Active Queue Management for Real-time IP Traffic

Xiaoyan Wang

SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Department of Electronic Engineering

Queen Mary

University of London

October 2006

To my parents and my husband, Daxu

Acknowledgements

I am truly grateful to everyone who has directly or indirectly supported and helped me complete this Ph.D thesis. First and foremost, I would like to thank my supervisor Prof. Jonathan Pitts who has given me all the support, assistance and encouragement that I needed throughout my Ph.D study. He offered so much insightful advice on my research and reviewed my draft of thesis with amazing turn-around times. This thesis would not have been possible without his support and guidance. You are the one who makes me believe that I can do this, thank you so much.

I also would like to thank everybody in the department of Electronic Engineering at Queen Mary, University of London for creating a nice and friendly working environment, especially thanks to Dr. Yue Chen, Qiang Yang, Jimmy Leung, Shaowen Lu, Bo Yu, Yong Zuo and Vindya for their help and friendship. Many thanks to Rupert and Touseef, for installing and maintaining the life of NS2 clusters.

I can not express enough gratitude to my parents, their constant love, encouragement and support have always been the source of my strength and the reason I have come this far.

The last but not least, my special thank you to my husband, Daxu, who helps me through the difficult times and gives me his unconditional love and support all the time.

Abstract

Real-time multimedia services have become increasingly popular over the IP network, in particular voice over IP (VoIP) and real-time video. Nevertheless, due to the IP network's best-effort nature, real-time traffic may experience less than desirable quality, and the performance is unsatisfactory particularly when the link is congested. Much effort has been directed to provide Quality of Service (QoS) for real-time traffic over IP networks. The two main QoS architectures are Integrated Services (IntServ) and Differentiated Services (DiffServ); these comprise many components such as Active Queue Management (AQM), scheduling mechanisms, resource reservation management and QoS signaling.

Of these mechanisms, Random Early Detection (RED) is useful in controlling the average queue size (AQS), and hence the queuing delay, with or without a cooperating transport protocol. This control of queuing delay is extremely helpful for delay sensitive real-time traffic. However RED has only been widely studied in the context of TCP congestion collapse; until now, the application of RED to UDP traffic has not been investigated.

This thesis presents a method of improving the QoS of real-time audio/video over IP network infrastructure at low service cost and with no changes to the existing network. Firstly, a thorough study of RED's effects on voice over IP is given, concluding that RED is able to control the delay distribution and jitter of VoIP under unloaded or loaded conditions.

Secondly, the effective loss experienced by VoIP is investigated: this comprises both those packets dropped within the network plus those that are delivered but arrive too late to be of use. By controlling delay, RED is able to decrease the proportion of packets that arrive too late; thus the effective loss is decreased. In addition, the random drop of RED is also able to decrease consecutive losses which are known to be harmful to voice and video streams.

Thirdly, RED configuration guidelines, which depend on the targeted performance requirements and the applied network load, are given. RED bounds the queuing

behaviour by limiting the actual AQS; hence the queuing delay that real-time traffic will experience within a router is known, and this can help network service providers predict whether the required QoS can be met. Based on this bound, service providers can configure paths with different delay and loss characteristics in order to support real-time services with different QoS requirements.

Fourthly, the application of RED is further extended by including a more realistic multi-hop topology, video traffic and DiffServ architecture, and it turns out that RED's performance control also works under these situations.

In conclusion, the use of RED in managing UDP traffic means that delay, jitter and loss can be controlled, thus providing a good solution to quality degradation of real-time traffic under congested network conditions.

Table of Contents

List of Figures

List of Tables

Glossary

AQS Average queue size

AQM Active Queue Management

CDF Cumulative Delay Function

CCDF Complementary Cumulative Delay Function

CBR Constant Bit Rate

DR Decay Rate

ER Excess-Rate

FEC Forward Error Correction

FIFO First In First Out

GAPP Geometrically Approximated Poisson Process

ITU International Telecommunication Union

IPP Interrupted Poisson Process

LBR Low Bit-rate Redundancy

LDT Large Deviation Theory

LRD Long Range Dependent

MOS Mean Opinion Score

MMPP Markov Modulated Poisson Process

MPEG Motion Picture Experts Group

PCM Pulse Code Modulation

PDF Probability Density Function

RED Random Early Detection

RTP Real-time Protocol

RTCP RTP Control Protocol

QoS Quality of Service

SIP Session Initiation Protocol

VBR Variable Bit Rate

VoIP Voice over IP

WDM Wavelength Division Multiplexing

WFQ Weighted Fair Queuing

List of Mathematical Symbols

C	the service rate of the buffer
$Q(.)$	queue length distribution
$P(.)$	probability function
c_p	packet-scale decay constant
c_b	burst-scale decay constant
η_p	packet-scale decay rate
η_b	burst-scale decay rate
N	the number of source
R	maximum transmission rate of a on/off source
R_{on}	mean rate in the ON state (aggregate model)
α	on/fff source's activity factor
ρ	buffer utilization
$T_{(on)}$	mean ON time (aggregate model)
$T_{(off)}$	mean OFF time (aggregate model)
F	the rate of flow arrivals
A_p	mean load in packets/s
R_{off}	mean rate in the OFF state (aggregate model)
A	the offered traffic in Erlang
x	knee point between packet and burst-scale queuing region
p_{er}	probability of excess rate
α	transitional rates from the On to Off state
β	transitional rates from the Off to On state
N_o	minimum number of active sources for burst-scale queuing
s	queue state

λ_ι	arrival rate
dr	decay rate
q	average queue size
F(x)	cumulative distribution function
$F_c(x)$	complementary cumulative distribution function

Chapter 1 Introduction

1.1 Motivation behind This Research

In the last few years there has been an explosive growth in multimedia applications over IP networks, e.g. IP telephony, teleconferencing, streaming video/audio, interactive games, distance learning. These new services have significantly different requirements from those traditional data-oriented applications such as Web text/image, e-mail, FTP. A key difference is that the multimedia applications are sensitive to end-to-end delay and delay variation, but can tolerate occasional loss.

It is well known that IP networks today only provide a best-effort service. The network makes its best effort to forward each packet, but it does not make any promise whatsoever regarding delay, jitter or loss for an individual packet. Due to the lack of any timely effort to deliver packets, it is a challenge to develop successful multimedia networking applications over IP networks. So to date, multimedia over the IP network has achieved limited success.

IP networks can carry a large variety of multimedia applications, and normally there are three broad classes of multimedia applications: streaming stored audio/video, streaming live audio/video and real-time interactive audio/video. Of these, real-time interactive audio/video applications, such as Internet phone and video conferencing, have been less successful than streaming live or stored audio/video, because real-time interactive applications have more rigid constraints on performance. In addition, the quality of real-time traffic will deteriorate to unacceptable levels when real-time traffic is delivered over a moderately congested link, and this feature brings a conflict with the desired situation to fully utilize the bandwidth. Although these problems exist for real-time traffic over IP networks, the low cost of services and potential new applications for real-time traffic over IP networks still attract the attentions of network service providers. Thus the provision of sufficient quality for real-time traffic over IP networks and making the most of the bandwidth is the motivation behind the research work described in this thesis. In the following part,

brief background descriptions are given about real-time traffic over IP networks.

1.1.1 Structure of Real-time Traffic over IP networks

Figure 1.1 briefly introduces the three main parts of voice/video service over an IP network. Briefly, it is a process of transforming analog signals to digital signals, transmitting digital audio/video signals across an IP network in the form of packets, and the reassembly and playback of these signals at the receiver. Firstly, at the sender, the input audio/video signals are digitized and compressed by the encoder in units as frames, then frames are packetized to form the payload of a packet and the headers (e.g. IP/UDP/RTP headers) are added to the payload to form a packet. Next the packets are transmitted over the network at regular intervals (the packet interval or packetization delay) and may suffer different network impairments (e.g. packet loss, delay and jitter) within IP networks during transmission. At the receiver, the playout buffer compensates for network jitter at the cost of further delay (buffer delay), then the packet headers are stripped off and voice/video frames are extracted from the payload by the depacketizer. The de-jittered voice/video frames are decoded to recover the analog audio/video.

As mentioned above, different network impairments will occur due to the behaviour of the best-effort IP network. To maintain the quality of received voice/video, much research has investigated how to provide Quality of Service (QoS) mechanisms suitable for these voice/video services in a packet-switched network. Next we will discuss these efforts based on the structure mentioned above.

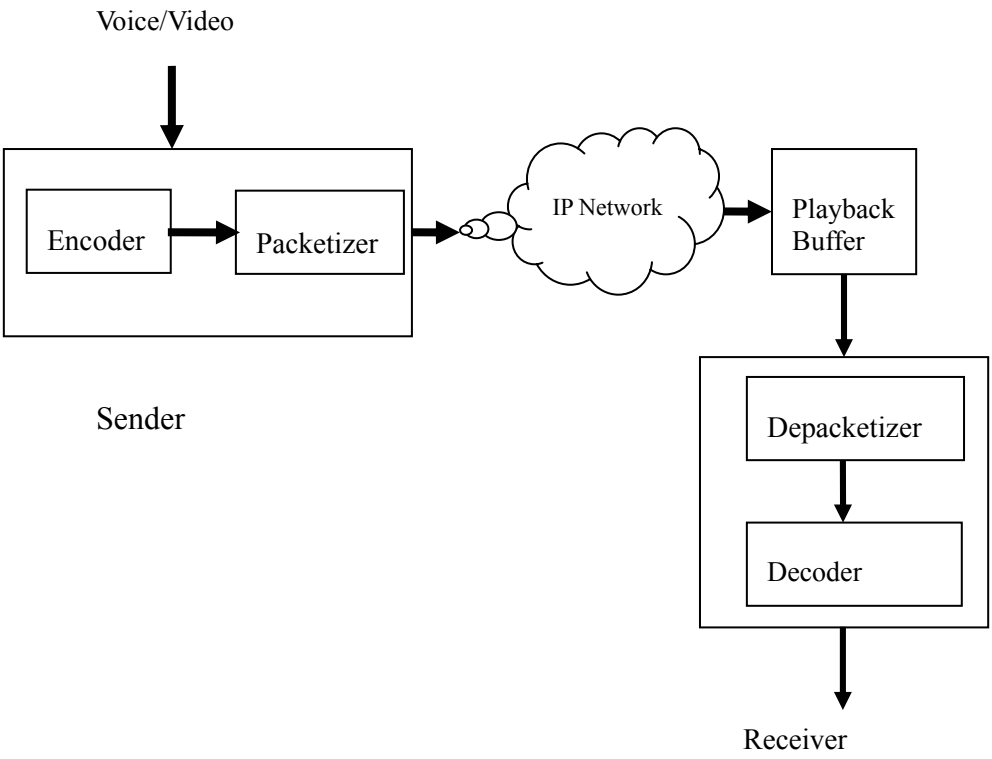

Figure 1-1 Structure of Voice/Video over IP Networks

1.1.2 QoS Efforts for Real-time Traffic over IP network

Firstly, at the sending end, voice/video signals are compressed before transmission in order to reduce the bandwidth consumed. For telephony applications, the basic encoding technique is Pulse Code Modulation (PCM), with the corresponding codec G.711 running at 64Kbps, but many low bit-rate codecs for compression are provided to save bandwidth, such as G.729 (8Kbps) and G.723.1 (6.3 or 5.3Kbps), which have quality almost identical to that of G.711. For video applications, this redundancy compression is even more important because video signals need much more bandwidth than voice. Many compression standards are provided, for example, MPEG 1 for CD-ROM quality video (1.5Mbps), MPEG 4 (typically between 5Kbps and more than 1Gbps) and H.261 (n*64Kbps).

To make voice/video traffic suitable to be transmitted over a packet-switched network and recognizable by routers, a standard packet structure that includes fields for audio/video data, sequence number, timestamp and other potentially useful fields is created, such as RTP (Real-time Protocol) [RFC3550], RTCP (RTP Control

Protocol) [RFC1889], SIP (Session Initiation Protocol) [RFC3261] and H.323. Real-time voice/video signals are encapsulated by these protocols at the sending end, so they can be transmitted as packets and recognized by routers and different receiving end systems. These protocols make the widespread implementation of real-time traffic feasible and easier.

Due to the strict delay requirements, real-time voice/video over IP networks usually means that there is no possibility of retransmissions. When a voice/video packet is lost over a best-effort network, the quality will degrade due to loss of information. To reduce the impact of lost packets, a scheme called Forward Error Correction (FEC) [RFC2733] is provided at the sending end, which adds data redundancy to the original packet streams and the redundant information can be used to reconstruct approximations or exact copies of the lost packets, for example, Parity coding, Reed-Solomon coding, and Hamming codes.

However, this reconstruction of FEC has its limitations. As introduced in [Riz97], a (n, k) Reed-Solomon FEC code adds $n-k$ redundancy packets to every k original packets, forming a block of n packets. A Reed-Solomon code can recover all losses in the same block only if at least k out of every n packets are received. If more than $n-k$ packets in a block are lost, then no lost packet in that block is recoverable. So FEC suffers from an "all or nothing" property, and can not work well during congestion periods. In addition, the cost of FEC is an increased transmission rate for the audio/video stream, thus burdening the network with extra load especially when bandwidth is most valuable.

Another variation is called Low Bit-rate Redundancy (LBR) [RFC2198], and it works by sending at least two streams for the same voice/video, one using the original codec, the other one using a lower bit-rate and thus lower quality codec. Obviously similar problems of bandwidth consumption exist.

Secondly, at the receiving end, apart from the schemes that cooperate with the sending end listed above (e.g. FEC, LBR), a number of techniques for Error Concealment initiated by the receiver of an audio stream have been designed which do not require assistance from the sender. These techniques can work when sender-based recovery schemes fail to recover all the loss. As discussed in [Per98], in

these Error Concealment schemes, a replacement for a lost packet which is similar to the original one is produced. It is possible because audio signals, in particular speech, exhibit correlations over short timescales. So these techniques work for relatively small loss rates (less than 15 percent) and for small packets (4–40ms). Packet repetition and interpolation are examples of receiver-based recovery, and the difference between them is the computational complexity. These error concealment schemes can work in tandem with sender-based repair but then the problem of increased bandwidth consumption still exists.

In addition, even if a real-time voice/video packet arrives at its destination, it does not mean this packet is transmitted successfully. Connectionless IP networks can result in packets arriving at their destination out of their original sequence order or with such delay variation (jitter) that a packet is effectively lost because it missed its playback deadline. A late loss has the same effect as a network loss in terms of quality degradation, correspondingly the mechanism known as playout adaptation or dejittering is used at the receiving end to compensate for jitter and reduce the number of late losses. The simplest playout algorithm uses a fixed playout delay. More advanced algorithms [Ram94] adapt the playout delay according to network conditions. But these dejitter and playout algorithms will add some extra delays for the arriving packets.

Generally these schemes used at the sender end and receiver end are passive, in other words, these schemes try to recover information when packets are lost but can not decrease the number of lost packets. Moreover, these schemes have the problem of consuming extra bandwidth with the insertion of redundant information at the sender. In contrast with these passive schemes, those efforts provided within the network to avoid losing packets are more active, so next we will introduce the proposals provided within the network for QoS.

In general, the QoS within a network aims to provide resource assurance and service differentiation in a network. Based on this aim, some researchers think it is not necessary to make fundamental changes to existing best-effort services and the underlying Internet protocols. Their approach is to add more bandwidth and switching capacity to provide satisfactory QoS over IP networks, (e.g. using

Wavelength Division Multiplexing (WDM) to make transmission bandwidth abundant and inexpensive). Opponents of this point argue that additional bandwidth can be very costly (it depends who owns it), and takes time to provision, and it will be consumed by new inelastic applications.

Another approach is to provide new service models and mechanisms to provide QoS for real-time traffic. There have been several service models proposed by the Internet Engineering Task Force (IETF), industry and the academic research community. Among these models, the most popular and widely-accepted are Integrated Services (IntServ) [RFC1633] and Differentiated Services (DiffServ) [RFC2474] [RFC 2475].

Integrated Service (IntServ) is a framework to provide individualized QoS guarantees to individual applications, but it needs some fundamental changes to the IP network. IntServ mechanisms need to reserve bandwidth, which means for real-time traffic transferring from source A to destination B, the application should be able to reserve bandwidth along the route so that enough bandwidth can be provided for it. But this reservation mechanism implies the need for some major changes to today's IP network, such as the need for a protocol obeyed by the applications to reserve link bandwidth, modifications to scheduling algorithms within router queues in order to honour the reservations, and the network itself must have a method to detect whether sufficient bandwidth is available to accept a new reservation request.

Differentiated Services (DiffServ) architecture provides the ability to handle different classes of traffic in different ways within the IP network. It makes relatively small changes at the network and transport layers, and introduces some simple policing schemes at the edge of network, i.e. at edge routers. DiffServ aggregates individual flows into different traffic classes at the edge routers, and then the core routers within the network forward each packet to its next hop according to the per-hop behaviour associated with the traffic class of the packet. However, due to flow aggregation and the lack of admission control, DiffServ does not provide end-to-end QoS guarantees to individual flows.

But as pointed out in [Kur05], until now neither IntServ nor DiffServ have taken off and found widespread adoption in reality due to economic and legacy reasons rather than technical reasons.

Besides these QoS networking architectures, various architecture components (mechanisms) are proposed to support and implement these network service models and provide service differentiation in packet networks, such as packet scheduling, Active Queue Management (AQM), congestion control, and admission control.

Among them, congestion control mechanisms are used to limit, shape or divert traffic so that the probability of congestion is reduced and thus enhancing QoS assurance. For example, some video sources are designed with a rate control mechanism that has the ability to reduce the transmission rate for the purpose of congestion control. Admission control protects the service quality for existing traffic in the network by determining whether or not to accept a new service request based on current resource utilization. A secondary goal of the admission control strategy is to achieve as high a utilization of the network as possible.

Queue management and scheduling are two classes of router algorithms. When packets belonging to different flows are multiplexed and queued for transmission at the buffers, Queue management algorithms manage the length of packet queues by dropping packets when necessary or appropriate, while scheduling algorithms determine which packet to send next and are used primarily to manage the allocation of bandwidth among flows. During periods of network congestion when buffer resources become scarce, the queue management strategy becomes critical to overall performance and QoS. So as specified in [RFC2309], Active Queue Management (AQM) has been proposed to replace the traditional drop-tail queue management in order to improve network performance by keeping the average queue size small, and reducing the number of lost packets when traffic bursts arrive etc. This is also the research focus of this thesis, and we will introduce it in detail later.

Until now a whole picture of QoS service for real-time traffic over IP networks has been given. However these proposed efforts either have their limitations such as consuming extra bandwidth or being passive, or have not taken off due to various reasons. None of them is strong enough and widely-accepted to provide a satisfying service over IP networks for real-time traffic. So more effort is needed within this area.

1.2 Objective of This Thesis

Since the quality of real-time audio/video over IP is not guaranteed and can deteriorate to unacceptable levels when traffic is delivered over a moderately congested link, what can be done to improve the quality? We know that AQM algorithms such as Random Early Detection (RED) [Flo93], can help control the average queue size (and thus queuing delay) within a network whether the transport protocol is cooperative or uncooperative. In fact this control of queuing delay is extremely helpful for delay sensitive real-time traffic. However, RED was widely studied to avoid TCP congestion collapse, and the application of RED to UDP traffic has not been studied until now. This becomes our starting point, and we make a thorough study about how to apply the RED algorithm to improve the QoS of real-time traffic over an IP network in terms of delay, jitter and loss. In this thesis, we investigate the use of RED to improve QoS for real-time voice and video traffic, especially when the load is high (which is when the QoS of real-time traffic normally deteriorates greatly). Our scheme is based on appropriately configuring the RED queue management within routers to provide predictable service, and it does not need changes to the fundamental algorithms and architectures, so it is easy to be widely used over existing networks at low cost.

1.3 Novelty and Contribution of This Research

The major contribution of this thesis will focus on the area of performance improvements of real-time traffic over IP networks, and it presents a method of improving the QoS of real-time audio/video over IP network infrastructure at low service cost and with no changes to the existing network.

Firstly, through a thorough study of RED's effects on VoIP, we conclude that RED is able to control the delay distribution and jitter of VoIP under both unloaded and loaded conditions.

Secondly, the effective loss experienced by VoIP is investigated: this comprises both dropped packets plus those that arrive too late to be of use. By controlling delay, RED is able to decrease the proportion of packets that arrive too late; thus the effective

loss is decreased. In addition, the random drop of RED is also able to decrease consecutive losses which are known to be harmful to voice and video streams.

Thirdly, RED configuration guidelines, which depend on the targeted performance requirements and the applied network load, are given. RED bounds the queuing behaviour by limiting the actual AQS; hence the queuing delay that real-time traffic will experience within a router is known, and this can help network service providers predict whether the required QoS can be met. Based on this bound, service providers can configure paths with different delay and loss characteristics in order to support real-time services with different QoS requirements.

In conclusion, the use of RED in managing UDP traffic means that delay, jitter and loss can be controlled, thus providing a good solution to quality degradation of real-time traffic under congested network conditions.

1.4 Layout of Thesis

The rest of this thesis is organized as follows:

In Chapter 2, a review of real-time traffic over IP networks is provided. This chapter also introduces the QoS performance metrics for real-time traffic and their corresponding requirements. Moreover, background information about Active Queue Management and its typical example Random Early Detection (RED) are also discussed.

Chapter 3 introduces the widely used traffic models for voice and video traffic, and their corresponding queuing behaviors are described as the basis for the theoretical foundation of simulations. Moreover the comparison between the theoretical results and the simulation results is provided as a verification for the accuracy of the simulations.

Chapter 4 concentrates on the analysis of RED's QoS improvements to VoIP traffic. Firstly the analysis about RED's ability to filter the decay rate is provided, followed by the effects on the performance metrics of VoIP traffic in terms of delay, jitter and loss. In addition, the performance control and management of VoIP by RED under loaded conditions is introduced.

Chapter 5 analyzes the burstiness of voice traffic. Based on the analysis for the most bursty voice traffic, RED configuration guidelines are derived for VoIP, which depend on the targeted performance requirements and the applied network load. Finally, the potential benefits of these configuration guidelines are illustrated.

In chapter 6, further applications of RED's performance maintenance are provided. Firstly, RED's end-to-end performance control of VoIP over a multiple hop topology is studied. Secondly, RED's performance control of video traffic under loaded conditions is demonstrated. Thirdly, the RED algorithm is applied to a more complex architecture: DiffServ, and the corresponding study about its management of delay, jitter and loss control is provided.

Chapter 7 summarizes the overall achievements in this thesis, drawing some conclusions and finally discussing the future work.

Chapter 2 Introduction to Real-time IP Traffic and Active Queue Management

As seen from the thesis title, there are two main parts in this thesis: Active Queue Management and Real-time IP traffic. This chapter firstly presents a brief overview of real-time IP traffic, such as the structure of VoIP packets and their codecs. Then the objective way to describe the QoS of real-time traffic in terms of delay, jitter, loss and throughput is described. Lastly, Active Queue Management is introduced, especially RED, the algorithm we have investigated to improve the performance of real-time IP traffic.

2.1 Real-time IP Traffic over IP Network

Real-time audio and video applications are two important multimedia applications on today's IP networks. These applications enable efficient communications through IP networks. For example, VoIP is similar to the traditional circuit-switched telephone service, but can provide local and long distance telephone service at very low cost. It can also facilitate the deployment of new services which are not easily supported by the traditional circuit-switched networks, such as videoconferencing, which enables a face to face meeting between groups of people at two or more different locations through both speech and sight. Next brief introductions about real-time voice and video over IP networks are described respectively.

2.1.1 Voice over IP

Voice over IP (VoIP) refers to the real-time delivery of packet voice across networks using the Internet protocols. The rapid growth of IP-based packet switched networks and the overall bandwidth efficiency of an integrated IP network make it an attractive candidate to transport voice connections.

Packet streams in voice over IP networks are often coded as constant bit rates. Packet

interval and packet size are constant, and the different codecs just produce different intervals and different packet sizes.

The purpose of a voice coder, also referred to as coder/decoder, or codec, is to transform and compress the analog signal of human speech into digital data. In addition, codecs can impact bandwidth. Firstly, this is because codecs determine the payload size of the packets transferred. By increasing payload size, the total numbers of packets sent are reduced, thus decreasing the bandwidth needed by reducing the number of headers required for the call. Secondly, the codecs define the size of the sample, and the number of samples per packet is another factor in determining the bandwidth of a voice call. Because the total number of samples placed in a packet affects how many packets are sent per second, so the number of samples included in a packet affects the overall bandwidth of a call.

G.711 is the traditional voice encoder, which uses Pulse Code Modulation (PCM) to generate 8 bits samples per 125 microseconds, leading to a rate of 64Kbps. More recent voice encoding schemes have been developed, leading to drastic rate reductions at the expense of additional encoding delay, such as 8Kbps for G.729, and 6.3Kbps for G.723.1.

The standard method of transporting voice samples through IP network requires the addition of three headers, these headers are for IP, UDP and RTP respectively. An IPv4 header is 20 octets (bytes), a UDP header is 8 octets (bytes) and a RTP header is 12 octets (bytes). So the total length of this header information is 40 bytes, or 320 bits. In addition, the link layer has different header sizes depending on the layer 2 protocol, such as Ethernet with 14 bytes header, PPP with 6 bytes header, and frame relay has 4 bytes header. These headers are sent each time a packet containing voice samples is transmitted. If we assume a voice packet is transmitted over Ethernet, the total extra header is 54 bytes.

