally distributed network random delays in the tandem nodes scenario.

Figure 6.14 depicts that when the network random delay is Normally distributed in tandem nodes scenario, the average display latencies of the I-Policy, E-Policy and

LMS increase as the initial buffer size increases. The FLA has the lowest almost constant average latency and outperforms others. The E-Policy has the largest one. The LMS and the I-Policy have values in between.

Figure 6.15 depicts that when the network random delay is Normally distributed in tandem nodes scenario, the loss probability of the I-Policy decreases quickly as the initial buffer size increases. On the other hand, the loss probability of LMS increases as the initial buffer size increases. The FLA and the E-Policy have zero values because of their definitions. When the initial buffer size is larger than 5 packets, the loss probability of the I-Policy tends to zero.

From the above simulation results (from Figure 6.10, to 6.15), we conclude that when the network random delay is Normally distributed, the two new algorithms also outperform the other two benchmark-algorithms, and the FLA is better than LMS mainly because of the zero loss probability.

6.5. Tradeoff Discussion

Obviously, the tradeoff of jitter buffer management is the gap probability and the display latency. As we assume that we have no control on the source and the network, so the only way we can do for the late packet is to wait. The problem is that waiting for the individual late packet will also delay other packet for displaying. Again, it is important to schedule the display timing of a real time application. Any interruption or violation of this pre-defined timing will degrade the voice quality and lead to poor audience perception. Therefore, how to manage the buffer, i.e. how to remove more gaps and in the mean time to keep the latency as low as possible is the key point of our task. The LMS

and the FLA use different approaches to solve the problem.

Although LMS has mechanism to drop the latency if the threshold is reached, the tradeoff is to discard the received packet(s). As the loss probability is getting larger, this will degree the voice quality and may lead to understanding problem for the receiver.

As revealed by simulations, and comparisons with the other three algorithms, the FLA has the capability to maintain a desirable gap probability, display latency and zero loss probability. The tradeoff is to alter the length of the silence period (in order to change the initial buffer time of each session). This may have an impact on the original voice rhythm if the frequency of the adjustment is too high. However, we believe that this is much better than deleting parts information of the voice. Although it needs no complicated calculation and proved to be easy to be implemented, it requires the expert knowledge to decide the control parameters, membership functions and inference rule matrix.

6.6. Design Guideline

Generally, we prefer the FLA because of several reasons. First, the FLA always has a low and stable gap probability and a low average display latency. In particular, the gap probability is well controlled around the reference gap probability whether the initial buffering is insufficient or excessive. This is a desirable feature when implemented. Second, unlike the LMS, which reduces the queue size by discarding packets, the FLA does not discard any packet, so it can recover the full original information. Third, the "IF... AND" rules in the FLA are much easier to be changed or modified if the FLA is implemented in a new environment.

Based on our simulation experience of the two new algorithms, we present some guidelines on how to choose the right parameters in order to get the desired results and to implement them.

6.6.1 General Design Guideline

1. Time slot: The time slot varies for different codec (e.g. 15ms, or 20ms). The time slot can be detected by finding the difference between the creation time of two continuous packets. In real application, for example, each packet header may have a timestamp, so the time slot value can be equal to the difference between two timestamp values of two continuous packets.
2. Initial buffer size: Although the two new algorithms have features to adjust the initial buffering. It is recommended that the initial buffer size is set within a moderate range. Because if it is too small, the receiver will experience large gaps before makes adequate adjustment. On the other hand, if the initial buffer size is set too large, the receiver will have large display latency before makes adjustment. From the large number of simulations we conducted, we suggest that a good range of initial buffer size is from 3 packets to 5 packets (using a time slot of 20ms). So both the gap probability and latency are within acceptable level.

6.6.2. Design Guideline Specific to LMS Algorithm

1. The buffer size of the LMS must be set greater than the threshold value in real application. There is no need to set to it very large. Since when the threshold is reached, the node may start to discard packets and the queue size will not exceed the threshold too much. It is possible that the node may not discard packet

because the prediction result permits it to receive more packets. So in this case there must be some extra space in the buffer reserved for the coming packet(s). From experience the buffer capacity can be set 50% more than the threshold value.