Packet frequency is the number of packets containing voice samples which are sent per second. Packet frequency is the inverse of the duration in seconds represented by the voice samples. For example, if the voice samples in one packet represent a duration of 20 milliseconds, then 50 of these samples would be required each second and the packet frequency would be 50. Each packet carries a IP/UDP/RTP and link

layer header overhead of 54 bytes, and so the header information will add an extra bandwidth requirement based on the packets per second. For example, if a 64Kbps algorithm such as G.711 is used, and normally the payload is set at 160 bytes based on 160*8*50=64Kbps, then total bandwidth required to transmit the voice packet stream would be (40+14+160)*8 bits/20 ms = 85.6Kbps. So in our experiments later, we will assume that the bit rate of voice traffic with G.711 is 85.6Kbps when the source is active. Table 2-1 gives the details of the VoIP packets of different codecs we will use in this thesis, which are computed from the reports of [Cis01].

Voice codec	Bit rate (Kbps)	Payload (bytes)	Interval times (ms)	Packets per second	Packet size (bytes)	IP bandwidth (Kbps)
G.711	64	160	20	50	214	85.6
G.723.1	6.3/5.3	30	38.46	26	84	17.6
G.729	8	20	20	50	74	29.6

Table 2-1 Different voice codecs

Detailed consideration of each coding method is not introduced here, but the different coding methods vary in their levels of complexity, delay characteristics and quality. Because G.711 is the most common algorithm, and G. 723.1 and G. 729 are expected to become prevalent within VoIP, that is why we adopt these three algorithms for our simulations later.

2.1.2 Video over Internet

A video is a sequence of images, typically being displayed at a constant rate, e.g. 24 or 30 images per second. An uncompressed, digitally encoded image consists of an array of pixels, and each pixel is encoded into a number of bits to represent luminance and color. The transmission of uncompressed video data over a computer network requires very high bandwidth. To save bandwidth, the video data must be compressed before transmission. Two commonly used compression formats for real-time video are the Motion Picture Experts Group (MPEG) compression standards and H.261 video compression standards.

The MPEG standard is designed to take advantage of the relatively small difference between one frame and another in video data. It does an excellent job in the compression of motion pictures, but does not compress text well. MPEG compression standards include MPEG 1 for CD-ROM quality video (1.5Mbps), MPEG-2 for high quality DVD video (3-6 Mbps), and MPEG 4 for object-oriented video compression. H.261 is a compression algorithm like MPEG. The difference between them is that H.261 has been optimized for lower line speeds than MPEG. H.261 is usable even on a 64Kbps channel, where MPEG is defined to use 0.9 - 1.5Mbps. Details of these compression schemes can be found in [Fur01].

Video traffic has different characteristics from VoIP. For example, voice packets have short and constant sizes, while video packets have large and variable sizes. Video traffic does not exhibit on/off characteristics but behaves as a Variable Bit Rate (VBR) source, which always has bits to send but with rates that are dynamically changing.

2.2 QoS Performance Metrics for Real-time Traffic

For real-time traffic transmission, perceived speech and video quality is the most important QoS metric, as it is related directly to the quality experienced by the end users. The factors which influence perceived quality include delay, jitter and packet loss, which are the three main parameters that determine the QoS of perceived voice/video.

There are other factors affecting end-to-end perceived speech and video quality such as echo, noise, cross-talk, etc [Har01]. However these factors are not resulting from the packet-level performance of network, so they are not considered in the research of this thesis.

Besides the objective QoS metrics such as the loss rate, delay and jitter, there are some subjective QoS metrics that tell whether the quality is good or bad. Subjective QoS, most notably the Mean Opinion Score (MOS), is usually obtained through human evaluation tests, and is labor-intensive and far more time consuming than objective QoS measurements. So MOS is a specific area in the field of performance of real-time traffic, and normally objective QoS metrics are widely used to study the performance of these traffic. So in this thesis, we evaluate the performance based on

the objective QoS metrics.

2.2.1 Delay

Delay means the period of time a packet takes to traverse from one point in the network to another. Furthermore the time required for a packet to be passed from the source to its destination is called end-to-end delay. The components that contribute to end-to-end delay may be fixed or variable, and the following is a discussion of the various sources of delay for an IP packet.

A) Sender end

1. Coding Delay

Codecs are used to convert analog voice (or video) signals to digital data. They also perform voice compression to reduce the bandwidth required over IP networks. This conversion from analog to digital and then compression introduces delays in the codec at the sender. Different coding can result in different codec delay, and higher compression is achieved at the price of longer delays. Table 2-2 gives encoding and decoding delays for several typical voice codecs standardized by the International Telecommunication Union (ITU). As for video, the codec delay may be in the order of tens of milliseconds.

Coding standard	Compression algorithm	Bit rate (Kbps)	Encoding delay (ms)	Typical decoding delay (ms)	Total coding delay (ms)
G.711	PCM	64	0	0	0
G.729	CS-ACELP	8	15	7.5	22.5
G.723.1	ACELP	5.3/6.3	37.5	18.75	56.35

Table 2-2 Encoding and decoding delays for voice packet [Has00]

2. Packetization Delay

The encoded bit-stream generated is then packetized, and this incurs a packetization delay. This packetization delay equals the IP payload size divided by the source

information rate.

3. Serialization Delay

Serialization delay is the time required to transmit all of the packet's bits onto the link. It depends on the packet's length and the transmission rate of the link. With higher line speeds, serialization delay can be greatly reduced. The serialization delay is often of the order of microseconds to milliseconds in practice.

B) Network

4. Propagation Delay

The time required to propagate from the beginning of the link to the next node is the propagation delay. The propagation delay is related to the physical medium of the link and its propagation speed, e.g. fiber optics, twisted –pair copper wire, and the distance between two nodes. The propagation delay is the distance between two nodes divided by the propagation speed of the link. In wide-area networks, propagation delay is typically of the order of milliseconds.

5. Queuing Delay and Processing Delay

Queuing delay is the time the packet stays in the node, and it comprises the processing delay and the queuing delay. For the processing delay, time is required for the router to determine the next hop location according to the packet header and the routing mechanisms. This processing time can be much reduced by using fast forwarding schemes like ATM (Asynchronous Transfer Mode) or MPLS (Multiprotocol Label Switching).

Queuing delay occurs when the queue service rate is not fast enough to serve all the incoming packets from different sources using this queue, therefore, the packets accumulate in the queue and wait for service. The queuing behaviour depends on the traffic pattern, i.e. the statistical nature of the arrival process. The queuing delay at a node can vary significantly from packet to packet due to the randomness of the incoming aggregate packet arrival process. So the characteristics of queuing delay typically are given by statistical measures, such as average queuing delay, variance of queuing delay, queuing delay probability function, and the probability that the

queuing delay exceeds some specific value.

C) Receiver end

6. Playback Delay

Due to variable queuing delays, it is impossible for all the packets in a stream to arrive at the receiver with the same inter-arrival time. To allow for variable packet arrival times and to achieve a steady stream of packets, the receiver holds packets in a buffer for a while before playing them. This holding time causes further delay for the application.

7. Decoding and De-packetization Delay

This is the time required to re-assemble the packets to the original form of the signal. They are the counterparts to coding and packetization at the sender, as shown in Table 2-2.

8. Summary of delays

Assuming that a packet starts from the source, passes through a series of routers, and ends in the destination, the packet will suffer several types of delays at each node along the path. These delays include coding delay, packetization delay, transmission delay, queuing delay, propagation delay, playback delay, decoding delay and de-packetization delay. The end-to-end delay is the sum of the delays experienced at each hop on the way to its destination.

Delay introduced above can be divided into two components, a variable component and a fixed component. The only variable part of delay is the time spent in the queues of the network nodes on the transmission path. They are very hard to predict and depend heavily on the current network load. So when network load is stable, investigating how to reduce this variable part of delay for real-time delay sensitive traffic is important for QoS research; the fixed part of delay is independent of network conditions and depends primarily on the number and the performance of network nodes on the transmission path and the link capacity, etc. The fixed component of delay includes all the other delays except queuing delays. These fixed delays are either configurable or easily calculated.

2.2.2 Jitter

Since queuing delay at each hop along the transmission path is variable, the end-to-end delay of packets is also variable. The time interval between the time when a packet is generated at the source until it is received at the receiver can fluctuate from packet to packet. This variability in the inter-arrival times of packets at the receiver is called jitter. Large jitter is unacceptable to real-time traffic and jitter will cause the signal to be distorted which will damage the multimedia traffic.

Eliminating jitter requires collecting packets and storing them long enough to allow the slowest packet to arrive in order to be played contiguously. The common way is to use a playout buffer at the receiver to absorb the jitter before playing the audio or video stream. This introduces additional delay for packets arriving early, but does allow the playing out of packets that arrive to be predictable.

Normally there exists several ways to measure jitter, including variation in the packet inter-arrival times at the receiver, the variation in end-to-end delay, the maximal variation of end-to-end delay, and the deviation from the average end-to-end delay. The first one is a receiver oriented observation of the variation, and it does not eliminate the time intervals between packets, whereas the second is a more network oriented observation, and it is a more direct indicator of a system's performance on multimedia streams, because it eliminates the inter-packet transmission periods of the source. The third one indicates the worst case jitter, which is the difference between the maximum end-to-end delay and minimum end-to-end delay, in other words, the range of the end-to-end delay. The last attribute describes an instantaneous packet delay variation from the mean. Each measurement describes an aspect of delay variation that is relevant to performance [Chun00] [Hus00].

2.2.3 Loss

Packet loss is possible in IP networks because packets from many sources are temporarily stored in a queue prior to transmission over an outgoing link in a router. Since the queue is finite, an arriving packet is lost (or dropped) in the network if there is no space left in the queue when the network is congested. For real-time traffic, loss

also occurs when a packet arrives at its destination but is too late to be played out as part of a continuous bit stream.

Packet loss is expressed as a probability value, i.e. the probability that an individual packet will be discarded by the network. Packet loss can cause severe damage to QoS for real-time audio and video traffic. Although the human brain is capable of coping with some lost information in speech (and loss recovery schemes help to minimize the impact of this loss), too much packet loss makes voice and video unintelligible.

2.2.4 Throughput

Throughput is used to describe the capability of a system to transfer data. It is also a measurement of the reliable data transfer rate, and it can also indicate the performance of the link, the routing systems and the host systems.

2.2.5 Performance Requirements for Real-time Traffic

High quality real-time voice/video imposes many performance requirements on the underlying IP network, and those requirements are different from traditional data traffic. Large end-to-end delay (latency) may cause speech overlap as well as mutual silence at the receiver. As described in [Kur05] [ITU G.114], it is recommended that the one-way end-to-end delay for real-time audio/video traffic should be no longer than 150 milliseconds, because these are not perceived by a human listener. Delays between 150 and 400 milliseconds can be acceptable but not ideal, and delays exceeding 400 milliseconds will be not acceptable at all.

We have assumed the only variable delay component over IP network is the queuing delay, and how to control and predict the variable queuing delay over IP network is the main focus of this thesis. So it is necessary to describe the performance requirements in terms of maximum queuing delay instead of maximum end-to-end delay. [Chua01] faced similar requirements, and we adopt a similar approach and

propose that queuing delay is at most 100 ms (for G.711 PCM) to satisfy the quality[1].

In addition, large packet inter-arrival delays (jitter) cause packets to arrive too late for their designated playback time, which forces the receiver to discard the late packets. The dejitter buffer hold time adds to the overall delay. Therefore, for high jitter, the overall perceived delay is high even if the average delay is low. For example, the overall delay is only 55ms for a moderate average delay of 50ms with a 5ms jitter buffer. In contrast, if the network has a low average delay of 15ms but occasionally the packet is delayed by 100ms, the delay buffer would have to be 100ms, thus the overall delay is 115ms. Normally jitter should be controlled within 50ms to 100ms [Has00].

The loss of a packet is not always disastrous. Firstly, real-time traffic is not so sensitive to packet loss, depending on the codecs and loss recovery schemes used at the receiver. But when packet loss exceeds 10 to 20 percent, then really nothing can be done to achieve acceptable audio quality. So packet losses less than 10 percent are generally tolerable [Meh01] [Kur05] [Has00]. Secondly lost packets can be repaired by some recovery schemes (receiver-based recovery schemes can work for relatively small loss rates (less than 15 percent), and for small packets (4-40 ms) [Per98]) or reconstructed by the human brain.

However this tolerance to packet loss is conditional since consecutive loss may lead to disruptions in the service of voice and video. As [Bor98] described, the consecutive packet loss resulting from congestion at routers is also known as burst loss. Burst loss can have a large impact on the voice and video performance, for example, for video, a “freezing” image is the result, and this effect is further aggravated by some interdependence between packets (i.e. that one frame can only be decoded when a previous frame before has successfully been received and decoded). For voice, as a single packet contains typically several voice frames, this consecutive

[1] Assume that PCM packetization and transcoding delay is negligible, and it takes 30ms propagation delay for voice packets to be transported across the United States [Kar01], so the total queuing delay should be kept within 120ms (150ms-propagation delay). We take 100 ms as the bound in our simulations.

loss degrades the quality of voice as well. In addition, burst loss is difficult to correct due to the effects of sequential errors, for example, recent research in [Sun04] shows that for some codecs (e.g. G.729), loss concealment works well for a single frame loss, but not for consecutive or burst losses. Lastly, some loss recovery schemes only recover a lost frame based on previous frames, thus consecutive loss makes the recovery difficult. In a word, real-time traffic is tolerant to non-consecutive packet loss.

After this description of these QoS performance metrics, how to describe the QoS of real-time traffic is clear. QoS can be expressed in terms of delay, jitter and loss instead of a vague concept of QoS. In the rest of this thesis, the comparison of QoS between different scenarios will be expressed via an objective concept of delay, jitter and loss.

In the next section, we move onto another important part of our research, AQM.

2.3 Active Queue Management (AQM)

2.3.1 Introduction to AQM

The traditional queue management is the drop-tail algorithm, which sets a maximum length (in terms of packets) for each queue, accepts packets into the queue if the queue length is less than the maximum length, and drops subsequent incoming packets if the queue reaches its maximum length until queue length decreases to be less than the maximum length. This method has been used in IP networks for years, but as pointed out in [RFC2309], it has two important drawbacks: lock-out and full queues. The former one means that in some situations drop-tail allows the queue to be monopolized by a single connection or a few flows, thus preventing other connections from getting room in the queue. The second one means that the drop-tail algorithm allows queues to remain almost full for a long period of time, because drop-tail signals congestion via a packet drop only when the queue is full, in other words, the queue will be almost full from the instant that the queue is full to the instant that sources began to reduce the sending rate. So to reduce the steady-state queue size is very important, and this is queue

management's most important goal.

The solution to the problem is for routers to drop packets before a queue becomes full, so that end nodes can respond to congestion before buffers overflow. This proactive approach is called "active queue management". By dropping packets before buffers overflow, active queue management allows routers to control when and how many packets to drop.

Random Early Detection (RED) is a widely used AQM, which we use as the basis of our study for the application of AQM to real-time traffic in the rest of this thesis.

2.3.2 Random Early Detection

2.3.2.1 Introduction to RED

Briefly, the RED algorithm [Flo93] drops arriving packets probabilistically and the probability of drop increases as the estimated average queue size grows. In other words, the RED algorithm consists of two main tasks: a) the estimation of the average queue size at the router and b) packet drop decision. The averaging process smooths out temporary traffic fluctuations and allows small bursts to pass through, dropping packets only during sustained overloads.

Consider a router with a buffer size of B packets. With the RED buffer management scheme, a router detects congestion by the average queue length $\bar{q}$, which is estimated using an exponentially weighted moving average: $\bar{q} = (1 - w_q) \cdot \bar{q} + w_q \cdot q$, where w_q is a fixed small parameter and q is the instantaneous queue length. How quickly $\bar{q}$ responds to bursts can be adjusted by setting w_q. A bigger w_q can result in $\bar{q}$ responding to q more quickly. When the average queue size is less than min_{th}, no packets are dropped. When the average queue size is greater than max_{th}, every arriving packet is dropped. When the average queue size is in the range (min_{th}, max_{th}), incoming packets are probabilistically dropped or marked with the Congestion Experienced bit. The probability that a packet arriving at the RED queue is either dropped or marked depends upon several control parameters of the algorithm. An

initial drop/mark probability P_b is computed based on this equation:

$$p_b = \begin{cases} 0 \quad \text{.....} if\ \bar{q} < \min_{th} \\ 1 \quad \text{.....} if\ \bar{q} > \max_{th} \\ \dfrac{\bar{q} - \min_{th}}{\max_{th} - \min_{th}} \cdot p_{\max} \quad \text{.....} otherwise \end{cases}$$

The actual probability is a function of the initial probability and a count of the number of packets enqueued since the last packet was dropped: $p_a = p_b / (1 - count * p_b)$.

The control parameters of the RED congestion control are thus w_q, min_{th}, max_{th} and p_{max}, of which, [Flo97] recommended the values of weight w_q and p_{max} as 0.002 and 0.1 respectively so that short-term increases in the instantaneous queue size do not affect the estimated average queue size and that the queue size estimate correctly reflects the trend of the instantaneous queue size. As for the values for min_{th} and max_{th}, there are no widely accepted values or guidelines to follow. As pointed out in [Hol01], "tuning of RED parameters has been an inexact science for sometime now".

2.3.2.2 Existing Study about RED

The RED mechanism was originally developed to work in conjunction with transport-layer congestion control protocols such as TCP. It attempts to avoid congestion by managing queue lengths and probabilistically dropping packets before the queue becomes full. TCP connections cooperate by adapting their window size, and hence their load on the network, in response to packet drops. The behaviour of RED in conjunction with TCP has been widely studied, and much work has addressed the difficulties of configuring RED parameters [Gev01].

Previous studies focusing on traffic mixes of TCP and UDP highlight the detrimental behaviour when they share queuing resources, as the widely referred paper [Bon00] analyzed, the TCP traffic suffers reduced throughput under congestion because UDP is (typically) inelastic and so does not back off; and the presence of TCP traffic

sharing a RED queue increases the jitter and loss rate of UDP traffic as well, even though the mean queuing delay is kept within bounds. One solution is to segregate UDP and TCP into separate classes and apply different forwarding treatments. [Yil01] concluded that isolation improves fairness for TCP flows and reduces their drop probability, and UDP traffic obtains a more predictable forwarding environment. Typically this is achieved by employing some form of priority in the scheduler. This may be a strict time priority scheme, a weighted fair queuing mechanism, or a combination which allows strict priority up to a configurable bandwidth limit [Zen04]. An alternative approach is to make the UDP flows responsive to congestion by providing congestion feedback and rate adaptation (e.g. [Men02] [Dra00]).

Although AQM is acknowledged to be a good way to keep mean delays under control, and hence of particular advantage for interactive applications [RFC2309], its behaviour in traffic scenarios comprising only real-time inelastic streams (UDP traffic) has not until now been reported. The RED research that concerns UDP traffic mainly emphasized the unfairness between TCP and UDP traffic and the corresponding effects on the performance of TCP traffic, but no special attention has been paid to RED's effects on UDP traffic. So in this thesis, we take Voice over IP (VoIP) and video as examples of inelastic UDP traffic that has real-time QoS requirements, and give a thorough study about the effects of RED on UDP traffic in terms of delay, jitter and loss.

Chapter 3 Traffic Models & Queuing Behaviors

Traffic models are at the heart of performance evaluation of telecommunications networks. A traffic model is used to mimic real network traffic. An accurate traffic model can assist the network performance analysis either through analytical techniques or computer simulations. Good traffic models allow a parameterization of the essential statistical characteristics of real network traffic. So by generating the traffic under these traffic models for a range of their parameters and then running simulations for each of these generated sources, we can say that the network has been tested under a representative set of plausible scenarios. In the absence of accurate traffic models the only way to study the performance of a network would be to simulate the network using real life traffic sources.

In this thesis we study the QoS improvements gained by configuring RED for real-time voice and video traffic. So firstly, we will give a brief literature review about commonly used voice/video traffic models and the voice/video models used in this thesis. Secondly, because queuing theory plays an important role in the design and performance analysis of telecommunication networks, the analysis for the queuing behaviour of voice traffic is given. These theoretical results based on well studied traditional models, are used as a validation for the simulations.

3.1 Traffic Models

3.1.1 Voice Traffic Model

3.1.1.1 A Single on/off Source

Modeling of voice traffic has been studied extensively in the literature, and has resulted in several models. For a detailed explanation of voice models see [Min98]. In general, these models represent speech as a Markov Chain with different numbers of states. The greater the number of states in the model, the more complicated the model is. Among these models, the simplest model for voice traffic is the on/off model first

proposed in [Bra69], where a two-state chain is assumed. This is the most widely used model for its simplicity, and highly acceptable results. The two states correspond to the talk spurt (On state) and silence periods (Off state). The transitional rates from the On to Off state and from Off to On state are α and β respectively, and the holding times in On and Off periods are assumed to be exponentially distributed with mean $1/\alpha$ and $1/\beta$ respectively. The activity factor, giving the proportion of time that the voice source is on, is $\frac{\beta}{\alpha+\beta}$. Figure 1 shows the on/off model for voice as an example. When the voice source is in the On state, fixed-size packets are generated at a constant interval. No packets are transmitted when the voice source is Off. The size of the packet and the rate at which the packets are sent depends on the corresponding voice codecs and compression schemes as mentioned in section 2.1.1.

Another single on/off model, the Interrupted Poisson Process (IPP) traffic model, is similar to the on/off traffic model introduced above, and has two states: Active and Idle. The time spent in Active and Idle states is exponentially distributed with mean $1/\alpha$ and $1/\beta$ respectively. But arrivals in the active state occur with a Poisson distribution with rate λ instead of constant bit rates. Therefore, IPP and on/off models differ only in their inter-arrival times during the active (On) state.

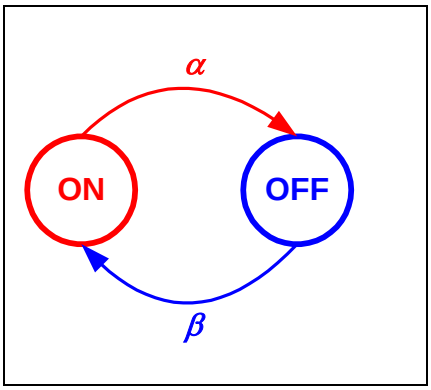

Figure 3-1 On/off model for voice

3.1.1.2 Superposition of on/off Sources

Various approximation models have been proposed and evaluated for the packet arrival process from multiple packet voice sources. [Hef86] attempted to

approximate superposition of voice streams by a Markov-Modulated Poisson Process (MMPP). This MMPP model is also used in [Nag91], and was found to be a better approximation in [Bai91].

The IPP model is a simple two state example of MMPP model, whose on state is with an associated positive Poisson rate, and the off state with associated rate zero. This basic MMPP model can be extended to aggregations of independent traffic sources, each of which is an MMPP, modulated by an individual Markov process Mi. So an MMPP process with M + 1 states can be obtained by the superposition of M identical independent IPP sources.

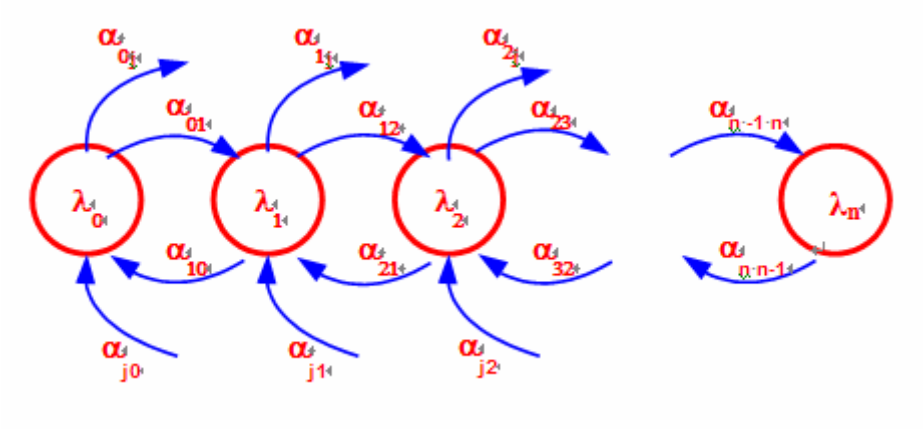

Figure 3-2 MMPP process

The MMPP model combines the simplicity of the modulating (Markov) process with that of the modulated (Poisson) process. MMPP uses the Poisson process as the modulated mechanism as shown in Figure 3-2. This model is fully characterized by the transition matrix $\{\alpha_{ij}\}$ and the Poisson arrival rate in each state λ_i. In this model, while in state s_k, the arrivals occur according to a Poisson process with rate λ_k. In this case, the modulation mechanism simply stipulates that in state k of s_k, arrivals occur according to a Poisson process at rate λ_k. As the state changes, so does the rate.

The superposition of on/off sources has also been found to mimic real traffic well and is found frequently in literature. Both [Bai91] and [Han94] use superposition of on/off models.

3.1.1.3 VoIP Models Used in This Thesis

VoIP refers to real-time delivery of packet voice across IP networks. The rapid growth of IP-based packet networks and the overall bandwidth efficiency of an integrated IP network make it an attractive candidate for voice transport. This multiplexing of data and voice results in a better bandwidth utilization than for traditional circuit-switched voice. So VoIP has become one of the key applications for recent telecommunication services.

On/off sources are widely accepted as models of telephony traffic, but the exact values for on/off times are not unique. In the case of on/off time, the most widely used values in the research area of VoIP are either Ton=0.352s and Toff=0.65s or Ton=1.004s and Toff=1.587s. [Sri86] [Ali04] [Den95] [Zen04] etc used the former model, while [Chua01] [ITU P.59] etc used the latter model. To give a full analysis, the voice models used in this thesis consist of both sets of parameter values. The size of the packet and the rate at which packets are sent depends on the corresponding voice codecs and compression schemes as listed in Table 2-1.

Model 1: The holding time in the talk state and the silence state is assumed to be exponentially distributed with mean times of 352 ms and 650 ms, respectively. And the voice source generates constant bit rate (CBR) traffic of 64Kbps when it is active (G.711), and the bandwidth required including header is 85.6Kbps. Packet size is 214 bytes, and packet rate is 50 pps (packet per second).

Model 2: Similar to model 1, but the average talk spurt and silence period is set as 1.004s and 1.587s, G.711 is used and the total bandwidth required to transmit packets when active is 85.6Kbps. Packet size is 214 bytes, and packet rate is 50 pps.

Model 3: Codec G.729 is used, whose bandwidth required when active is 29.6Kbps, and Ton=1.004s, Toff=1.587s. Packet size is 74 bytes, and packet rate is 50 pps.

Model 4: G.723.1 is used, with a sending rate of 17.6Kbps when active, Ton=0.352s and Toff=0.65s. Packet size is 84 bytes, and packet rate is 26 pps.

To evaluate the robustness of our proposed mechanisms against the diversity of Internet workloads, we will consider these four kinds of traffic source models in the

simulations. Each of these models has its own distinct statistical properties and can be used to represent a variety of latency sensitive applications.

3.1.2 Real-time Video Traffic

VoIP traffic can be modeled by the superposition of on/off traffic with constant fixed packet size and constant sending rate when active. But video traffic is a more complex real-time traffic. It has variable bit rate, variable sending rate and variable packet size. Real-time video traffic has distinctly complex characteristics, and a well accepted theoretical model is not available. Video traces are often used to study the performance of video traffic and the video traces we use to conduct our study are obtained from an online trace library [Lib]. We chose this public library because of the diversity of the traces it provides and it is widely used in network performance and video quality studies, e.g. in [Ala05] [See04] [Xu05] [Ziv02]. This library contains traces of 26 different hour-long video sequences, each encoded using two different standards, the MPEG4 and the H.263 standard, and under three different quality indices (low, medium and high). Moreover, for each recorded trace, statistics reflecting its characteristics are also available. The detailed description of the trace collection procedure and the methodology used to extract the trace statistics can be found in [Fit01].

3.2 Queuing Behaviours for Voice Traffic

As described in section 2.2.1, the only variable part of end-to-end delay is caused by the queuing delay within routers, so if the queuing delay is estimated properly, then the end-to-end delay and its variation (jitter) are known, which will greatly help to predict the QoS of real-time traffic. So in this section, the analysis about queuing behaviors (queue length) which determine the queuing delay is given.

3.2.1 Queue Length Distribution

A variety of queuing analysis has been performed to study the queue length behaviour under various conditions, and a queuing analysis based on Large Deviation Theory (LDT) [Duf98] [Cha00] reveals that, for wide-ranging traffic input, the tail of the

queue length distribution decays exponentially in the regime of large buffers. The rate of decay is known as the decay rate and the offset is the decay constant. The decay rate can be estimated by using the arrival rate function using the service rate. Traffic measurement is used to estimate the cumulate generating function of the arrival process. The measurement method is outlined in [Duf98]: in each time slot *s*, the traffic volume (in terms of either number of packets or total number of received bits) is measured and recorded. The traffic sample $(X_i)_{i=1,2,\ldots T}$ is further grouped and the measurement window is divided into *n* blocks of length *t* such that the aggregate arrival $Y_{j,t}$ over time *t* is defined as follows:

$$Y_{j,t} = \sum_{i=(j-1)t+1}^{tj} X_i \qquad \text{Equation 3-1}$$

The cumulate generating function for the traffic arrival $\Lambda_{A,t}(\theta)$ is estimated as follows:

$$\Lambda_{A,t}(\theta) = \frac{1}{t}\log\left(\sum_{i=1}^{n} e^{\theta Y_{i,t}} \Big/ n\right) \qquad \text{Equation 3-2}$$

Likewise, the cumulate generating function for traffic departure $\Lambda_{B,t}(-\theta)$ is given as:

$$\Lambda_{B,t}(\theta) = \frac{1}{t}\log\left(\sum_{i=1}^{n} e^{-\theta Z_{i,t}} \Big/ n\right) \qquad \text{Equation 3-3}$$

where $Z_{i,t}$ denotes the aggregate departure amount for period *t* of sample *i*.

In a special case G/D/1, when the departure rate is constant for a fixed bandwidth *C*, then Equation 3-3 becomes:

$$\Lambda_{B,t}(\theta) = -\theta \cdot C \qquad \text{Equation 3-4}$$

From the Large Deviation Theory analysis, the tail of the queue length process for wide-range of traffic model satisfies the following equation:

$$\lim_{x\to\infty} \frac{1}{x} \log P[Q \geq x] = -\theta^* \qquad \text{Equation 3-5}$$

where $\theta^* > 0$ is the largest root of the equation.

$$\Lambda_A(\theta) + \Lambda_B(-\theta) = 0 \qquad \text{Equation 3-6}$$

where $\Lambda_A(\theta)$ and $\Lambda_B(-\theta)$ are the cumulate generating function as defined as in Equation 3-2 and Equation 3-3 respectively that can be estimated from measurement.

Then, the queue length process is estimated as:

$$P[Q \geq x] \approx e^{-\theta^* x} \qquad \text{Equation 3-7}$$

Equation 3-7 can be used to estimate the overflow probability in a queue with a large finite buffer of size *x*. If we are interested in the queue length distribution, i.e. $P[Q=x]$, then it can be estimated from Equation 3-7 as follows:

$$P[Q = x] = \frac{d}{dx}(1 - P[Q \geq x]) \approx \theta^* e^{-\theta^* x} = c\eta \qquad \text{Equation 3-8}$$

From Equation 3-8, it is obvious that the queue length distribution decays exponentially. In a log-linear plot, this queue length distribution forms approximately a straight line with offset c and the slope logη. The symbol c and η are the decay constant and decay rate which are equal to θ^* and $e^{-\theta^*}$ respectively. More details about LDT can be found in [Duf98] [Cha00]. But the queue length distribution estimated by LDT is an asymptotic one, therefore, a tight estimation of the queue length distribution is not guaranteed.

With Markovian traffic, as a model of multiplexing voice traffic, the queue length

distribution also decays exponentially for a FIFO queue. This exponential decay of the queue tail distribution has also been studied in various papers, for example [Aba95] [Jel95] [Nor91] [Pit01] [Rob91]. [Pit01] used Excess Rate (ER) to analyze the decay rate of exponential decay of the queue tail distribution, and this thesis adopts ER analysis in estimating the theoretical distribution, (details of ER analysis are given in section 3.2.4).

Figure 3-3 depicts the typical queue length distribution for a FIFO queue multiplexing Markovian traffic. The key feature is the exponential decay of queue state, in other words, the state probability tends to form two straight lines when the queue size (state) is plotted on a log-linear scale. This is a common feature of queuing systems. If we define the state probability as

S(k) = Pr {there are k packets in the queuing system}

The decay rate (DR) can be defined as the ratio [Pit01]:

$$dr = \frac{s(k+1)}{s(k)} \qquad \text{Equation 3-9}$$

Then the two straight lines, each with a different decay rate, form the queue length distribution of Markovian traffic, the two regions are called packet-scale region and burst-scale region [Pit01] respectively. The explanation of this behaviour is given in the following sections.

3.2.2 Packet Scale Queuing

As shown in Figure 3-3, the packet-scale has the steepest decay rate, and dominates for small queue sizes. Packet scale queuing occurs when a bunch of packets from different sources happen to arrive together, even though the aggregate arrival rate is less than the buffer's service rate. It results in packet loss for a small buffer size, but the packet loss probability drops rapidly as the buffer size increases. This behaviour is always present and arises because streams are being multiplexed together from diverse inputs. Details of packet scale queuing can be found in [Rob91] [Nor91] [Pit01].

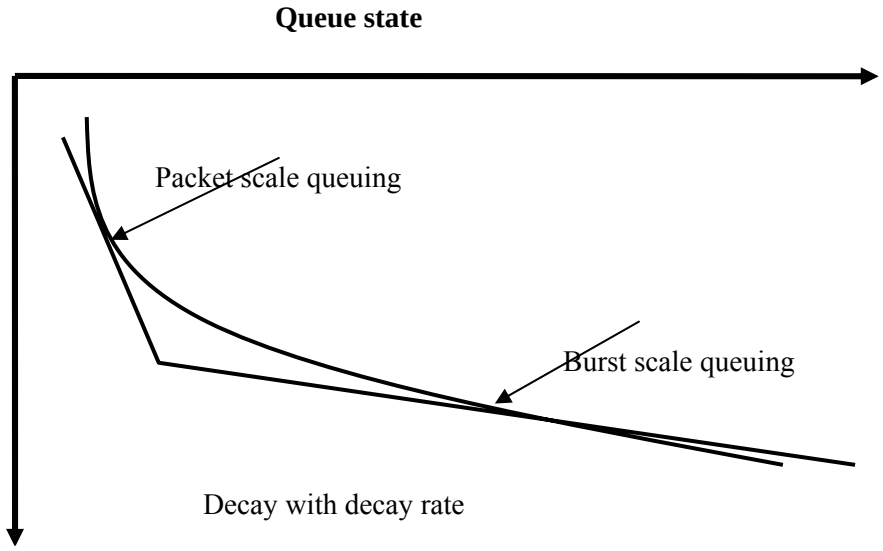

Figure 3-3 Typical Characteristic of Queue State pdf for Multiple on/off Sources

3.2.3 Burst Scale Queuing

The burst-scale component normally is less steep, and dominates queue behaviour at larger queue sizes. Burst scale queuing occurs when the overall arrival rate is temporarily greater than the buffer's service rate. Burst scale reflects the accumulation of delay when the instantaneous arrival rate of the superposition is greater than the multiplex link service rate.

A queue begins to build up when the service rate is smaller than the arrival rate. In one of the nodes in a real network, when N on/off sources are aggregated together, the aggregated arrival rate can be greater than the service rate over a certain period. During this period the queue builds up. When sufficient sources are in their OFF period and so the aggregate arrival rate is less than the service rate, the queue size begins to decrease. If the period of congestion is sufficiently short, bursts can be accommodated in the buffer and losses will not occur. Normally if the buffer is large, the packet loss rate will be low although this is at the cost of a long queuing delay in the queue.

To summarize, packet-scale queuing behaviour is caused by coincident packet arrivals from more than one connection, although the aggregate arrival rate is smaller

than the service rate. As for burst scale, this happens when several connections generate bursts of packets simultaneously, and the arrival rate is greater than the service rate. If this phenomenon lasts for a sufficiently long period, the queue begins to fill and the burst-scale queuing effect occurs. The more connections that are in a burst period simultaneously, the larger the queue length. But as it is unlikely that all connections send bursts at the same time, very large queue lengths have a small probability.

As shown in Figure 3-3, an interception point connects the packet scale with burst scale. [Mic97] indicated some general trends in this behaviour: when the number of connections or the peak rate increases, which means the load applied to the queue increases, the interception moves leftward, reducing the packet-scale queuing, and the burst-scale becomes flatter (bigger). If the load applied to the queue is kept constant but the burst length is increased, the knee point is more or less unchanged but makes the burst component is flatter (bigger).

As described in [Pit01], a formula which describes the relationship between average queue size q and decay rate dr can be deduced:

$$q = \frac{dr}{1 - dr} \qquad 0<dr<1 \qquad \text{Equation 3-10}$$

The dr in Equation 3-10 can be either the decay rate of the packet scale or the burst scale, and so this equation will calculate the long-term average (based on decay rate of packet scale) or the conditional average queue size (based on decay rate of burst scale) respectively. Obviously, a bigger dr will result in a bigger q, and it means that bigger (flatter) burst scale results in a bigger conditional AQS, which will increase the queuing delay and jitter as well. In other words, when the burst scale becomes bigger (flatter), the mean queue size q increases. We will use this conclusion later.

3.2.4 ER Queue Length Distribution Expression

Since the typical queue length distribution of a FIFO queue multiplexing on/off sources can be regarded as two straight lines, the queue state probability q(x) can be modeled by the following expressions with decay constant and decay rate [Leu03].

In the packet scale region:

$$q(x) = \Pr(X = x) = c_p \cdot \eta_p^x \qquad \text{Equation 3-11}$$

In the burst scale region:

$$q(x) = \Pr(X = x) = c_b \cdot \eta_b^x \qquad \text{Equation 3-12}$$

Here, c_p and c_b are the decay constants of packet scale and burst scale respectively, η_p and η_b are the decay rates of packet scale and burst scale respectively. In this thesis, our theoretical analysis about this queuing behaviour is based on previous research results named Excess Rate (ER) analysis [Sch96] [Pit01]. ER arrivals occur when the service rate is smaller than the instantaneous packet arrival rate, and these ER arrivals must be buffered in the queue or lost if the queue is full. The excess-rate packets mean the packets that can not be served upon arrival because they arise from "excess-rate" bursts. So in essence the queuing analysis evaluates the demand for the temporary storage of these excess rate packets.

The packet-scale decay rate (for fixed packet size) can be approximated by GAPP analysis [Sch96] as shown in:

$$\eta_p = \frac{\rho e^{\rho} - e^{\rho} - \rho^2 + \rho + e^{-\rho}}{\rho - 1 + e^{-\rho}} \qquad \text{Equation 3-13}$$

Where ρ denotes the utilization of the queue.

A closed-form expression for the burst-scale decay rate is also available [Pit01] and details are attached in appendix A.

$$\eta_b = \frac{1 - \dfrac{1}{T_{(on)}\left(R_{on} - C\right)}}{1 - \dfrac{1}{T_{(off)}\left(C - R_{off}\right)}} \qquad \text{Equation 3-14}$$

Queue state probability of Packet scale:

$$q(X) = (1 - p_{er}) * (1 - \eta_p) * \eta_p^X \quad \text{Equation 3-15}$$

Queue state probability of burst scale:

$$q(X) = p_{er} * (1 - \eta_b) * \eta_b^X \quad \text{Equation 3-16}$$

Overall queue state probability:

$$q(X) = p_{er} * (1 - \eta_b) * \eta_b^X + (1 - p_{er}) * (1 - \eta_p) * \eta_p^X \quad \text{Equation3-17}$$

Where, p_{er} is the probability that a packet is an excess packet.

As shown in figure 3-3, there is a partition point between packet scale and burst scale

$$p_{er} * (1 - \eta_b) * \eta_b^X = (1 - p_{er}) * (1 - \eta_p) * \eta_p^X \quad \text{Equation3-18}$$

Then the partition point can be expressed as

$$x = \frac{\ln(\frac{(1 - p_{er}) * (1 - \eta_p)}{p_{er} * (1 - \eta_b)})}{\ln \frac{\eta_b}{\eta_p}} \quad \text{Equation3-19}$$

These theoretic formulas will be compared with our simulation results as a validation for those simulation results.

3.3 End to End Delay Distribution

The length of a queue within a buffer affects the packet delay, loss and jitter within the buffer. There is a direct relationship between the queue length and the queuing delay. The larger the queue length when a packet arrives, the longer the queuing delay it will experience. When a new incoming packet arrives, the existing queue

length reflects how long this packet needs to wait before being served (If FIFO is assumed as the scheduling discipline). Therefore, the statistic or the information of this queue length is useful to predict the queuing delay of this incoming packet. For instance, if the packet size is fixed at 1000 bits and the service rate for the queue is 10Kbps, then the queuing time for this incoming packet will be 1 second if the existing queue length is 10. Therefore, the queue state probability density function (pdf) can be used to approximate the queuing delay pdf when all the packets are of fixed size.

In Section 2.2.1, we explained the various delay components existing in a network. This consists of a variable component and a fixed component. The queuing delay contributes to the variable delay, while other delays remains fixed. Although the processing delay and transmission delay may cause a small delay variation between different packets due to different packet size and different path, in this thesis, we assume both factors are less significant compared to the queuing delay. A similar assumption can be found in [Yat93] [Rib00]. Since the variation of the packet delay is assumed to be due to queuing delay, the end-to-end delay is the sum of the queuing delays experienced at each hop and a constant delay time caused by the fixed delay component.

3.4 Simulation Verifications

This section discusses some basic concepts related to the simulations in this thesis. In modern IP networks, because of the size and the complexity of the network, it is very difficult, or even impossible to investigate performance using analytic methods. As a result, simulation plays an important role in the performance analysis of a network. With simulation techniques, a network can be modeled to an arbitrary level of detail, thus providing a good tool for evaluating the behaviour of interest. In some cases, simulation is the only feasible modeling approach. [Kur88] gave an example of how often people tend to use simulation instead of other approaches. There are now available well-developed software packages, e.g. OPNET[2], NS2[3] etc, and these

[2] OPNET® is a commercial simulation tool with established libraries developed by Mil3.

provide an excellent platform for network modeling with built-in standard components.

Certain components of a network (buffer queues in particular) were studied in this research, and implementations were done using NS2. To verify that the simulation results are correct, firstly, we run simulations whose theoretical queuing behaviors are known, and compare the simulation results with the theoretical results to verify the accuracy of the simulation. Secondly, more simulations are run to evaluate the performance with diverse types of workload and queue management algorithm.

Next a comparison between theoretical and simulation results is given. The simulations were investigated using the NS2 simulator. Multiple on/off sources are multiplexed into a single queue with FIFO scheduling. The drop-tail mode is used (with a capacity of 1000 packets to ensure zero loss). To form a load of 80% on a single bottle-neck link (a T1 link of 1.544 Mbps), 41 voice model 1 and 37 voice model 2 (listed in section 3.1.1) are used as source traffic respectively. Meanwhile, the queue state (queue size) is monitored and recorded to generate the queue state probability distribution.

The theoretical decay rates for the burst and packet scale components are based on the equations introduced in Appendix A and section 3.2.4. Table 3-1 lists the corresponding theoretical results for the two voice models.

	Packet scale dr	Burst scale dr	p_{er}	Partition point x
Voice model 1	0.6587	0.9757	0.076083	13
Voice model 2	0.6542	0.9907	0.07184	15

Table 3-1 Theoretical results

As shown in Figures 3-4 and 3-5, the black simulation curve corresponds closely to the red theoretical curve drawn by Equation 3-11. So the method we use to measure

[3] NS2 is the 2nd version of NS. It is a jointed project developed by UC Berkeley, USC/ISS, LBL and Xerox PARC for discrete event simulation modelling. The software is available at http://www.isi.edu/nsnam/ns/

the queue size and the way we analyze the data is accurate, and similar measurements can be used to show the difference between different workloads and queue management algorithms.

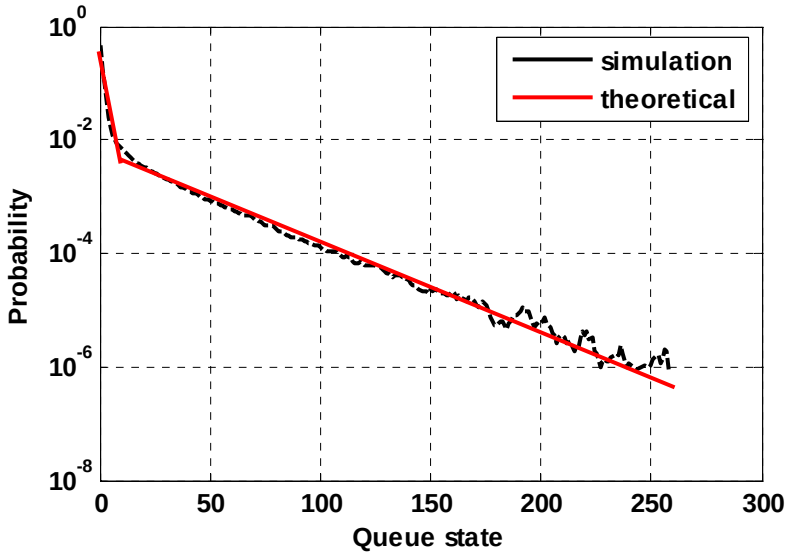

Figure 3-4 Simulation and theoretical result for voice model 1

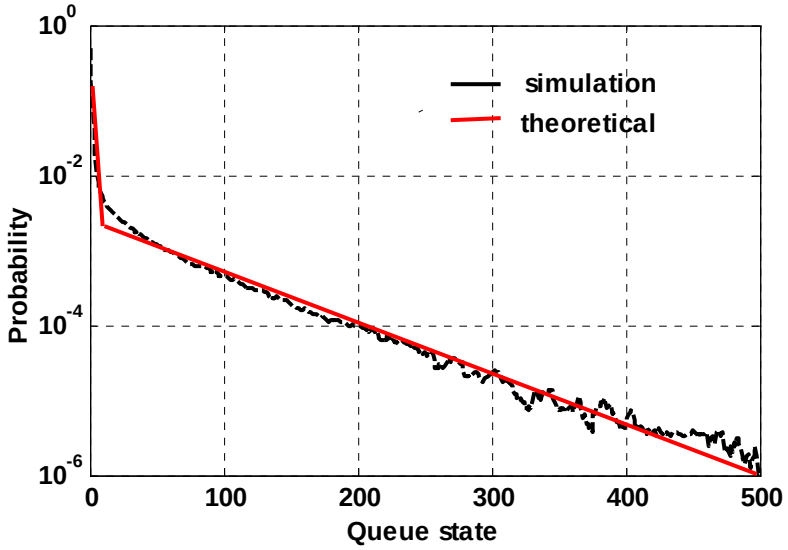

Figure 3-5 Simulation and theoretical result for voice model 2

After the verification of the accuracy of our simulations, in the next chapter we will begin with the decay rate analysis for the RED algorithm to give a full evaluation of its queuing behaviour and corresponding performance measure of delay, jitter and loss.

Chapter 4 Using RED to Improve VoIP QoS

As an active queue management algorithm, RED is useful in controlling the average queue size (queuing delay) with or even without the presence of a cooperating transport protocol. This control of queuing delay is extremely helpful to delay sensitive real-time traffic. However RED was widely studied to avoid TCP congestion collapse, the application of RED to UDP traffic has only been studied in the context of mixed TCP/UDP scenarios until now.

In this chapter, we run extensive simulations with the NS2 network simulator to study the performance of voice traffic under RED, including decay rate, delay, jitter, effective loss and throughput. We take voice over IP as an example of inelastic UDP traffic that has real-time QoS requirements, and assume typical talk spurt-silence (on/off) behaviour. In addition, because the FIFO drop-tail is the most widely used queue management algorithm in today's IP network, we compare our simulation results under RED with the results for FIFO drop-tail to quantify the improvements.

In our simulations, we use the single-bottleneck topology to uncover and identify the important issues. This dumbbell topology is widely used in research about queue management algorithms and differentiated service [Flo02].

The single-bottleneck topology is shown in Figure 4-1. N voice connections are made between sources on the left and their corresponding sinks on the right. There is a bottleneck link between Router 1 and Router 2, and the bottleneck link bandwidth is a T1 link of 1.544Mbps. Other non-bottleneck links have 10Mbps capacity. Router 1 uses either FIFO scheduling and drop-tail queue management or RED queue management. The parameters for the RED algorithm are adjusted based on different scenarios. The buffer size for drop-tail is set as 1000 packets, which is big enough to avoid packet loss due to physical buffer overflow. With drop-tail, no special effort, e.g. early drop, priority drop etc, is made for the arriving packets so the rough queue state of the voice traffic exists.

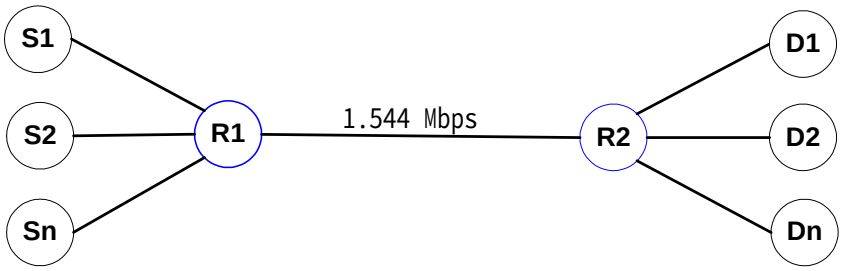

Figure 4-1 Single bottleneck topology

4.1 Decay Rate Filter to Voice Traffic by RED

As we introduced earlier, decay rate reflects the queue state distribution, and hence the queuing delay. First, we study the difference between the decay rate of voice traffic under RED and drop-tail algorithms. Next we apply the RED algorithm with different parameter values to the same VoIP traffic mix to observe the corresponding queue state distribution.