2. The queue threshold: The queue threshold value is mainly determined by the gap probability, loss probability and the display latency. From a large number of simulations, we suggest that a good range is from 3 packets to 6 packets. If it is too small, the frequency that it is reached is high and more packets will be discarded. Therefore, the voice quality will be degraded. On the other hand, if it is set too large, the average display latency is large and this may prevent the voice communication being interactive.

6.6.3. Design Guideline Specific to FLA

1. The packet number of one session N is designed to prevent the node from adjusting the initial buffer time excessively. By experience, it can be set from 1/30 to 1/50 of the total pack numbers.
2. Other parameters like M and K are the factors related to the control "strength". The good range for M is less than 0.05 and the one for K is less then 0.04. All the current values are tested to be ideal by a great amount of simulations under the scenarios described in this thesis.
3. It is possible to set the target gap probability lower than 4%, e.g. 2%, however, as a tradeoff, the average display latency is larger, we therefore set it to be 4%, which is quite acceptable while keeping the latency low.

Chapter 7

Conclusion

In this thesis, we have reviewed jitter management algorithms based on network's source end, intermediate nodes and destination end. We advocate the use of the destination-based approach. We think it is the most practical way among the many scenarios in heterogeneous network environments.

We have incorporated LMS method on the jitter buffer management. Simulation results showed that the LMS algorithm is an effective prediction-based approach. It works well to keep the display latency under threshold while maintaining the gap probability and loss probability under acceptable level. The only weak point of this algorithm is that it may discard too many packets when the initial buffer size is large. Therefore it may corrupt the intelligibility of the voice communication.

We have also applied the Fuzzy Logic method on the jitter buffer management at the destination end. The simulation results showed that the FLA is superior to other evaluated algorithms and can maintain a stable constant gap probability, an acceptable average display latency, and zero loss probability. It can recover all the original information and it needs no complicated calculation. It has proved itself to be easily implemented. However, it requires the expert knowledge to decide the control parameters, membership functions and inference rule matrix. Overall, we prefer the FLA as the first choice.

We have made comparison among the several algorithms in terms of performance

in the gap probability, display latency, and the loss probability. We have run extensive simulations to test their performance on different network environments, to be sure that the results are universal and applicable in the real application. From these results, we have also attempted to make some design recommendation based on our experience.

7.1. Future Work

Much work can be done in the future. More simulations could be conducted to reveal the relations between some other parameters and the performance. In the FLA, the parameters can be more self-tuning so the system will have more flexibility to adaptively adjust to the changing network. For the prediction algorithm, other predicting method can be used in order to reach the high accuracy and fast convergent speed. It would also be useful to establish some theoretical (or just empirical) relationship between gap probability and initial buffer size (or between latency and initial buffer size). We can also investigate the performance evaluation on different codec of our two algorithms or using real cross traffic in the tandem nodes model. Finally, we can determine a good jitter definition that would simplify the implementation of jitter buffer algorithm.

References

[AbSo94] Shunji Abe, "A traffic control method for service quality assurance in a ATM network", IEEE Journal on Selected Areas in Communications, Vol. 12, No. 2, February 1994. Pages 322-331.

[Adas98] A. Adas, Using Adaptive Linear Prediction to Support Real-Time VBR Video Under RCBR Network Service Model, IEEE/ACM Transaction on Networking, Vol.6, No. 5, Oct. 1998, Pages. 635-644.

[Bolo93] J.C. Bolot, "Characterizing end-to-end packet delay and loss in the Internet", High Speed Networks, Vol. 2, No. 3, December, 1993, Pages 305-323.

[BuVl01] Maarten Büchli, Danny De Vleeschauwer, Jan Janssen, Annelies Van Moffaert and Guido H. Petit, "Resource allocation and management in DiffServ networks for IP telephony", 11th International workshop on Network and Operating Systems support for digital audio and video, New York, June, 2001, Pages 33 – 39.

[CaCr96] R. L. Carter & M. E. Crovella, "Measuring bottleneck link speed in packet-switch networks", March, 1996.

[CaFi96] V. Catania, G. Ficili, S. Palazzo, D. Panno, "Using Fuzzy logic in ATM source traffic control: lessons and perspectives", IEEE Communications Magazine, November, 1996, Pages 70-74.