The VoIP traffic parameters and typical values are given in section 3.1.1. The details of the scenario parameter values are as follows:

Number of VoIP sources, N = 41

Mean talkspurt duration, Ton = 0.352 s

Mean silence duration, Toff = 0.65 s

Packet size, b = 214 octets

Service capacity = 902 packets per second (pps) (T1 link of 1.544Mbps)

Source packet rate during talkspurt = 50 pps (G.711 peak rate of 85.6Kbps)

Each VoIP source contributes a mean load of 17.6 pps which, with 41 sources, gives a total mean load of 80% of service capacity. Based on the analysis in section 3.2.4, the packet and burst scale conditional mean queue sizes are approximately 2 and 24 respectively. These will serve as a reference value for threshold settings.

The scenario was investigated using the NS2 simulator with N=41 on/off sources multiplexed into a single queue with FIFO scheduling and either drop-tail or RED AQM. The drop-tail mode was used (with a capacity of 1000 packets to ensure zero loss) so that a benchmark validation could be obtained. We investigated w_q and p_{max}, and found that queue state (thus delay and jitter) is driven by RED thresholds. So for the RED simulations, we set $w_q = 0.002$, $p_{max} = 0.1$(as the recommendations in [Flo97]), and mainly investigated the effects of changing the minimum and maximum thresholds for the queue state. We use the notation RED (min_{th}, max_{th}) to denote the specific configuration. We fix the random number generator seed for each run to obtain comparable results when we compare different scenarios. The queue state distribution of a single simulation run is shown for drop-tail in Figure 4-2. This probability distribution tells us the probability of a particular value occurring.

In Figure 4-2, the thresholds are set below, between, and above the conditional mean queue sizes (packet scale: 2; burst scale: 24) and their sum. In the latter case, RED (26, 50), the queue state probability is quite close to the drop-tail result without consideration of fluctuations of small probability. The minimum threshold is set to the sum of the mean queue size of both packet scale and burst scale, and this appears to be big enough to hold the temporary extra packets in the queue without triggering RED's early drop. For the case that the minimum threshold is set at 1, and the maximum threshold is varied from 10 down to 2, the results show a progressive removal of the burst-scale component, such that for RED (1, 2) only packet-scale queuing is present. For the two cases with thresholds between, i.e. RED (5, 20) and RED (5, 10), a steeper burst-scale component is evident, and the larger the maximum threshold the flatter the burst-scale. For the case with the same maximum threshold but a different minimum threshold, i.e. RED (1, 10) and RED (5, 10), there is little difference between them, which suggests that the minimum threshold does not affect the decay rate so much as the maximum threshold.

Figure 4-2 only gives the queue state probability under different parameters for voice model 1. As for other models introduced in section 3.1.1.3, similar distribution can be achieved.

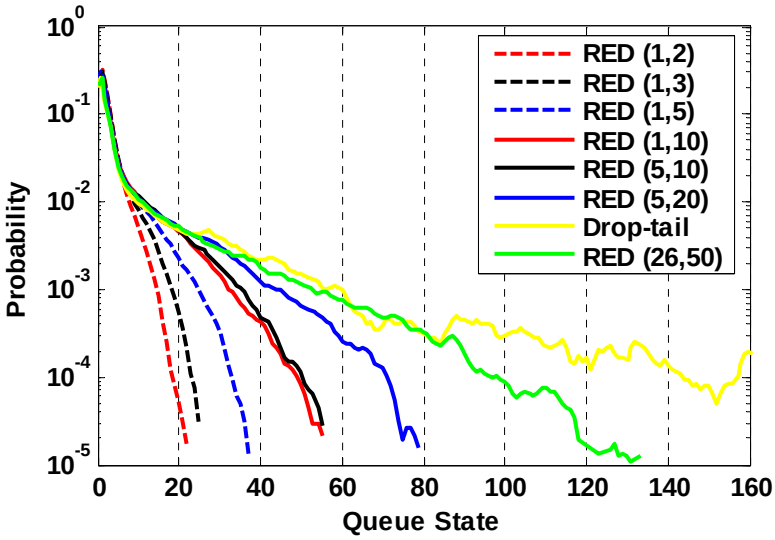

Figure 4-2 Queue state probability under different parameters for voice model 1

Figure 4-3 gives the corresponding best-fit lines based on least squares curve fitting. Through calculating the slope of these best-fit lines, the values of decay rate for each scenario can be calculated. Descriptions about the least squares curve fitting method can be found in Appendix C.

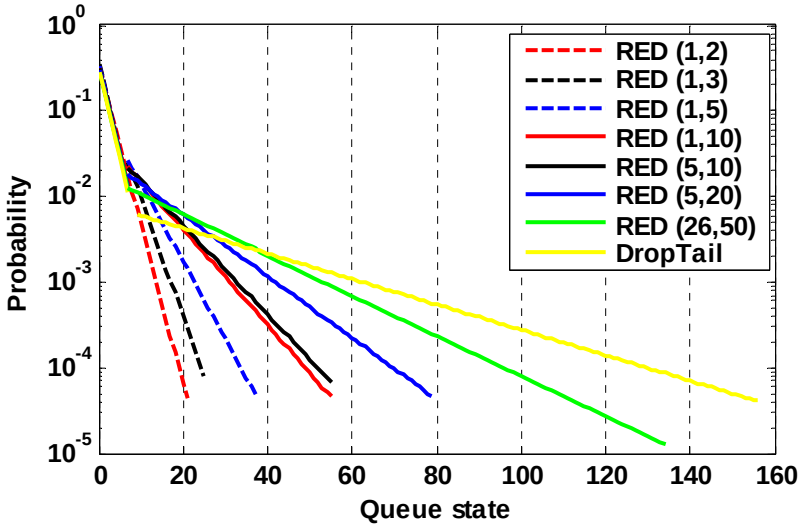

Figure 4-3 Best fit lines of queue state probability under different RED parameters

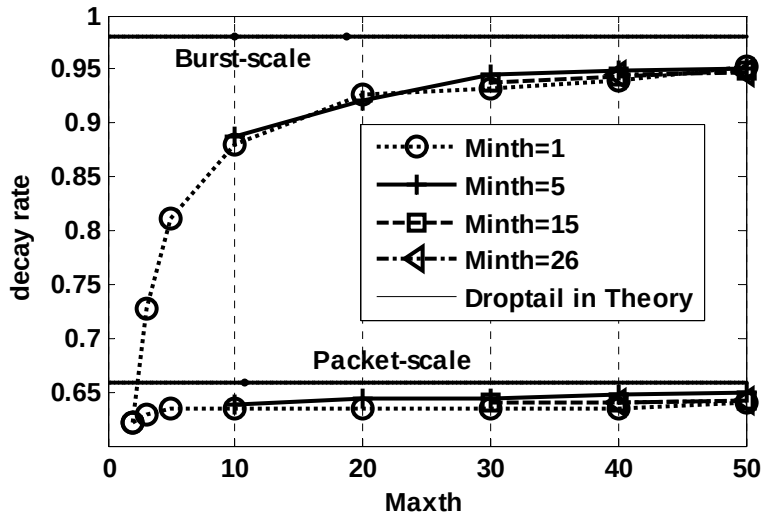

Figure 4-4 Decay rate Vs. max threshold

Through those calculated decay rates of both packet scale and burst scale, Figure 4-4 gives a clearer view of how the decay rates change with the maximum threshold for more parameter settings, in which four different minimum threshold values: 1, 5, 15 and 26 are used. The values of these decay rates in Figure 4-4 can be found in Appendix B.

These results show that the RED mechanism is able to regulate both packet- and burst-scale decay rates of queuing behaviour for inelastic voice traffic with real-time QoS requirements, especially for the burst scale component, which dominates larger queue sizes. Different RED thresholds can result in different decay rates, with the maximum threshold affecting the decay rate the most. There is only a marginal difference in decay rate with the same maximum threshold but different minimum thresholds. In addition, a larger maximum threshold will result in a bigger (flatter) burst scale decay rate, in other words, if maximum threshold is decreased, the results show a progressive removal of the burst-scale component, while packet scale decay rate is quite stable whatever the minimum and maximum thresholds are.

As shown in Figure 4-3, this gradual removal of the burst scale component reduces the probability of the queue size being large, which means that not only the mean queue size (and hence the mean queuing delay) but also the jitter (e.g. as measured by the standard deviation of the queuing delay) experienced by such traffic can be controlled by adjusting the minimum and maximum thresholds of the RED

mechanism. In the following section, a detailed analysis for the delay and jitter is given.

4.2 Delay

We have analyzed the control (removal) of the burst scale decay rate by RED for VoIP traffic, but the relationship between this removal and the QoS is not direct. In this section, we will explain this removal in terms of delay, which has a straightforward relationship with QoS.

As introduced in section 3.3, the queue state distribution has approximately the same shape as the queuing delay distribution when all the packets have a fixed size, the only difference is the scale on the X axis. Here, rather than rescaling the same graphs, we give the corresponding Complementary Cumulative Distribution instead.

In probability theory, the Cumulative Distribution Function (abbreviated CDF) describes the probability that the random variable X is less than or equal to x, for every value x.

$$F(x) = P(X \leq x)$$

To ask how often the random variable is above a particular level, we need the Complementary Cumulative Distribution Function (CCDF), defined as

$$Fc(x) = P(X > x) = 1 - F(x) .$$

As introduced in section 2.2.5, to study the probability that delay is beyond the maximum acceptable bound, the complementary cumulative queuing delay distribution, which gives the probability that delay is above a particular bound, can give a better indicator of whether the delays satisfy the requirements. Figure 4-5 draws the corresponding complementary cumulative queuing delay distribution for the same scenarios as Figure 4-2. Different RED thresholds are set to show the effect on the complementary cumulative delay distribution, and are compared with a simple drop-tail algorithm.

As shown in Figure 4-5, if we set the bound of queuing delay as 0.1s (100 ms),

(which means the packets whose queuing delay is larger than 0.1s arrive at the receiver too late to be of use), we can find that, firstly, if the RED maximum thresholds are set the same while minimum thresholds are different, there is little difference between these curves. Secondly, under smaller RED maximum thresholds such as maximum thresholds of 10 or 20, there are not any packets beyond the 100ms bound due to RED's control. Thirdly, if the maximum thresholds are increased, the portion of packets beyond the bound increases as well. Through these curves for complementary cumulative delay distribution under different RED thresholds, RED's control of delay is obvious. We can select different RED thresholds to obtain different queuing delay.

Figure 4-5 draws the complementary cumulative delay distribution for a load of 0.8, which is not a heavily loaded condition. How about a congested condition? In fact, this delay control by RED is more obvious in a heavily loaded network. Figure 4-6 and Figure 4-7 give the complementary cumulative queuing delay distribution for the same VoIP traffic for loads of 0.9 and 1.0, which are heavily loaded situations. More scenarios are drawn in these three figures to give a clearer view of the effects of RED thresholds.

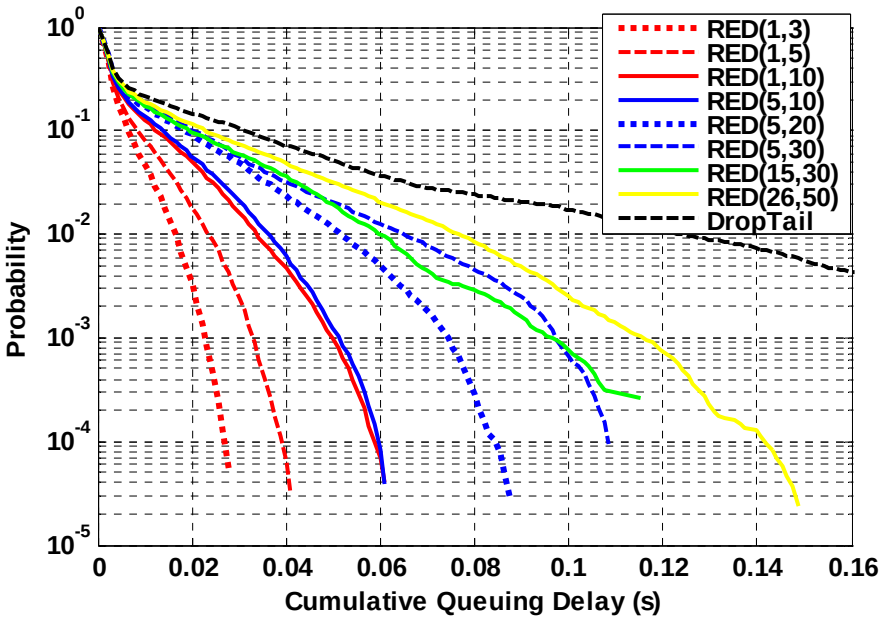

Figure 4-5 Complementary Cumulative Queuing Delay distribution (load=0.8)

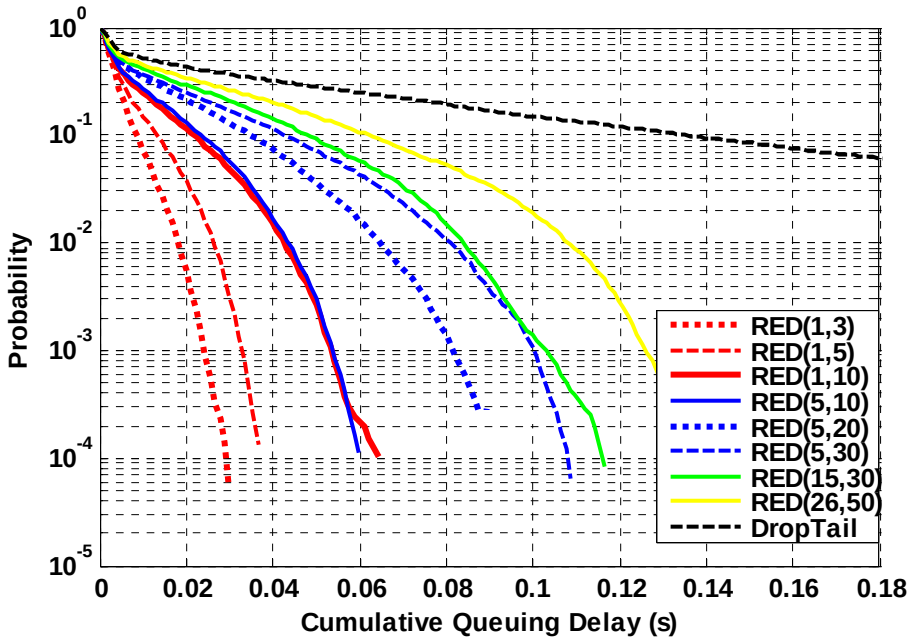

Figure 4-6 Complementary Cumulative Queuing Delay distribution (load=0.9)

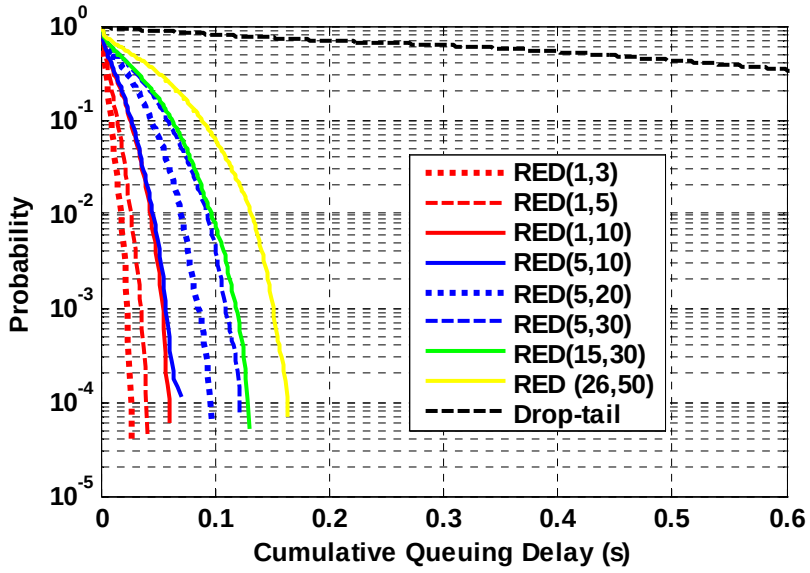

Figure 4-7 Complementary Cumulative Queuing Delay distribution (load=1.0)

Obviously, when the network load is higher, such as 0.9 and 1.0, the out of bound probability for drop-tail will increase greatly and becomes far beyond the requirements, so the perceived performance is very poor. With a proper configuration of the RED algorithm, the out of bound probability can be reduced greatly.

Table 4-1 gives the value of out of bound probability shown in Figure 4-5, Figure 4-6

and Figure 4-7. The difference between the out of bound probability of different RED thresholds and drop-tail is obvious, which means the effects of different RED thresholds are significant and a further study is worthwhile.

	Load=0.8	Load=0.9	Load=1.0
RED (5,30)	0.0594%	0.0858%	0.393%
RED (15,30)	0.0703%	0.125%	0.722%
RED (26,50)	0.2346%	1.77%	5.92%
DropTail	1.71%	14.79%	80.7%

Table 4-1 Probability of out of bound (queuing delay>100 ms)

A question could be why the queue size of drop-tail is set as a very large value 1000 (in terms of packets) instead of a similar size to RED's maximum threshold? One reason lies in the fact that our research aims to observe RED's decay rate control and its potential improvements to the performance of voice traffic, so we need to know the original queue state distribution of these traffic scenarios without any control. This is achieved by setting a large enough buffer size. Then we can compare the queue state distribution under RED with the original one to observe RED's impact on traffic performance. Another reason is that, the shape of the queue state distribution under drop-tail does not change with its buffer size. In other words, if the buffer size of drop-tail is set to be a smaller value, for example 100, as shown in Figure 4-8, the queue state distribution keeps its shape but is much shorter than the distribution for a buffer size of 1000, and a pulse occurs at the end of the tail distribution (100 in this example). This "pulse" is approximately equal to the cumulative probability above 100 in the distribution for a buffer size of 1000. In these situations, the packet arrival process keeps the buffer of size 100 full, and excess packets are dropped. Since the shape of the queue state probability is almost the same, the corresponding decay rate of packet scale is the same (both 0.6514), while there is little difference between the decay rates of burst scale (0.9678 and 0.9760 respectively). So different buffer sizes for drop-tail do not greatly affect the decay rate. In addition, by setting a smaller buffer size for drop-tail, the delay and jitter can be controlled as well. However, as we will discuss in section 4.4, drop-tail

algorithm will result in a greater consecutive loss than RED algorithm. This is the reason why we study RED algorithm instead of a drop-tail algorithm with a small buffer size.

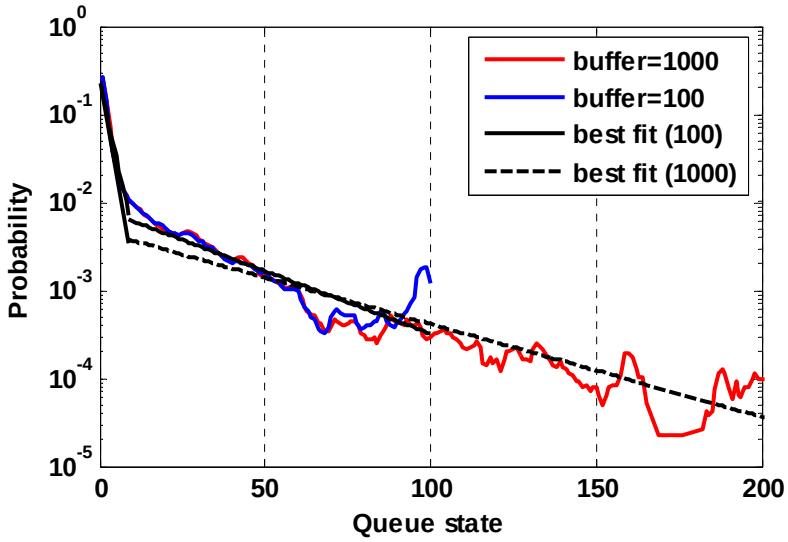

Figure 4-8 Queue state under drop-tail with different buffer size

So these curves for both RED and Drop-tail can clearly show the delay control by RED under different loads. In addition, these results under heavy load show that delay for voice traffic under RED can be maintained even if heavy congestion occurs, so that the QoS can degrade gracefully when congestion occurs.

4.3 Jitter

As introduced in section 2.2.2, jitter can be measured in several different ways. The focus of the study in this thesis is network queuing delay, and since the second method is a more network oriented observation of delay variability, we take the second method, and use the standard deviation of end-to-end delay to measure the jitter.

The standard deviation is a common measure of statistical dispersion. It is a measure of the average distance of the data values from their mean. If the data points are all close to the mean, then the standard deviation will be low. If many data points are very different from the mean, then the standard deviation is high. In telecommunication systems, this standard deviation of end-to-end delay means the

jitter. The bigger the standard deviation (jitter), the bigger the difference of the data points (packet end-to-end delay) from the mean (mean delay), which will degrade the QoS of real-time traffic.

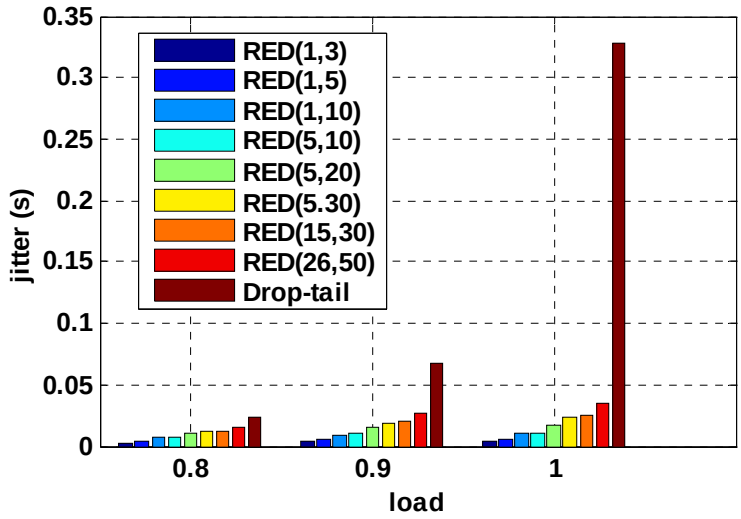

Figure 4-9 Jitter vs. load under different RED thresholds and drop-tail

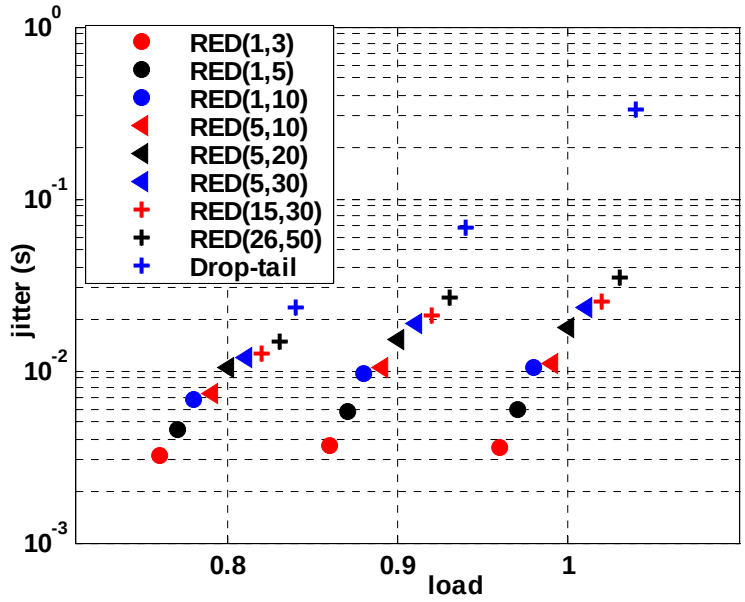

Figure 4-10 Jitter vs. load (log-linear scale)

Figure 4-9 plots the standard deviation of queuing delay for different loads and different RED thresholds. It is obvious that RED can control the jitter, even when the load is very high. The reason lies in the fact that RED's removal of the burst scale component decreases the probability of having a longer queue. So through RED's control of delay, meanwhile, jitter can also be controlled. Moreover, because jitter is

caused by variations of end-to-end delay (in essence queuing delay in our analysis), when RED maximum thresholds are set the same, the queue state distributions are quite similar, as shown in Figure 4-5, Figure 4-6 and Figure 4-7. Thus the corresponding jitter values are quite close as well, as shown in Figure 4-9 and Figure 4-10, the latter one draws the same graph in a log-linear scale to give a clearer view.

Next we consider another important performance metric for real-time traffic: loss. The discussion about loss is divided into two parts: effective loss and consecutive loss.

4.4 Loss

4.4.1 Effective Loss

When congestion occurs, the packet loss caused by RED resulting from a fully loaded queue must be higher than the loss caused by drop-tail, because RED detects congestion early before the queue is full while drop-tail will only drop packets after the queue is full. But due to the special characteristics of real-time traffic, if a packet arrives too late, which means the actual end-to-end delay is longer than the maximum acceptable delay, it is equivalent to the packet being lost. So we propose the concept of a packet being out of contract, in other words, effective loss, which we define as the sum of the packets lost within the network (dropped by the queue) and the packets arriving too late. This definition will give a fair comparison between RED and drop-tail in terms of loss. The definition of probability of out of contract is given as:

Pr {out of contract} = Pr {delay > α | packet not lost} * Pr {packet not lost} + Pr {packet lost}, where α is the maximum packet delay that a useful voice packet experiences.