[ChSa98] G. Chiruvolu, R. Sankar, and N. Ranganathan, "Adaptive VBR Video Traffic Management for Higher Utilization of ATM Networks", Vol. 28, No. 3, July 1998, SIGCOMM Computer Communication Review, Pages 27-40.

[ClTa99] Mark Claypool and Jonathan Tanner , "The effects of jitter on the perceptual quality of video"; Proceedings of the seventh ACM international conference (part 2) on Multimedia (Part 2), 1999, Pages 115 – 118.

[DeCh00] C. Demichelis, P. Chimento, Instantaneous Packet Delay Variation Metric for IPPM; IPPM draft, 2000.

[ElEl96] T. A. ElBatt, S. El_Henaoui, S. Shaheen, "Jitter Recovery Strategies for Multimedia Traffic in ATM Networks", GLOBECOM96', Pages 1202-1206.

[FuLi98] C. Fulton & S. Q. Li, " Delay jitter first-order and second-order statistical functions of general traffic on high-speed multimedia networks", IEEE/ACM Trans. Networking, Vol. 6, No. 2, April, 1998, Pages 150-163.

[Hayk01] Haykin, Simon S., "Adaptive filter theory", 4th ed., Prentice – Hall, 2001.

[HuPe00] Rose Qingyang Hu and David W. Petr; "A predictive self-tuning fuzzy-logic feedback rate controller", IEEE/ACM Trans. Networking 8, 6 (Dec. 2000), Pages 697 – 709.

[ITU93] Telecommunication Standardization Sector of ITU. ITU-T Recommendation G.114. Technical report, International Telecommunication Union, March 1993.

[JiSc00] Wenyu Jiang and Henning Schulzrinne "Analysis of On-Off Patterns in VoIP and Their Effect on Voice Traffic Aggregation", ICCCN, Las Vegas, Nevada, Oct. 2000.

[KaTo01] M.J. Karam and F.A. Tobagi, "Analysis of the Delay and Jitter of Voice Traffic Over the Internet", Proceedings of INFOCOM 2001, Volume 2, pp. 824-833, Anchorage (AL), USA, April 2001.

[LaYa00] M. Lazar and O. Yang, "Empirical Study of End-to-End Jitter in Data Networks", Proc. Telecommunication Symposium, Washington D.C., April 2000. Pages 121-126.

[LiTs95] N. Likhanov and B. Tsybakov, "Analysis of an ATM buffer with self-similar ("Fractal") input traffic", Proc. IEEE INFOCOM'95, Boston, April, 1995, Pages 985-992.

[MaPa01] "Jitter Control in QoS Networks", Yishay Mansour and Boaz Patt-Shamir, IEEE/ACM Transactions on Networking, Vol. 9, No. 4, August 2001, Pages 492 – 502.

[Matl01] http://www.mathworks.com

[NaKl82] W. E.Naylor & L. Kleinrock, "Stream traffic communication in packet switched networks: destination buffering considerations", IEEE Transactions On Communications, Vol. COM-30, No. 12, December, 1982.

[Opne01] http://www.opnet.com/products/modeler/home.html

[PaFl95] Vern Paxson & Sally Floyd, "Wide Area Traffic: The Failure of Poisson Modeling", IEEE/ACM TRANSACTIONS ON NETWORKING, Vol. 3, No. 3, June 1995, Pages 226-244.

[PiLa97] A. Pitsillides & J. Lambert, "Adaptive congestion control in ATM based networks: quality of service and high utilisation", Computer Communications, Vol. 20, No. 14, Dec.1997, Pages 1239-1258.

[Qiu97] B. Qiu, "A predictive fuzzy logic congestion avoidance scheme", Globecom'97, Vol. 2, Nov.1997, Pages 967-971.

[ScCa96] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", rfc1889, January 1996.

[ShYa99] T.Shan and O. Yang, "An Effect Admission Control Scheme for Real-time VBR Traffic in the ATM Network: Deterministic Bandwidth Allocation", Computer Communications. Vol. 22, No. 10, June 1999, Pages 966 – 979.

[StJe95] D. L. Stone & K. Jeffay, " An empirical study of delay jitter management policies", Multimedia Systems, Vol. 2, No. 6, January 1995, Pages 267-279.