This definition combines the delay with loss. We plot the complementary cumulative distribution function, weighted by the probability that a packet is not lost, and add it to the loss probability. This enables us to read off directly from the figure the effective loss for any value of end-to-end delay.

Figure 4-11 shows the probability of "Out of Contract Packets" of the voice model used above, which can show both the delay and loss performance in one figure. The X axis is delay, and the Y axis is the probability of out of contract packets, which includes the probability of both the too late packets and dropped packets within a queue. If we still take 100 ms as the bound of queuing delay, then the packets whose delays are longer than 100 ms are effectively lost. The actual probability of out of contract can be understood as follows: for the case when delay is never beyond 100 ms, the value at the Y axis at the end of each curve is the probability of being out of contract, for example the small ellipse in Figure 4-11; for the case when delay is longer than 100 ms, the value of the probability of being out of contract is the intersection between the curve and the line of delay=100 ms, as shown by the big ellipse in Figure 4-11.

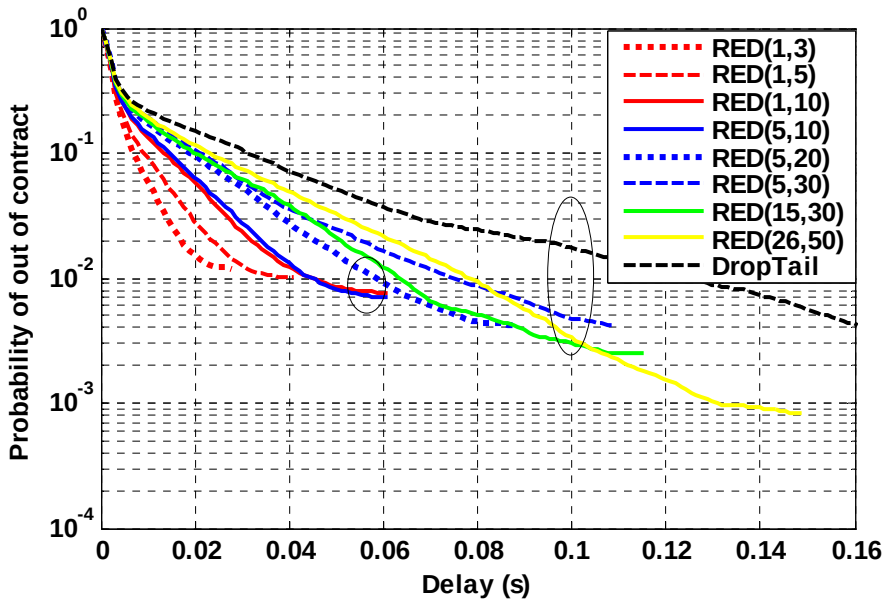

Figure 4-11 Probability of out of contract (load=0.8)

Clearly the drop-tail algorithm has the largest probability for out of contract packets whereas the RED mechanism indicates lower probability values. Thus the RED mechanism provides a much better performance when considering both delay and effective loss.

This performance improvement is especially obvious when congestion occurs under heavy load (e.g. load=0.9 or 1.0) as shown in Figure 4-12 and Figure 4-13, which gives the out of contract probability when the applied load is 0.9 and 1.0 respectively. If we take Figure 4-12 as an example, the out of contract probability for RED lies in

the interval of 0.019 and 0.035, which is acceptable to most voice traffic. Whereas the out of contract probability for drop-tail is as high as 0.15, which is on the boundary of what would be considered acceptable.

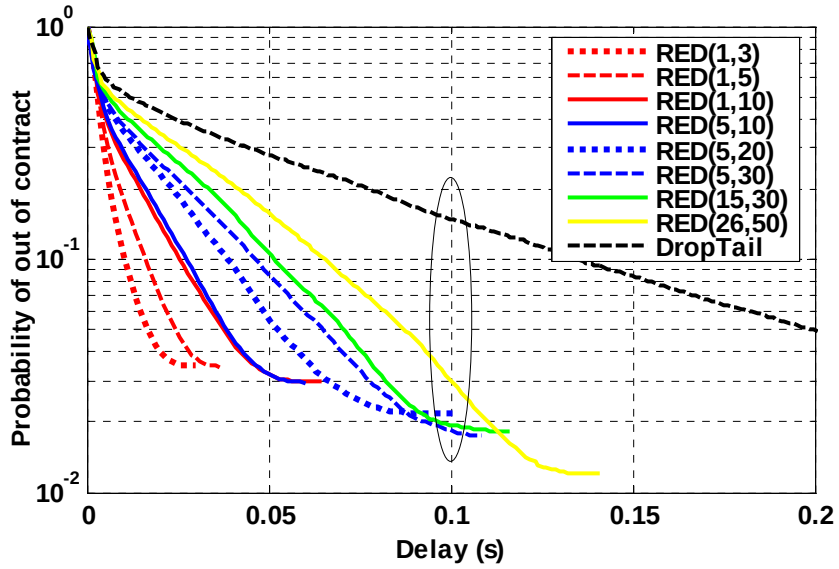

Figure 4-12 Probability of out of contract (load=0.9)

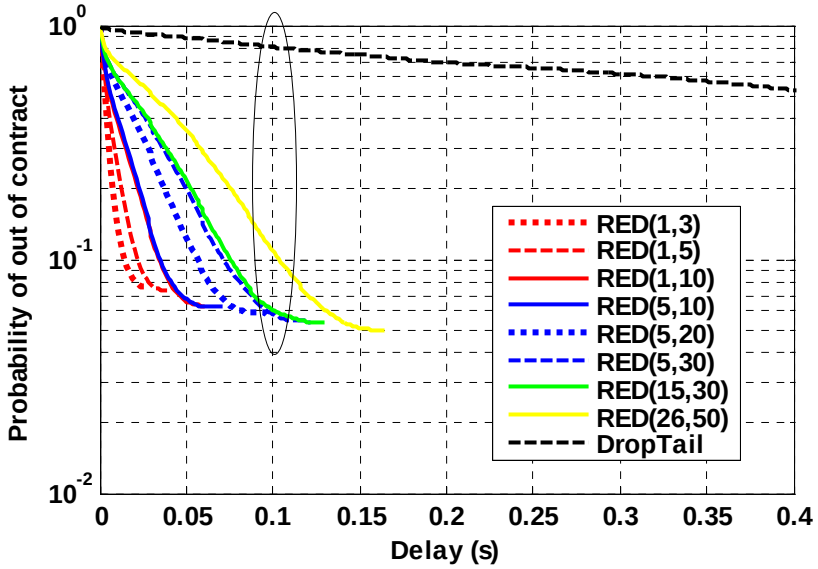

Figure 4-13 Probability of out of contract (load=1.0)

So a properly configured RED algorithm can greatly decrease the out of contract probability for VoIP traffic.

4.4.2 Consecutive Loss

As described in section 2.2.5, consecutive lost packets deteriorate the quality of

voice and video greatly and cause problems for loss recovery schemes. So besides the effective loss rate, the sequence of those lost packets is very important as well. In other words, when loss has to occur, a better algorithm is one that is able to drop the packets randomly. Packet id is the identifier of each packet, and a subsequent packet's id is set one more than the previous one automatically by the sender. So if the lost packet's id is not consecutive (lost packet id interval > 1), it means the dropped packets are not consecutive. By observing the lost packet id and the interval between lost packet ids, the contiguity of lost packets is clear. We know that an important feature of RED is that it drops packets randomly. Will this random drop bring some advantage when considering consecutive loss?

We take RED (15, 30) as an example, and compare the distribution of packet id intervals between lost packets (both dropped and late) with the equivalent distribution for the drop-tail algorithm (the buffer size of 1000 as before). The simulation topology and traffic scenarios are kept the same as before. Besides the effective packet loss rate, we measure the effective lost packet id interval distribution for both RED and drop-tail algorithms, taking 100ms as the delay bound (beyond which an arriving packet is too late to be of use).

Figure 4-14 and Figure 4-15 give the distribution of packet id interval between lost packets (both dropped and late) for drop-tail and RED respectively, when the load is 0.8. (NB: only part of the X axis is given for clarity). Figure 4-16 gives the corresponding distribution of lost packet id interval with a log-linear scale to give a clearer view. The smaller the probability of having a small interval, the better. Since the buffer size of the drop-tail algorithm is set as 1000, the packet loss with drop-tail mainly results from packets arriving too late to be of use. In contrast, the loss with the RED algorithm is mainly due to the random packet drops when the mean queue exceeds the minimum threshold. Figure 4-14 clearly indicates that almost all the lost packets under drop-tail are consecutive (lost packet id interval=1), while the range of RED's lost packet id interval is much wider than the drop-tail's and the probability of the lost packet id interval being 1 is much less (about 0.33), confirming the random, and less consecutive behaviours of RED's lost packet process compared to drop-tail. It is easier to recover from non-consecutive packet losses and hence this will not degrade the quality of traffic so much.

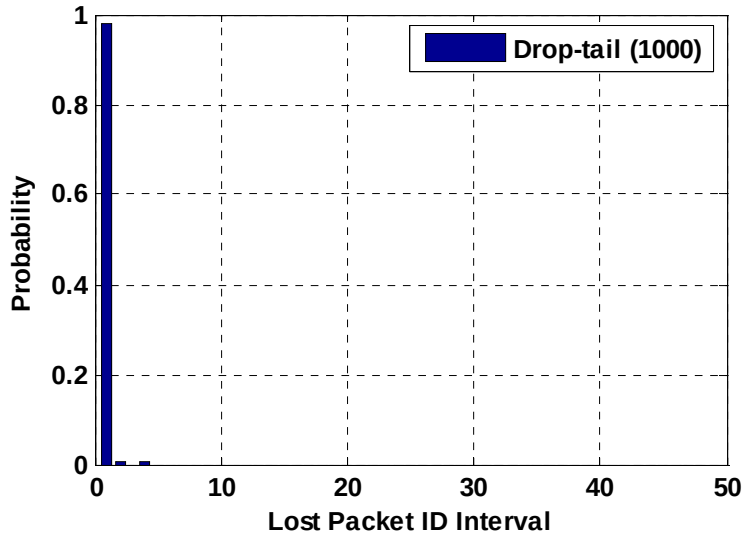

Figure 4-14 Lost packet ID interval distribution for Drop-tail

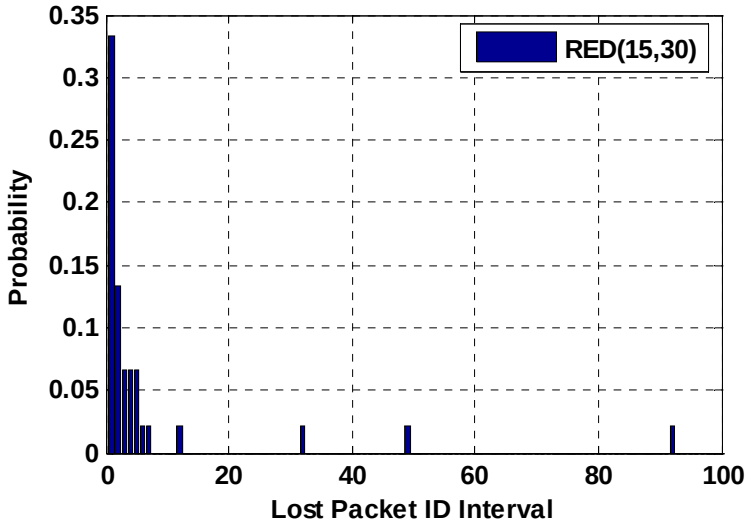

Figure 4-15 Lost packet ID interval distribution for RED (15, 30)

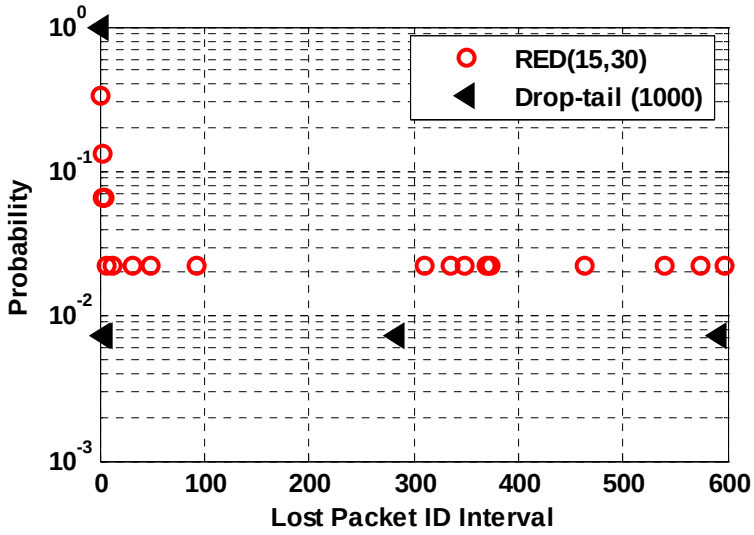

Figure 4-16 Lost packet ID interval distribution (log-linear scale)

In addition, Figure 4-17 and Figure 4-18 give the packet loss burst length distribution (in packets) for drop-tail and RED respectively. These figures indicate that, for RED, almost all the packet loss bursts have a length that is less than 5, while for drop-tail, the packet loss burst length is much longer and lies in the range of 5 to 30. As discussed in section 2.2.5, the decreased packet burst loss with RED is easier to recover from and helps to maintain the quality of traffic.

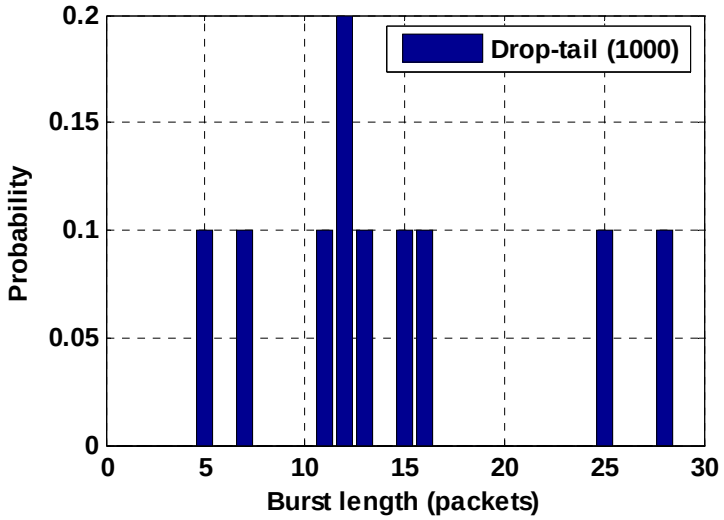

Figure 4-17 Packet loss burst length for drop-tail

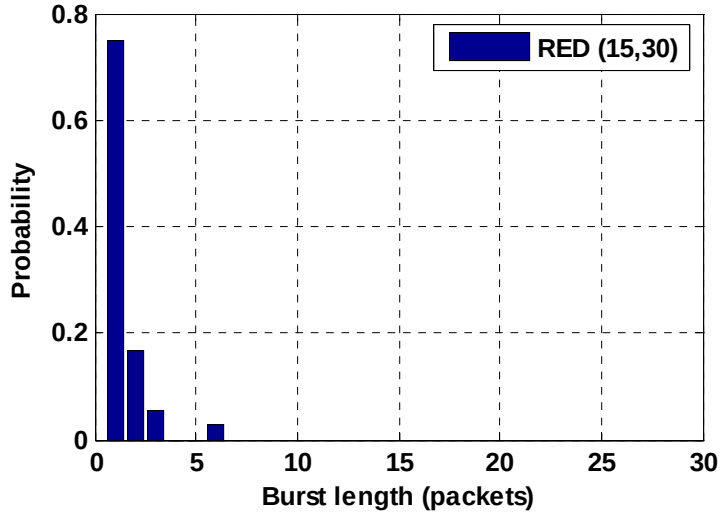

Figure 4-18 Packet loss burst length for RED

4.5 Congestion Control

It is clear from the performance evaluation results presented in the previous sections that the RED algorithm can help to maintain the performance in terms of delay, jitter and effective loss when congestion occurs. Figure 4-6 and Figure 4-7 show the complementary cumulative delay distribution under load 0.9 and 1.0, and they exhibit the queuing delay control by RED. Figure 4-9 shows that jitter can be maintained at a stable value by RED even if the network is heavily loaded. In Figure 4-12 and Figure 4-13, we can see the out of contract probability under RED is still acceptable if appropriate RED thresholds are set even when network load is very high such as 1.0.

Until now we have clearly seen that RED does control and maintain the QoS of inelastic VoIP traffic in terms of delay, jitter and loss. This effect exhibits a way to provide satisfactory QoS for VoIP traffic through properly configured RED algorithms. How to configure RED parameters to satisfy specific QoS requirements will be discussed in the next chapter.

4.6 Throughput

Throughput is used to describe the capability of a system to transfer data. Throughput is a metric for voice traffic as well, but is not so important as delay, jitter and loss.

Table 4-2 gives a full analysis of throughput, which shows the throughputs for different loads and different RED maximum thresholds, and compares these results with the Drop-tail algorithm. Additional tables of results can be found in Appendix E. Simulation results show that when load is not very heavy, e.g. 0.7, and 0.8, there is no difference between RED with different thresholds and Drop-tail in throughput, while when the network is heavily loaded, such as load is 0.9, 1.0 or 1.1, the throughput under RED is not as high as that of Drop-tail. Thus RED's performance improvements in terms of delay, jitter and out of contract loss are achieved with some sacrifice of throughput.

Max_{th}\load	0.7	0.8	0.9	1.0	1.1
Offered load	1.088	1.235	1.389	1.544	1.698
Drop-tail	1.085	1.232	1.383	1.528	1.543
50	1.085	1.231	1.370	1.459	1.507
45	1.084	1.231	1.367	1.458	1.504
40	1.084	1.230	1.365	1.455	1.503
35	1.083	1.230	1.361	1.451	1.500
30	1.083	1.228	1.358	1.445	1.498
25	1.083	1.228	1.354	1.442	1.497
20	1.082	1.227	1.354	1.440	1.493
15	1.082	1.226	1.351	1.433	1.488
10	1.082	1.226	1.348	1.432	1.485

Table 4-2 Throughput in Mbps for min_{th}=5 under different loads and max_{th}

4.7 Summary

In this chapter, a full evaluation of RED's QoS improvements for VoIP traffic is presented. Firstly, extensive simulation results show that the RED mechanism is able to regulate both packet- and burst-scale decay rates of queuing behaviour for inelastic voice traffic with real-time QoS requirements, especially to the component of burst scale, which dominates the larger queue size. This removal of the burst scale

component can help to decrease the probability of large queue sizes and hence reduce the mean queue size (mean queuing delay).

Secondly, since the relationship between decay rate and QoS is not direct, a detailed analysis about how RED's filtering of the burst scale decay rate can help to control the delay, jitter and effective loss is given. It indicates that through proper configuration of the RED algorithm, RED can control the queuing delay and jitter within acceptable ranges. The concept of effective loss is used to describe the loss rate for VoIP traffic. RED exhibits good control of effective loss. Also consecutive loss and burst loss, which is harmful to voice traffic, can be decreased by RED as well. In addition, simulations under loaded conditions show that these performance improvements by RED can work when the load is very high, thus providing a good solution to quality degradation of real-time traffic under congested network conditions. In conclusion, QoS improvements by RED are easy to apply at low service cost and with no changes to the existing network.

Chapter 5 RED Configuration Guidelines for Voice Traffic

In chapter 4, we have demonstrated through extensive simulations that RED does provide QoS improvements to voice traffic in terms of delay, jitter and effective loss. In general network scenarios we face the task of configuring those RED parameters per hop in order to achieve satisfactory end-to-end QoS. If we look back at the efforts to apply RED to TCP, (and the behaviour of RED in conjunction with TCP has been widely studied), one of the major limitations of RED for TCP is that its performance depends strongly on the settings of its parameters in the specific context of the network and traffic scenarios studied. Much work has addressed the difficulties of configuring RED parameters [Gev01] [Hol01], and how to configure RED parameters for TCP still remains an open issue. As for the configuration of RED for UDP traffic, no research has been done in this area. So in this chapter, RED configuration guidelines for voice traffic are proposed.

First an evaluation of the burstiness of voice traffic is provided, and this forms a foundation for the configuration guidelines. Secondly, we study the effects of max_{th} on the average queue size and the effective loss rate. Based on extensive simulation studies, we identify bounds on the AQS for voice traffic under different loads and thresholds.

5.1 Burstiness of Voice traffic

on/off times and the peak rate are the main parameters used to describe voice traffic modeled by on/off sources. In a traffic scenario, different values for these parameters will clearly result in different queuing behaviors. As introduced in chapter 4, the queuing behavior has a direct impact on the delay, jitter and effective loss experienced by voice traffic, and that means queuing behaviours are essential to our study of QoS. So we will study the essential queuing behaviors caused by voice models with a range of different parameters in this section.

A traffic process A is said to be more bursty than a traffic process B if a single server queuing system yields a longer system size tail distribution when the process A is used to drive the queuing system [Gao00]. Hence we will evaluate the burstiness of a range of voice models by measuring the queuing behaviors produced from our simulation results.

5.1.1 Different Peak Rate

In this section, the burstiness of voice traffic caused by different peak rates is studied, i.e. voice models using the same on/off times but different codecs and hence different peak rates. Voice traffic with different peak rates is applied to a simple FIFO queue. The queue size is set big enough to hold each arriving packet and to keep its original queue state distributions. Once the queue state distribution of each individual traffic model is known, then the burstiness of each type of traffic is evident: the longer the tail of the queue state probability distribution, the burstier the traffic.

	Voice codec	Bit rate (Kbps)	Packet size (bytes)	IP bandwidth (Kbps)	Ton (s)	Toff (s)
Scenario1 (s1)	G.711	64	214	85.6	0.352	0.65
Scenario2 (s2)	G.729	8	74	29.6	0.352	0.65
Scenario3 (s3)	G.711	64	214	85.6	1.004	1.587
Scenario4 (s4)	G.729	8	74	29.6	1.004	1.587
Scenario5 (s5)	G.723.1	6.3	84	17.6	0.352	0.65
Scenario6 (s6)	G.711	64	214	85.6	1.004	0.001

Table 5-1 voice scenarios for different peak rate and different on/off times

Table 5-1 summarizes the details of the simulation scenarios used in this section. Firstly the on/off times are set as 0.352s/ 0.65s respectively, and G.711 (with the higher sending rate of 85.6Kbps) and G.729 (with the lower sending rate of 29.6 Kbps) codecs are used to send voice packets to the same single FIFO queue. Multiple identical on/off traffics sources which start at randomly selected times are used to generate different aggregate load levels applied to the queue, e.g. 0.7, 0.8, and 0.9.

Secondly, the voice model with on/off times of 1.004s/1.587s is used, and sending rates as before. The queue state distribution for each scenario (s1 vs s2, s3 vs s4) is compared and analyzed.

Figure 5-1 gives the corresponding queue state probability for scenario 1 and scenario 2 under different loads. Obviously, scenario 1 in red has a much longer and flatter tail for its queue state distribution, implying that scenario 1 traffic is much burstier than scenario 2. Similarly, the queue state probability in Figure 5-2 indicates that scenario 3 traffic (with the higher sending rate) is burstier than scenario 4.

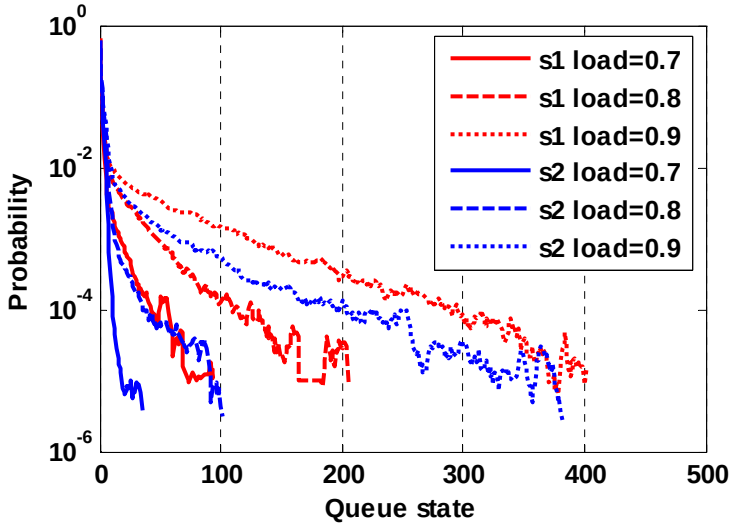

Figure 5-1 Burstiness of scenario 1 and 2 under load 0.7, 0.8 and 0.9

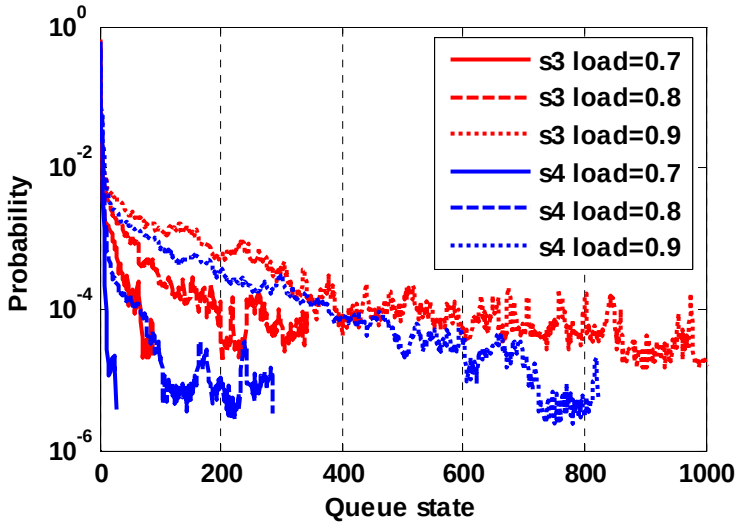

Figure 5-2 Burstiness of scenario 3 and 4 under load 0.7, 0.8 and 0.9

The reason lies in the mixture of multiple on/off sources. In this comparison, there are different peak rates between scenario 1 and scenario 2, and so different numbers of sources are multiplexed to form the same overall mean load. For on/off sources, each source produces one burst every cycle of (t_{on}+t_{off}) seconds. Now, bursts from multiple traffic sources may not arrive at exactly the same instant during each cycle time ((t_{on}+t_{off}) seconds). During each cycle, packets from one source may arrive when other sources are in their off state. In this situation, the total arriving rate multiplexed by several G.729 sources may be temporarily less than the arriving rate caused by only one G.711 source, and so the queue fluctuation can be less although the mean load is the same (2.89 G.729 sources are equivalent to one G.711 source, in terms of mean load). As for s3 and s4 in Figure 5-2, the same underlying behaviour results in a similar outcome. So the first conclusion is that when the on/off times and mean load are the same, voice traffic with a higher peak rate results in burstier aggregate traffic (longer queue state distribution and flatter burst scale queuing component).

5.1.2 Different On/off Times

Another factor to describe voice traffic behaviour are the on/off times. In this section, we analyze the burstiness of on/off sources caused by different on/off times. The on/off times are set as the widely accepted values of 0.352s/0.65s and 1.004s/1.587s respectively, and the same sending rate is applied. The corresponding queue state probability can be found in Figure 5-3 and Figure 5-4, the former figure is with the higher rate of 85.6Kbps, the latter figure is with the rate of 29.6Kbps. These figures show that a voice model with the on/off times of 1.004s/1.587s (in black) is burstier than the model with the on/off times of 0.35s/0.65s (in red). Because a source produces a burst during its on time, for the traffic with an on/off time of 1.004s/1.587s, G.711 produces 50 packets (10700 bytes) in an average size burst (and G.729 produces 50 packets = 3700 bytes). For the traffic with an on/off time of 0.352s/0.65s, G.711 produces 17.6 packets (3766 bytes) in an average size burst (G.729 produces 17.6 packets =1302 bytes).Clearly more bits or packets are produced for the longer on/off time cycle (1.004s/1.587s), so the queue needs more space (buffer) to hold or absorb the burst brought by this longer burst.

We can also conclude that when the sending rate and mean load are the same, voice

traffic with a longer on time is burstier than voice traffic with a shorter on time.

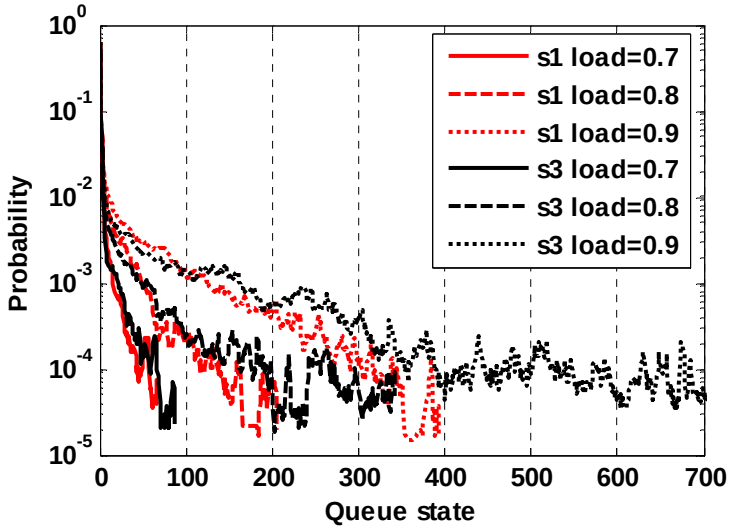

Figure 5-3 Burstiness of scenario 1 and 3 under load 0.7, 0.8 and 0.9

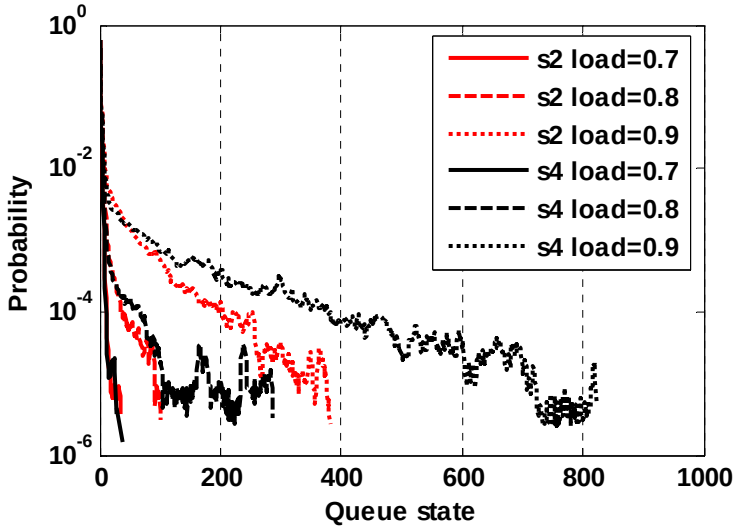

Figure 5-4 Burstiness of scenario 2 and 4 under load 0.7, 0.8 and 0.9

Since a longer on time will result in a burstier traffic, will the traffic become more bursty if the on time is increased and the off time is decreased to a small value? Under this assumption, the queuing behaviours are quite similar to CBR (Constant Bit Rate) traffic, and CBR traffic does not exhibit a burst scale component, in other words, CBR is not bursty. The burst scale occurs when the transient arriving rate is bigger than service rate. Now the arriving rate from CBR traffic is constant, and at mean loads below 100% will always therefore be lower than the service rate. So no burst scale queuing occurs, and there exists only a short queue state caused by packet scale queuing. Figure 5-5 gives a comparison between scenario 3 (in red) and scenario 6

(in blue) whose on time is 1004 times larger than the off time. Scenario 6 shows very short queue state due to the lack of burst scale.

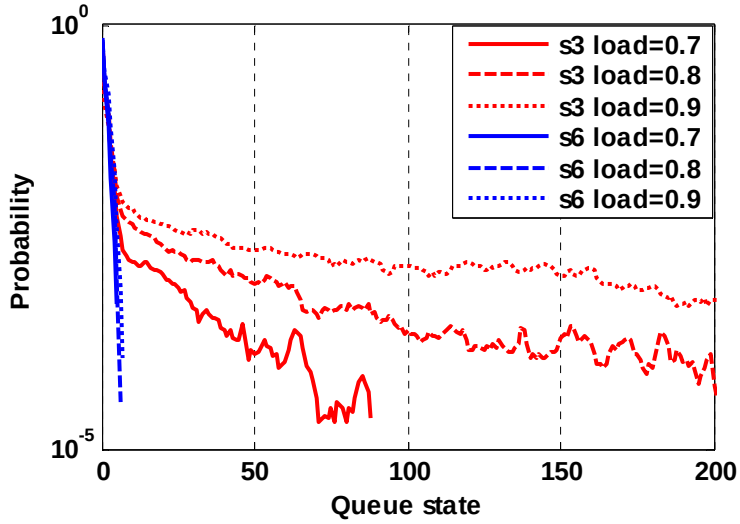

Figure 5-5 Burstiness of scenario 3 and 6 under load 0.7, 0.8 and 0.9

5.1.3 Summary

Buffering in the network is designed to absorb data bursts and to transmit them during the following bursts of silence. This is essential to permit the transmission of bursty data. In routers sufficient queue capacity is provided to absorb the bursts, and the mean queue size should normally be small to provide lower queuing delay. In general, a rule of thumb is that queue limits should reflect the size of bursts we need to absorb. This is the reason why we study the burstiness of voice traffic.

Through extensive simulations with on/off traffic, we can draw the conclusion that for voice traffic modeled by multiple identical on/off sources, higher sending rates and longer on times will result in a longer queue state tail distribution and a flatter burst scale queuing component. This results in a longer average queue size. Our simulation results also accord with the general trends mentioned in [Mic97] as well. That paper indicated that when the number of connections or the peak rate increases, which means the load applied to the queue increases, the interception of the two components moves leftward, reducing the packet-scale queuing, and the burst-scale becomes flatter (bigger). When the mean load applied to the queue is kept constant but the burst length is increased, the knee point is more or less unchanged but the

burst component becomes flatter (bigger).

Since burstier traffic will result in a longer queue state tail and a flatter (bigger) burst scale component, this results in a larger average queue size within the queue (introduced in section 3.2.3). The burstier a voice traffic source is, the longer the AQS of the traffic, and so on/off traffic with a realistic highest peak rate and longest on time will result in a maximum or worst case AQS. This worst case AQS can be used to estimate the worst mean queuing delay the voice traffic will experience. We will discuss this application in detail in the rest of this chapter.

5.2 Estimated AQS Bound

Based on our previous observations reported in chapter 4, we find that RED parameters can be configured to regulate and control the burst scale decay rate of voice traffic, especially the RED maximum threshold, (the bigger the RED maximum threshold, the bigger (flatter) the decay rate for the burst scale). Now a larger burst scale results in a larger queue size (thus longer queuing delay), so by controlling the burst scale, the AQS and hence the average queuing delay is controlled as well.

To give a clearer analysis of this decay rate control, [Pit06] provides a derivation of Equation 5-1 using excess rate queuing theory. This is an analysis of the M/M/1/RED queue model and gives explicit formulas relating mean queue size, load and the RED parameters: $\min_{th}$, $\max_{th}$, p_{max} for Poisson traffic. The mean queue size, $\bar{k}$, can be expressed as a quadratic, whose solution is given by:

$$A = \rho \cdot \varphi \cdot (1 + \rho \cdot \varphi)$$

$$B = 1 - \rho \cdot \varphi \cdot \min_{th} - \rho^2 + \rho \cdot \varphi - 2 \cdot \rho^2 \cdot \varphi^2 \cdot \min_{th}$$

$$\bar{k} = \frac{-B + \sqrt{B^2 - 4 \cdot A \cdot C}}{2 \cdot A} \qquad C = \rho \cdot (\rho \cdot \varphi^2 \cdot \min_{th}^2 - 1 - \rho - \varphi \cdot \min_{th}) \qquad \text{Equation 5-1}$$

$$\varphi = \frac{p_{\max}}{\max_{th} - \min_{th}}$$

Where ρ is the load offered to the queue (i.e. before any drop process). Based on Equation 5-1, Table 5-2 gives the offered load (denoted $\rho_{\max}$) for which the mean

queue size, $\bar{k}$, is equal to the maximum threshold, max_{th}. Figure 5-6 draws the relationship between max_{th} and the average queue size for M/M/1/RED model under different loads, where min_{th}=1 and p_{max}=0.1. Since the mean queue size $\bar{k}$ cannot exceed max_{th}, we can calculate not only the corresponding load offered to the RED queue ρ_{max} at which $\bar{k}$=max_{th} but also the resulting maximum decay rate dr_{max} under this ρ_{max}. At loads above ρ_{max}, the AQS is limited to max_{th} and the decay rate is limited to dr_{max}. These limits only depend on the RED parameters, and are not related to the offered load. So if we assume Poisson traffic entering a RED queue, and the RED parameters are known, the maximum for the mean queue size (thus the corresponding maximum for the mean queuing delay), the maximum load admitted to the RED queue, and the maximum decay rate are known. Moreover, if the actual Poisson traffic load offered to the RED queue is known (e.g. via the admission control), the estimated mean queue size and, hence, mean delay can be calculated as well.

Max_{th}	2	4	5	10	15	20	25
ρ_{max}	0.7091	0.8466	0.8809	0.9585	0.9875	1.0027	1.0120
Max_{th}	30	35	40	45	50	55	
ρ_{max}	1.0184	1.0229	1.0264	1.0291	1.0313	1.0330	

Table 5-2 Max_{th} and the corresponding ρ_{max}

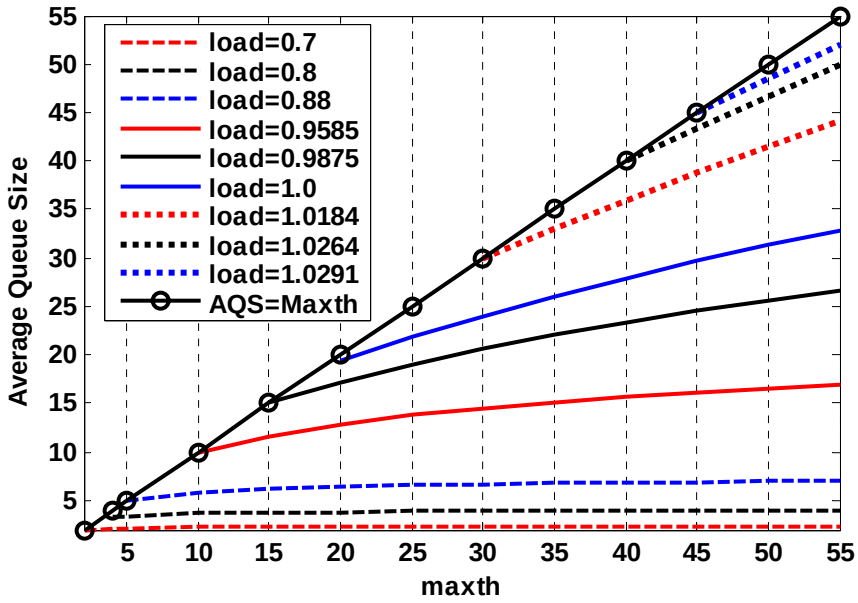

Figure 5-6 AQS vs. Max_{th} for M/M/1/RED model

The next step is to consider the on/off traffic of voice models. Is it possible to provide a similar AQS bound for on/off traffic? The following section will address this question.

It is assumed that in general UDP sources do not implement a congestion avoidance mechanism to reduce their transmission rate due to packet loss. Depending on the amount of UDP traffic arriving at the RED queue and the value of p_{max}, RED should keep the average queue size below max_{th}. But when congestion occurs, in other words, the network is overloaded, due to the arrival rate for traffic being higher than the bottleneck link capacity, RED will keep the average queue size at a level of around max_{th}.

Simulations were run to verify this behaviour. Scenario 3 summarized in Table 5-1 has the realistic highest peak rate and longest on time, and results from section 5.1 have verified that scenario 3 is the most bursty (with the largest decay rate and thus the longest AQS). So we will take scenario 3 as the traffic source here. The mean arrival rate is set as 110% of the bottleneck link capacity. Different RED minimum and maximum thresholds are used, and p_{max} and w_q are set as 0.1 and 0.002 individually as recommended in [Flo97], which are the most often used values for p_{max} and w_q as well. The traffic offered to the bottleneck link is larger than the link

capacity, and RED will drop the extra 10% traffic to keep the bandwidth almost fully utilized at ρ_{max}, and the average queue size is estimated to be around the maximum threshold. Figure 5-7 gives the corresponding simulation results and we can find: firstly, the AQS is around the maximum threshold as estimated; Secondly, the AQS increases with the maximum threshold; thirdly, there is little difference for the AQS when the maximum thresholds are set the same while minimum thresholds are different, in other words, similar to the results obtained in chapter 4. Based on both theoretical analysis and simulations results, the linear function y=x can be used as an estimated bound for the actual AQS of a RED queue when the load is around 1.1 whatever the RED thresholds are.

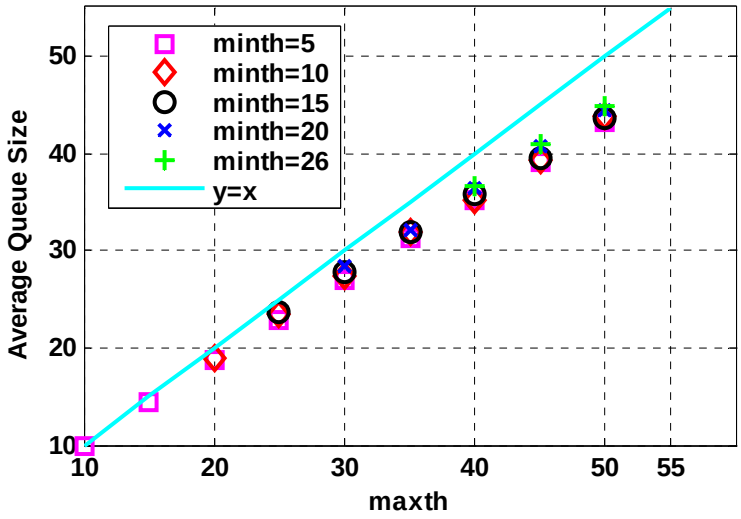

Figure 5-7 AQS Vs. max_{th} (load=1.1)

Moreover, as shown in Figure 5-7, the AQS has an approximately linear relationship with the maximum thresholds. So, using linear regression (see Appendix C), we obtain the function which describes this relationship.

Based on this observation, we can explore the relationship between maximum threshold and AQS at lower loads, to see whether a linear relationship is also a good approximation.

Simulations were run to observe the AQS under different RED thresholds and different loads, with the same voice traffic of scenario 3. To give an accurate estimate of AQS, each simulation was repeated 10 times with different random seeds. The simulation results are shown in Figure 5-8, and we can see that, similar to Figure

5-7, whatever the load, the AQS under different RED thresholds always approximately forms a straight line. Each different load has a different slope, and the corresponding best fit lines (the cyan lines) are obtained using linear regression (see Appendix C). As we have mentioned, scenario 3 is the most bursty on/off traffic for modeling voice, thus it has the longest queue state tail distribution and biggest burst scale component, so scenario 3 exhibits the largest AQS. Thus these best-fit lines from scenario 3 could provide the worst case estimated AQS bound for all the other less bursty voice scenarios.

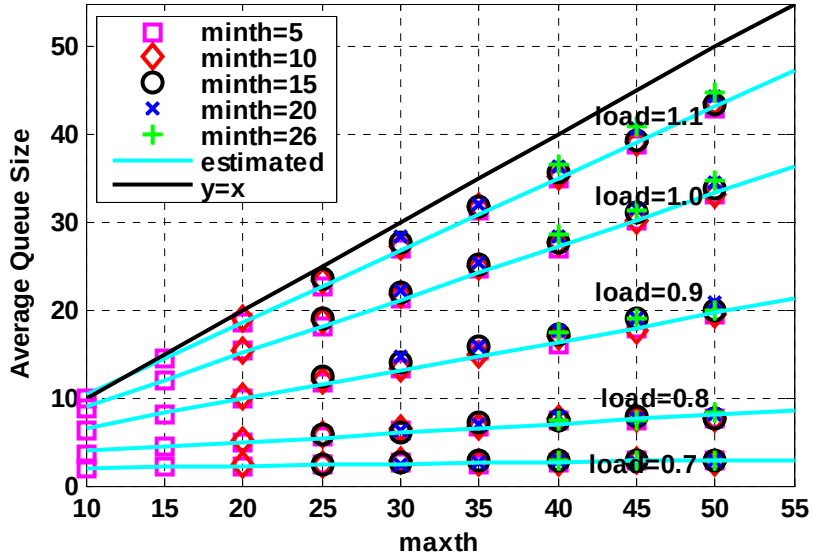

Figure 5-8 AQS Vs. max_{th} under different load and different min_{th} (scenario 3)

To verify the assumption above, the other three typical voice models, scenario 1, scenario 4 and scenario 5 summarized in Table 5-1, are applied to the same link. Figure 5-9, Figure 5-10 and Figure 5-11 give the corresponding results for AQS against maximum threshold under different load for scenarios 1, 4 and 5. In each case, the best-fit lines from scenario 3 are shown as the "estimated" worst case bound on AQS. NB: for a load of 1.1, the y=x bound is used instead of the best-fit line shown in Figure 5-8.

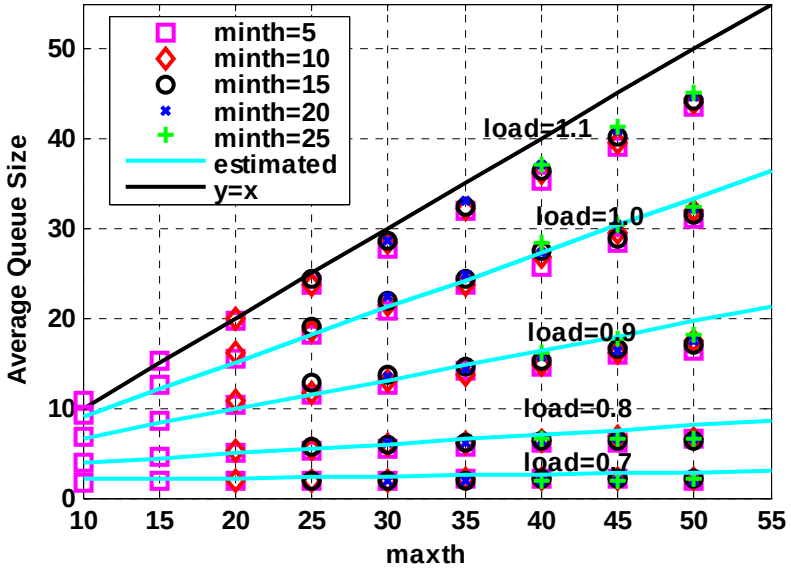

Figure 5-9 AQS Vs. max_{th} under different load and different min_{th} (scenario 1)

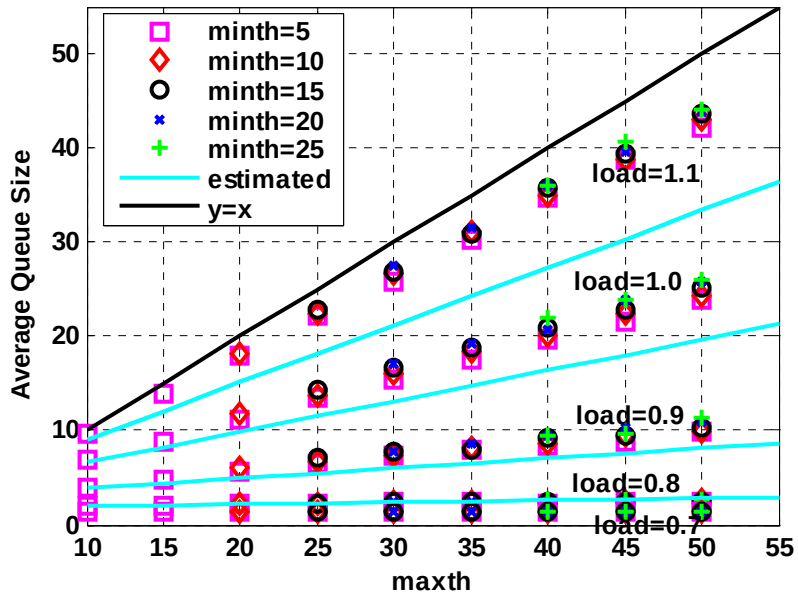

Figure 5-10 AQS Vs. max_{th} under different load and different min_{th} (scenario 4)

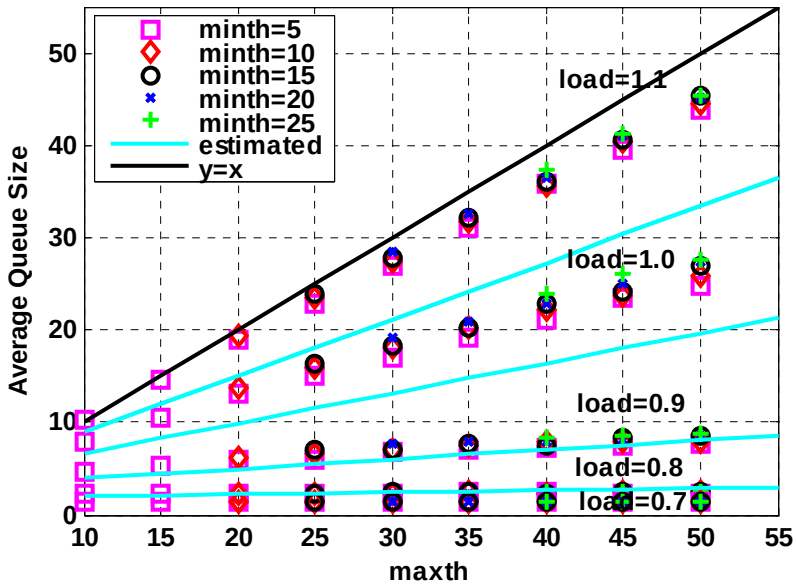

Figure 5-11 AQS Vs. max_{th} under different load and different min_{th} (scenario 5)

As shown in Figure 5-9, Figure 5-10 and Figure 5-11, the estimated best-fit lines do work as a worst-case AQS bound for scenario 1, scenario 4 and scenario 5 under different loads.

Now, scenarios 1 and 5 have the same on/off times, but scenario 1 has a higher sending rate than scenario 5, and hence is burstier than scenario 5. This is why the actual AQS of scenario 1 is closer to the estimated bound than scenario 5. Scenario 3 and scenario 1 have the same peak rate while scenario 3 has a longer on time than scenario 1, which means scenario 3 is burstier than scenario 1. Thus scenario 3 has a higher AQS than scenario 1.

Once the bound for the AQS is known, the corresponding average queuing delay can be calculated based on:

delay = AQS * Packet_Size * 8 / Service rate Equation 5-2

Take scenario 3 as an example, if the applied load is around 0.9, and RED maximum thresholds are known to be set as 50 (minimum threshold is not important), then the corresponding AQS is 20 (see from Figure 5-8). So with a packet size of 214 bytes and a link service rate of 1.544Mbps, the queuing delay bound is about 22.2 ms. If this queuing delay does not satisfy the service providers, two solutions are possible: to reduce the applied load over the link (by admission control) or to decrease the

maximum threshold. If the applied load is reduced to 0.8 and other parameters are kept unchanged, the AQS decreases to 8, and hence the corresponding queuing delay is 8.8 ms; if the applied load is unchanged and the maximum threshold is decreased to 35, the AQS decreases to 15, and hence the corresponding queuing delay is 16.6 ms. These calculations of queuing delay are based on G.711, as for other codecs such as the G.729 and G.723.1, the queuing delay is much shorter, because these two codecs have smaller packet sizes. For example, if the applied load is 0.9 and the maximum threshold is set as 50, the corresponding queuing delay for G.729 and G.723.1 is 7.7 ms and 8.7 ms respectively.

In addition, when the maximum threshold is larger than 50, which is beyond the scale of the X axis in Figure 5-8, the AQS can be calculated from the corresponding derived functions in Table 5-3.

load	Loss rate	Function of AQS
0.7	<0.002	Y=0.0212X+1.7984
0.8	<0.01	Y=0.1042X+2.866
0.9	<0.03	Y=0.3254X+3.3623
1.0	<0.07	Y=0.6096X+2.8875
1.1	Around 0.1	Y=X

Table 5-3 Configuration guideline for voice traffic under RED

Here, Y is the estimated bound (in terms of packet), and X is the maximum threshold of RED algorithm.

Queuing delay is not the only performance metric which affects the QoS of voice traffic. In the following section, network loss is analyzed. There is a direct relationship between the queue and network loss, because the network loss mainly occurs when the average queue size is beyond the preset thresholds (for RED) or the queue size is beyond the queue limit (for drop-tail). The larger the average queue size, the higher the probability of being beyond the preset threshold (or queue limit), and hence the higher the network loss rate.

Here we only study the network loss instead of the effective loss which includes both the packets lost in the network and the packets arriving too late to be of use. This is because the too late loss varies under different circumstances. For example, first, the too late loss is determined by the target bound for end-to-end delay, which may vary with service requirements at different levels. Secondly, different codecs have different packet sizes, as shown in Equation 5-2, which will result in different queuing delays. Similarly, the service rate varies under different circumstances, which results in different queuing times as well. So these uncertain factors make it impossible to give a general estimate for too late loss.

In the following section, firstly, the network loss of the most bursty voice traffic: scenario 3 is observed. Different numbers of scenario 3 traffic are multiplexed to form different traffic loads on a single bottle-neck with a RED queue. We fix the random number generator seed for each run to obtain comparable results when we compare different scenarios.

Figure 5-12 gives the network loss against different maximum thresholds under different network loads for scenario 3 traffic (to give a clear view, the Y axis is drawn in a logarithm scale). As shown in Figure 5-12, first, the higher the load, the higher the loss rate. The reason is obvious, as described in section 5.1, because the higher the load, the burstier the traffic, and hence the longer the AQS. When the thresholds are set as the same values, the traffic with a longer AQS will have to suffer a higher loss due to the fact that it would need more buffer space to hold its longer queue state tail. Secondly, large maximum thresholds result in a reduced loss rate. Although the effects of maximum thresholds on the loss rate are not as obvious as their effects on AQS, especially when the load is high, it does mean that changes of RED thresholds can help adjust the resulting AQS without causing large variation in the loss rate. So when the applied load is the same, the highest loss rate occurs when the maximum thresholds are set as very small values (such as 10), and the loss rate under this circumstance could be a rough estimate for the worst loss rate. The actual loss rate is no larger than this bound, and hence provides a worst-case estimate for the loss rate that voice traffic suffers.

Moreover, simulations based on the other less bursty voice traffic are run to give the

corresponding loss rate analysis. It is assumed that the less bursty traffic should exhibit lower loss rate compared with the scenario 3 traffic in Figure 5-12 when the applied load is the same. Figure 5-13, Figure 5-14, and Figure 5-15 give the loss rate for scenario 1, scenario 4 and scenario 5 respectively.

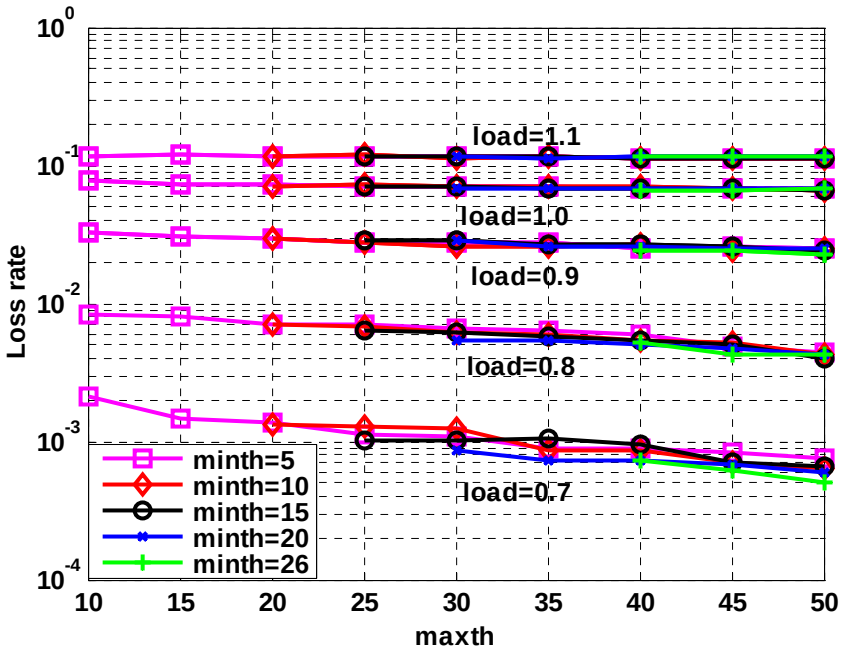

Figure 5-12 Network loss rate vs. max_{th} (scenario 3)

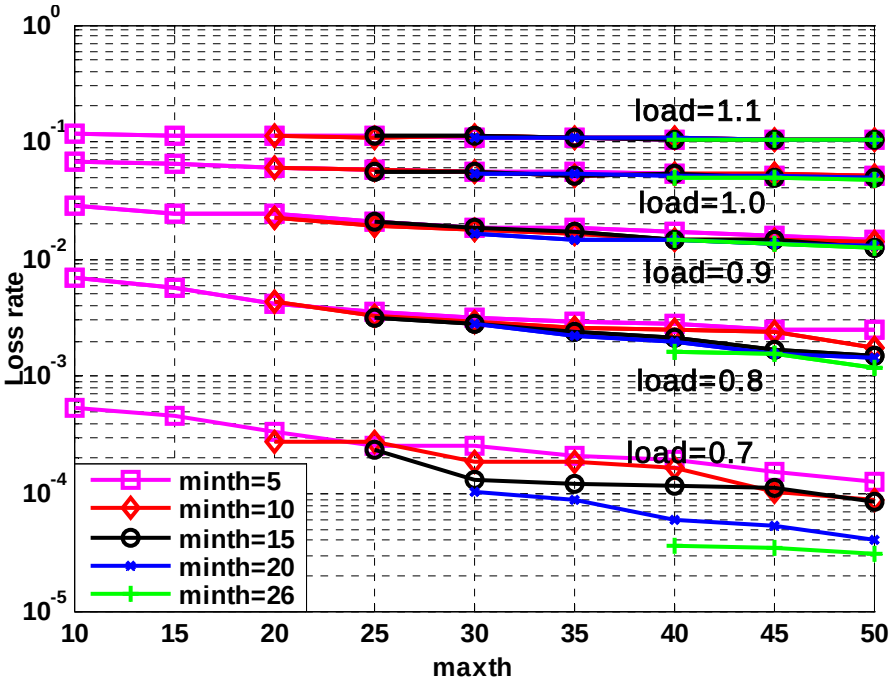

Figure 5-13 Network loss rate vs max_{th} (scenario 1)

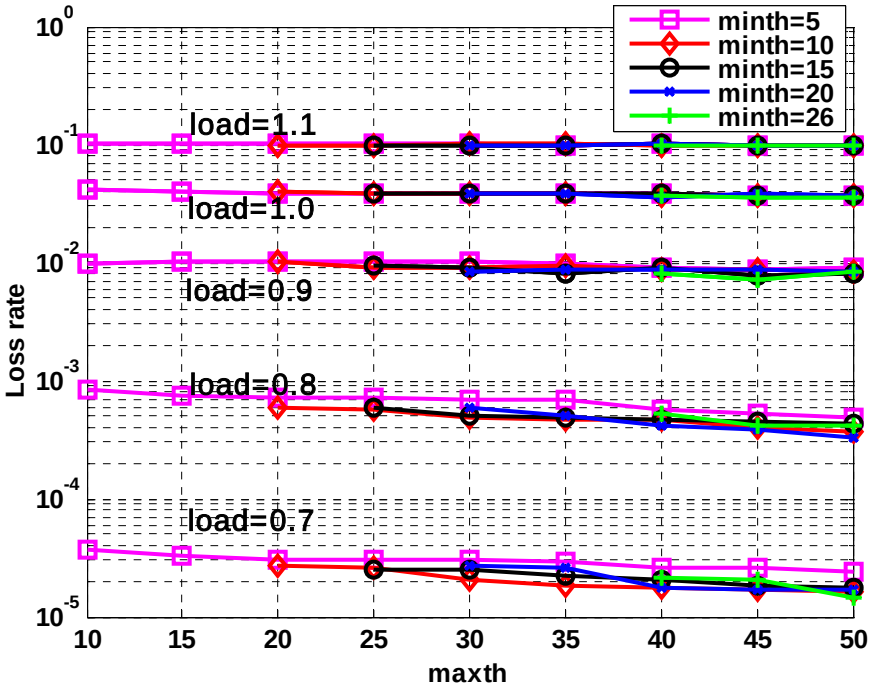

Figure 5-14 Network loss rate vs max_{th} (scenario 4)

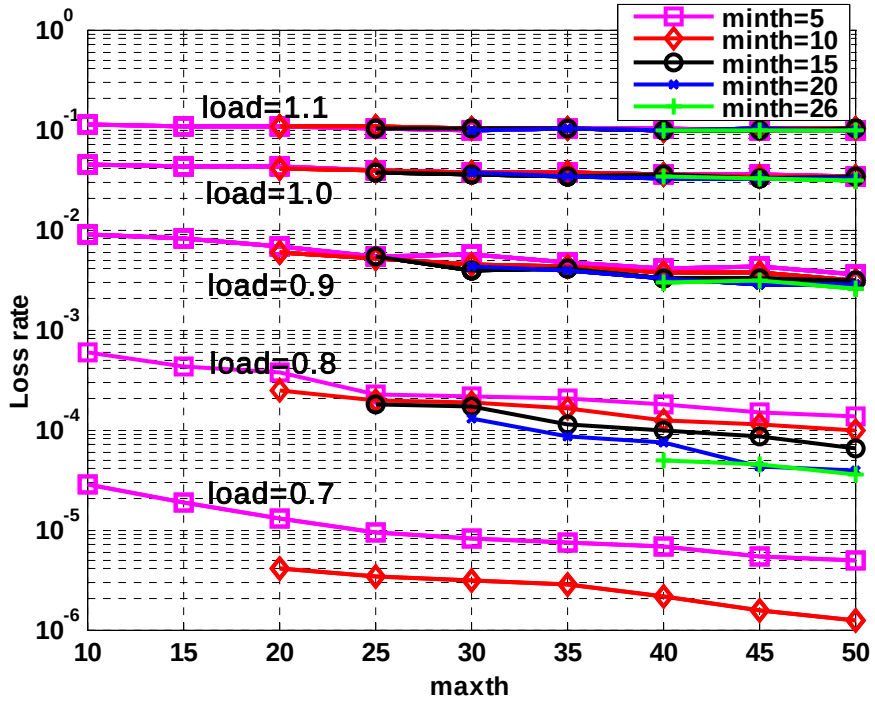

Figure 5-15 Network loss rate vs max_{th} (scenario 5) (when load=0.7, min=15, 20 and 26, loss rate=0)

If we compare the loss rate in Figure 5-12, Figure 5-13, Figure 5-14 and Figure 5-15, it can be found that for loss rate: s3 > s1, s4 and s5, s1 > s5, and s4> s5. If we observe the corresponding burstiness analysis in section 5.1, we can find that: for burstiness: s3 >s1, s4 and s5, s1 > s5, and s4 > s5. So the assumed relationship between

burstiness and loss rate is verified, which means when the RED thresholds and applied load are the same, burstier voice traffic has a higher loss rate. As we mentioned above, the reason is that burstier traffic can result in a longer queue state distribution and larger (flatter) burst scale component and hence a longer AQS, which needs more buffer space to hold its longer queue state tail. If the maximum threshold is set as the same length, obviously the traffic which can produce a longer AQS (in tail-drop) has a higher probability of reaching the maximum threshold and hence a higher loss.

So we can estimate a worst case loss rate for voice traffic under each load, which is based on the worst loss rate of the most bursty traffic: scenario 3 traffic under each load. For a given load, the worst (highest) loss rate occurs when the maximum thresholds are set very small such as 10, and the worst loss rate of the most bursty traffic could be a rough estimate for the other less bursty traffic. In other words, for all the on/off voice models, their loss rates will not surpass this estimated worst case. The estimate of this worst loss rate is less accurate than the estimate for AQS, but it is acceptable because voice traffic is more tolerable to loss than to delay. The worst loss rate for each load is also summarized in Table 5-3. Because when the load is 0.7, both the AQS and loss rate are very small, the bounds for loads less than 0.7 are not given. The bounds for load 0.7 can be applied as a bound for loads less than 0.7.

Figure 5-16 and Table 5-3 summarizes the estimated loss rate, AQS and max_{th} in the same figure, which can be used as a guideline for the configuration of RED thresholds to voice traffic. For example, if a network is known to be configured with RED (10, 20) and designed to work up to a load of 0.8 (which can be achieved through the application of admission control), then the estimated AQS is 5 (queuing delay of 5.5 ms for G.711, queuing delay of 1.9 ms for G.729, queuing delay of 2.2 ms for G.723.1 over a T1 link) and the loss rate is less than 0.01. So we can predict the queuing delay of 1.9 ms for G.729 and loss rate of no more than 1% per hop, which can provide a satisfactory performance for voice traffic.

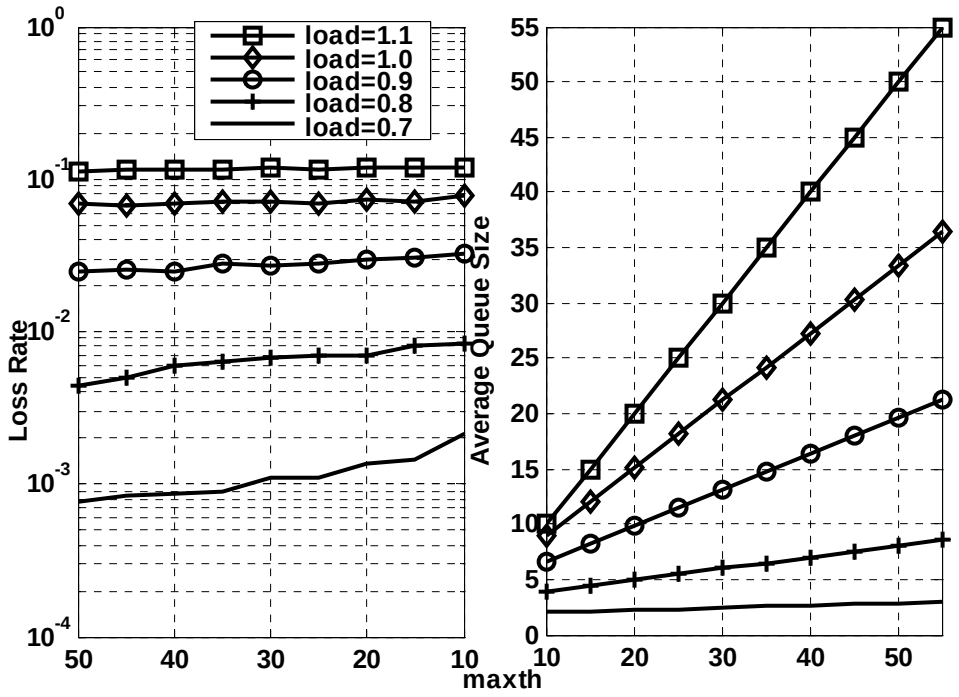

Figure 5-16 RED Configuration guideline for voice traffic

Supplements: when maximum thresholds are set larger than 50, the estimated AQS can be calculated from the corresponding functions shown in Table 5-3. For example, if a RED queue works up to a load of 0.8 and its thresholds are set as (30, 60), the estimated AQS bound is 9 (queuing delay 3.5 ms for G.729), and the corresponding loss rate is no more than 0.01.

5.3 Discussion about the Applications of the Guideline

The AQS increases with the increasing max_{th}, however, this increase does not continue indefinitely. As shown in Figure 5-1 and Figure 5-3, when the load is the same, different scenarios will result in different queue length distribution since different traffic types have different burstiness. The burst from a burstier traffic occupies a longer buffer. But the traffic will not occupy all the buffer space available when the load is not high, e.g. for loads of 0.7 and 0.8. In this case, the mean occupied queue length (i.e. the AQS) will keep constant once the maximum threshold exceeds a particular value. This particular value depends on the burstiness of the traffic. The burstier a traffic type, the larger this value. So the RED's max_{th}

should be set within this corresponding bound to provide a more accurate estimate for the AQS when using the linear functions given in Table 5-3. Once the value of RED max_{th} is set beyond this bound, the increasing max_{th} will not affect the queue length distribution any longer.

The estimated queuing delay is related to the packet size. The queuing delay based on the maximum packet size can provide a worst case queuing delay for the voice traffic. Moreover, the application of Weighted Fair Queuing (WFQ) can provide a more accurate prediction for the queuing delay. WFQ is a scheduling algorithm used to separate different traffic flows and allocate a proportion of the link bandwidth to these different traffic types. The mixture of different types of voice traffic with different packet sizes and different burstiness can overestimate or underestimate the actual queuing delay. If traffic types are segregated using WFQ then the delay estimate is based on each traffic type's own characteristics, such as its packet size and burstiness. This separation of traffic by WFQ has been reported in [Ali04] [Yil01] [Kar00] and it can be applied with low cost and complexity.

This configuration guideline is derived from the simulations run over a T1 link. Can the guidelines be applied to links with different bit-rates? How can we recommend applying the guidelines under these circumstances? The following section will discuss this question.

This question occurs due to the fact that when we scale up link bit-rate and load (to keep the normalized load constant, e.g. at 0.8), the resulting burst scale component (and hence the AQS) from the voice traffic with the same characteristics (the same on/off times and peak rate) is different. We take scenario 3 traffic as an example, and apply 37 sources of scenario 3 traffic over a T1 link of 1.544Mbps or 49 sources over a E1 link of 2.048Mbps (to keep a load of 0.8). Figure 5-17 indicates that the same type of traffic over a different link bit-rate has different queue state distribution, and the queue state distribution over an E1 link is shorter due to its higher service rate. Based on the formulas introduced in section 3.2.4, the corresponding decay rate and AQS are shown in Table 5-4.

	Burst-scale decay rate	AQS by burst-scale	Packet-scale decay rate	AQS by packet-scale	Total AQS
T1 link	0.9851	66	0.6587	2	68
E1 link	0.979	46	0.6587	2	48

Table 5-4 Burst scale decay rate and AQS over different link (scenario 3 traffic, load=0.8)

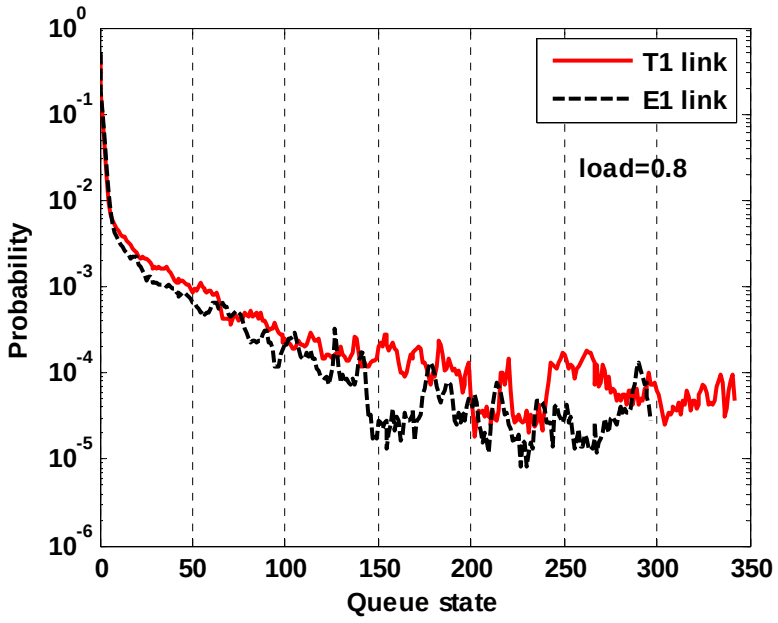

Figure 5-17 Queue state distribution over different link (scenario 3 traffic)

Since the resulting AQS over an E1 link is shorter than the AQS over a T1 link, the guidelines derived for a T1 link continue to apply over an E1 link. Taking the T1 link as the lowest link speed worth considering, as we scale up link bit-rate and load (to keep the normalized load constant, e.g. at 0.8) then the burst scale decay rate, and hence AQS decreases (and the loss decreases, too). Hence the guidelines derived for the T1 link will continue to apply over other higher bit-rate links, but will just not be such a tight bound.

5.4 Summary

In Chapter 4, we have discussed RED's performance improvements for voice traffic, but a challenge for the wide application of RED still exists, that is what QoS in terms of delay, jitter and loss can be achieved with the RED algorithm and how to configure those RED parameters to achieve a satisfactory QoS. Actually, although

the behaviours of RED in conjunction with TCP have been widely studied, the configuration of RED's parameters, especially the minimum and maximum thresholds, has been the major limitation of RED's application to TCP.

In this chapter, first, based on the analysis of traffic burstiness, we identify the most bursty voice traffic modeled by exponential on/off sources. This has the longest queue state distribution, the largest (flattest) burst scale component and hence the longest AQS.

Secondly, a RED configuration guideline for voice traffic is derived from extensive simulation results, and it takes into account the targeted performance requirements of voice traffic and the applied load. Because the average queue size is determined by the queue state distribution, burstier traffic causes a longer queue state tail thus resulting in a longer AQS. For a fixed buffer size, burstier traffic suffers higher loss due to the fact that it needs more buffer space to hold its longer queue state tail. So based on this analysis for the relationships between buffer size and traffic characteristic, the most bursty traffic was used to derive bounds for the worst case AQS and loss rate. These derived functions describe the AQS for the most bursty voice traffic, and so can be used as a bound for other less bursty traffic as well. As for loss rate, the loss rate of the most bursty voice traffic under a small maximum threshold can be used as the bound for other less bursty traffic. This estimate of loss rate is not as accurate as for the AQS, but because VoIP traffic is more tolerant of loss, this rough estimate is acceptable.

This predicted bound for AQS and loss rate with VoIP traffic under RED will help network service providers to predict the worst case loss rate and delay that the voice traffic will experience per hop given specific RED parameter values and hence the end-to-end performance. If the required QoS cannot be met, the end-to-end performance can be adjusted by changing the admissible load and maximum threshold.

Thus the estimated bound will help service providers to set appropriate RED parameter values within the routers based on target delay and loss requirements for voice traffic. This can be extended to provide appropriate service for voice traffic at different levels, e.g. gold, silver and bronze service. A service provider can allocate

voice traffic with different QoS requirements to different class of queues configured with different RED parameter values. For example, the gold level voice traffic experiences RED queues with smaller maximum thresholds and lower applied load in order to benefit from lower delay and loss.

In addition, the configuration guideline can help increase the bandwidth utilization. As the configuration guidelines in Table 5-3 show, when the load is not high, e.g. load=0.7 or 0.8, the loss rate is no more than 0.01, which is acceptable for most voice traffic, and the AQS is small and stable whatever the RED thresholds are. The QoS of voice traffic under this situation is satisfactory. But service providers always want to achieve a high utilization, i.e. approaching 1.0, but need to maintain QoS commitments. Based on the configuration guidelines, a service provider can select the highest load under which the delay and loss is acceptable. For example, we can increase the load to 0.9, and the resulting loss rate is no more than 0.03. If this loss rate is acceptable, the RED thresholds can be adjusted based on the required delay, and this provides a flexible and predictable solution for different scenarios.

Lastly, applications of the configuration guideline are discussed. With traffic segregation and WFQ scheduling, the guideline can provide a more accurate estimate of delay, and it can be extended to links with higher bit-rates as well.

To summarize, different RED thresholds, especially different RED maximum thresholds, will result in distinctly different but predictable AQS. Of course, the smaller AQS occurs with the cost of a higher loss rate, but the fluctuation of loss rate is quite small. Since real-time traffic is more tolerant of loss than delay, the shorter queuing delay at the cost of a little higher loss rate provides an acceptable trade-off. So following the configuration guideline, a predictable service for VoIP is able to be provided and adjusted through changing the maximum threshold of RED based on the QoS requirements.

Chapter 6 Further Applications of RED's QoS Improvements

We have discussed RED's performance improvements for VoIP traffic over a single bottle-neck topology, and simulations have verified that RED can greatly improve the QoS of VoIP traffic in terms of delay, jitter and loss over loaded and unloaded networks. Is the RED algorithm robust enough to provide performance improvements for VoIP over more realistic and complex scenarios? Do the performance improvements also work for other real-time traffic, such as video traffic? This chapter investigates these questions. First, a study about the end-to-end performance of VoIP traffic over a multiple hop topology is provided. Secondly, the RED algorithm is applied to video traffic to observe the corresponding performance. Lastly, the performance of VoIP traffic is studied over a more complex architecture: the DiffServ architecture.

6.1 VoIP over Multiple Hops

In this section, a more realistic multiple bottleneck topology is used to evaluate the end-to-end performance of VoIP traffic. This multiple-bottleneck topology is often used in the study of end-to-end performance, such as in [Kar01] [Ste02] [Bou02] [Liu02] etc. As depicted in Figure 6-1, a set of traffic traverses a congested network path that consists of n hops before reaching the receiving end. This is often referred to as foreground traffic [Liu02] [Ste02]. Cross traffic is generated at each intermediate router Ci (i=1, 2,…n) from background traffic sources. This background traffic competes for resources with the foreground traffic within router queues and leaves the system. The foreground traffic continues on to the next hop, C_{i+1}, but the background traffic at C_i is routed elsewhere. As a result, the load within each hop can be adjusted by applying different numbers of background traffic sources.

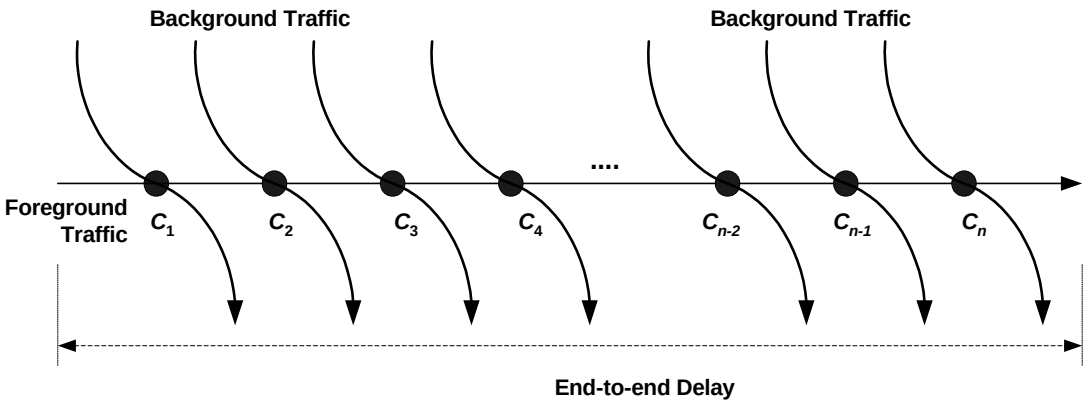

Figure 6-1 Multiple-hop Topology

In our simulations, the queue management algorithm within each router is set as RED or drop-tail as appropriate. All links other than those between adjacent routers have 10 Mbps capacity and 0 ms one-way propagation delay. As for the links between adjacent routers, the link bandwidth is set as 1.544 Mbps and 0 ms.

In this section, to provide a comparable analysis, we still use the same voice scenario illustrated in Chapter 4, whose characteristics are summarized in Table 5-1 and identified as scenario 1. Foreground traffic traverses 4 hops from the source to the destination. At each hop, background traffic with the same characteristics is injected into the router to compete for bandwidth with the foreground traffic and provide a high mean load within each queue. Meanwhile the end-to-end performance in terms of delay, jitter and loss, (instead of per hop performance) is observed.

6.1.1 End-to-end Delay Distribution

The minimum and maximum thresholds of the four routers that the foreground traffic traverses are set as 10 and 20 respectively as an example to observe the corresponding end-to-end performance, and to compare RED with a simple drop-tail (queue size 100). To give a full analysis under a range of network loads, the number of foreground traffic sources is varied to increase the load from 0.7 to 1.1 with a step of 0.1, while the background traffic is always kept the same.

Figure 6-2 and figure 6-3 show the end-to-end delay distribution measured across

the four hops for RED (10, 20) and drop-tail (100) scenarios respectively, and Figure 6-4 and Figure 6-5 give the corresponding complementary cumulative delay distribution. These figures demonstrate the substantial differences, particularly at high load values. When the load offered to each link is above 100%, it is clear that the bulk of the end-to-end delay distribution is above 200ms for drop-tail queue management. For the RED scenario, it is the opposite. These results demonstrate how RED is able to regulate very effectively the end-to-end delay distribution for VoIP traffic on a congested path.

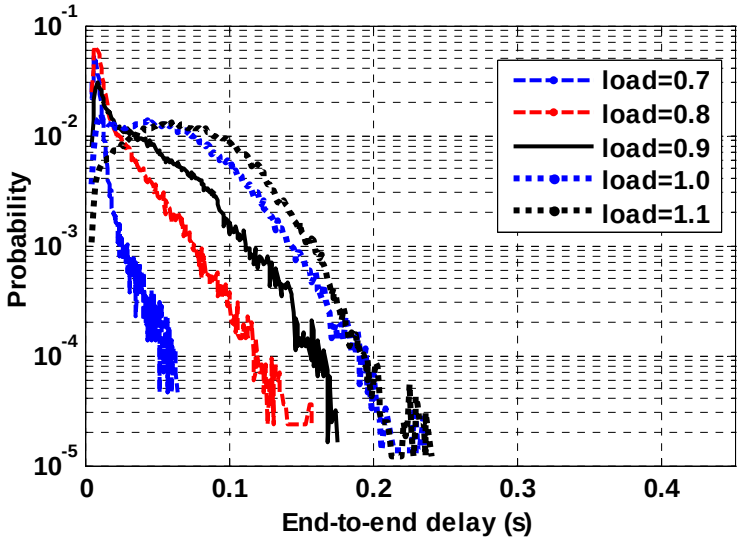

Figure 6-2 End-to-end delay distribution for 4 hops under RED algorithm

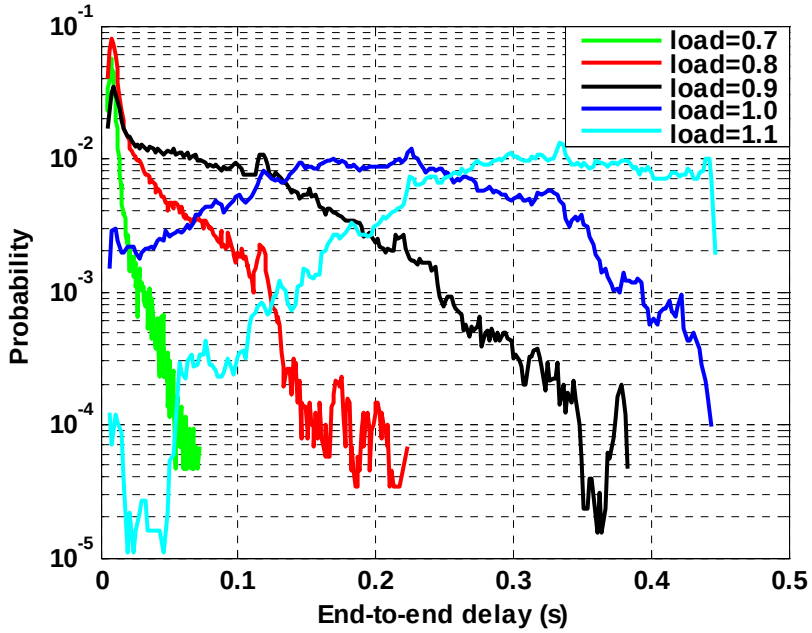

Figure 6-3 End-to-end delay distribution for 4 hops under drop-tail algorithm

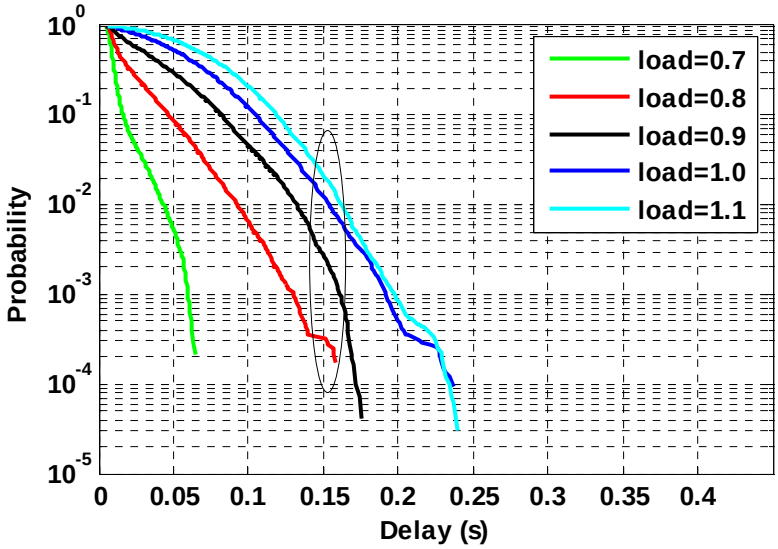

Figure 6-4 Complementary Cumulative Delay Distribution for 4 hops under RED algorithm

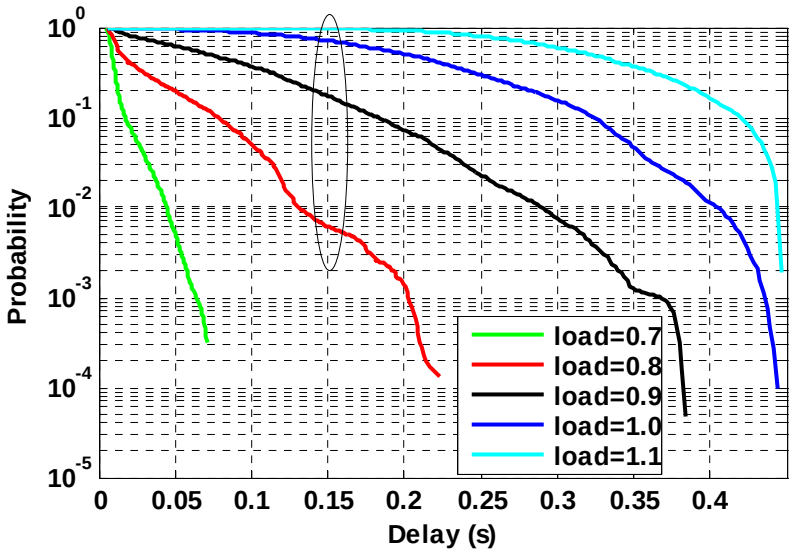

Figure 6-5 Complementary Cumulative Delay Distribution for 4 hops under drop-tail algorithm

Taking 150ms as a delay bound, the probabilities of exceeding this delay, for both RED and drop-tail, are summarized in Table 6-1. We can see that if a simple drop-tail algorithm is used, when the path is heavily loaded, lots of packets will arrive too late to be useful. However if RED algorithm is used, the corresponding tail probability is decreased greatly. For example, even if the link is overloaded with a load of 1.1, the tail probability under RED is still acceptable, while the counterpart under drop-tail is totally useless. This delay control can make it possible to utilize more bandwidth (by applying higher load) while keeping the delay under the bound. Even if some sudden bursts occur and the load is temporarily higher than

the service rate, the delay will degrade gracefully and not to unacceptable levels.

Load	Drop-tail	RED
0.7	0	0
0.8	0.006	0.0003
0.9	0.2	0.003
1.0	0.7	0.01
1.1	0.95	0.02

Table 6-1 Probability of exceeding a delay bound of 150ms for RED & drop-tail

6.1.2 Jitter

Next we consider another performance metric: jitter. Figure 6-6 gives the corresponding jitter comparison between RED and drop-tail under different loads for the scenario described above. It clearly shows that jitter increases greatly with increasing load under drop-tail where there is not any QoS control. In contrast, the RED algorithm can maintain a stable level of jitter within an acceptable range under overload conditions. This controlled jitter is important to maintain the QoS of real-time traffic.

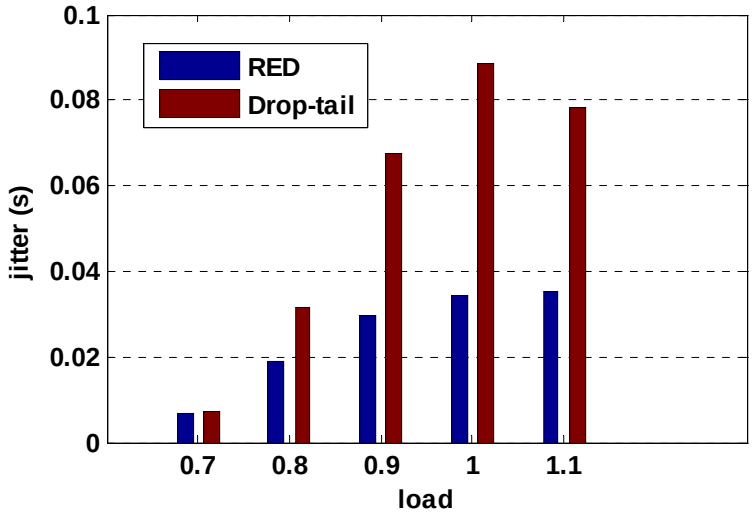

Figure 6-6 Jitter for 4 hops under drop-tail and RED algorithm

We can see that jitter increases with increasing load, because the higher the load, the longer the queue will be occupied, and the more space for fluctuation in the queue size (which is the source of jitter). In addition, the jitter under load 1.1 is lower than the jitter under load 1.0 for drop-tail. This phenomenon results from the full queue. When the path is heavily loaded by on/off traffic, such as 110% of the bandwidth, the probability of each queue being full increases, as seen from Figure 6-3, where the largest end-to-end queuing delay corresponds to all four queues being full. As this probability increases, for loads above 1.0, the fluctuations that constitute jitter decrease in probability. Hence overloaded tail-drop queues have large delay and reduced jitter. As for RED, packets are dropped before the queue becomes full, so this full queue phenomenon seldom occurs.

6.1.3 Loss

6.1.3.1 Effective Loss

As explained in chapter 4, the concept of effective loss (including both the actual losses in the network and those packets that are delivered, but too late to be of use) is a more accurate way to explain the loss performance of real-time traffic. So we give the corresponding probability of being out of contract on this 4 hop path for both RED and drop-tail under different load conditions. As shown in Figure 6-7 and Figure 6-8, if we still take 150 ms as the bound for end-to-end queuing delay, RED can greatly decrease the probability of packets being out of contract compared with drop-tail. This can help to maintain VoIP QoS when the network faces an unexpected increase in load.

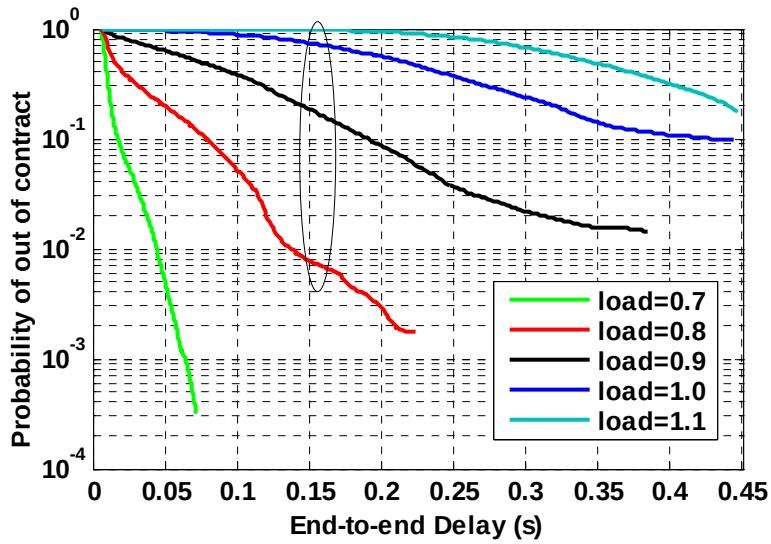

Figure 6-7 Out of contract probability for drop-tail algorithm

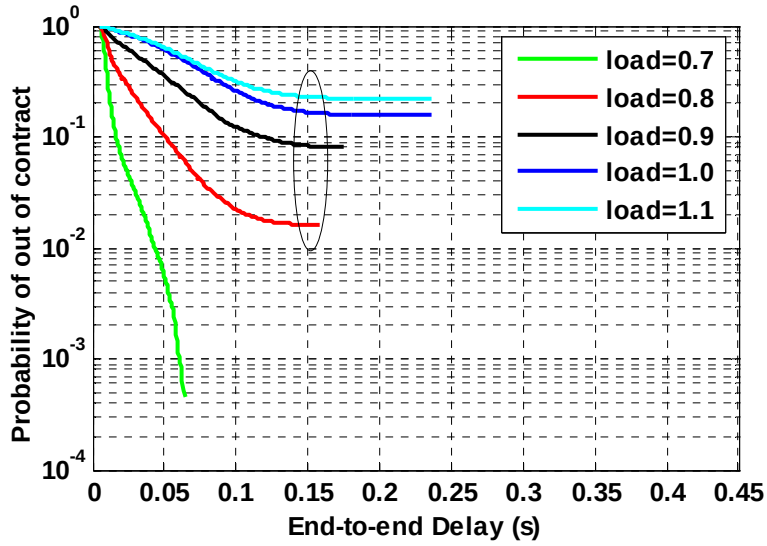

Figure 6-8 Out of contract probability for RED algorithm

6.1.3.2 Consecutive Loss

Consecutive loss can greatly harm the QoS of real-time traffic. In this section, we analyze packet loss of foreground traffic on a congested path. We analyze the sequence of effective lost packets to observe their contiguity. We record the lost packet ID, compute the intervals between successive lost packet IDs, and estimate the probability of each interval. If the lost packet's id is not consecutive (lost packet id interval > 1), it means the dropped packets are not consecutive. The lower the probability of small intervals, the better. Figure 6-9 and Figure 6-10 gives the results for both RED and drop-tail under the load of 0.8. Figure 6-11 gives the same

results in a log-linear scale to show details in the Y axis. As shown in these figures, it indicates clearly that RED decreases the probability of consecutive loss to a smaller value (about 0.16) than drop-tail (about 0.9) and RED distributes the sequence of lost packet ID across a wider range. This decreased consecutive loss will help to maintain the QoS of VoIP when some packets have to be dropped.

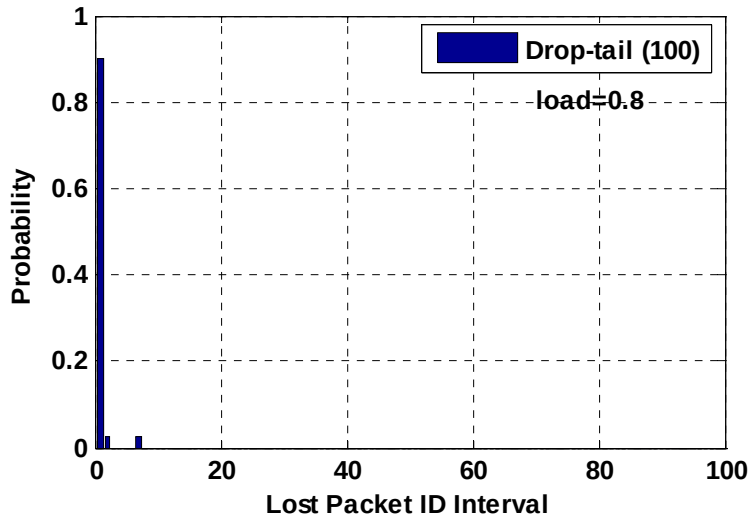

Figure 6-9 Lost packet ID interval distribution for drop-tail (load=0.8)

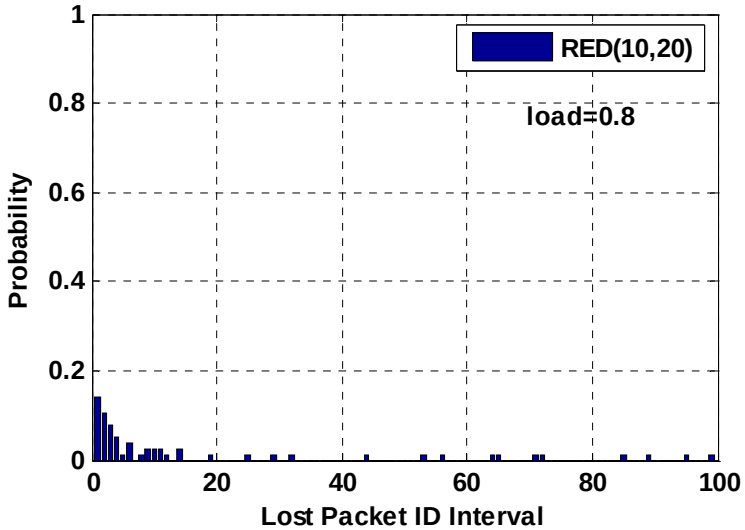

Figure 6-10 Lost packet ID interval distribution for RED (load=0.8)

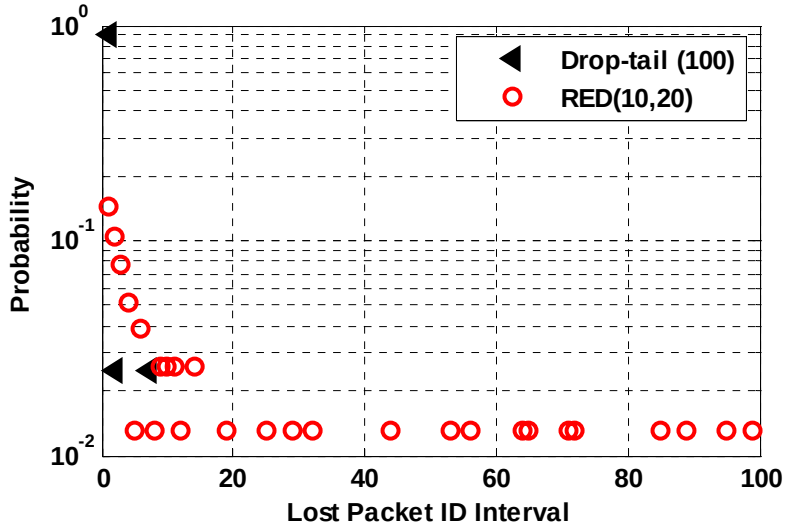

Figure 6-11 Lost packet ID interval distribution (log-linear scale, load=0.8)

In addition, Figure 6-12 to Figure 6-17 give the lost packet ID interval for the higher load of 0.9 and 1.0, of which, Figure 6-14 and Figure 6-17 are shown with a log-linear scale to show the details in the Y axis. These figures also indicate clearly that, RED decreases the probability of consecutive loss to a smaller value (about 0.28 and 0.4 respectively) than drop-tail (about 0.9 and 0.99 respectively) through RED's distributing the sequence of lost packet ID across a wider range. This decreased consecutive loss will help to maintain the QoS of VoIP over a congested path.

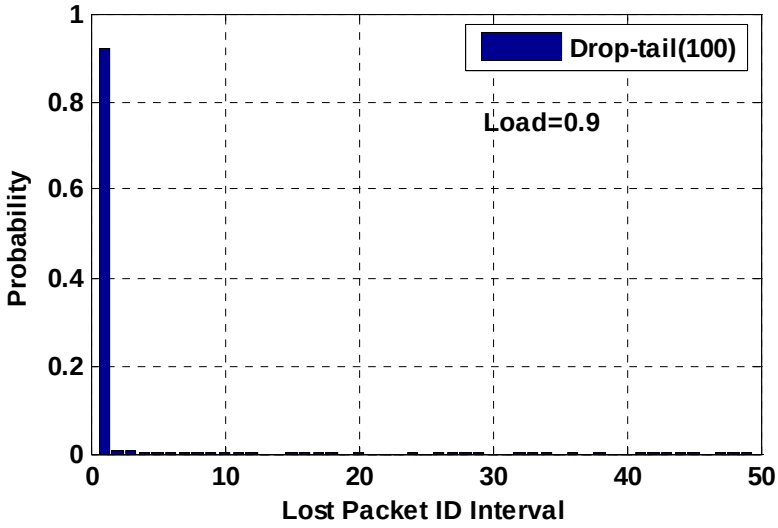

Figure 6-12 Lost packet ID interval distribution for drop-tail (load=0.9)

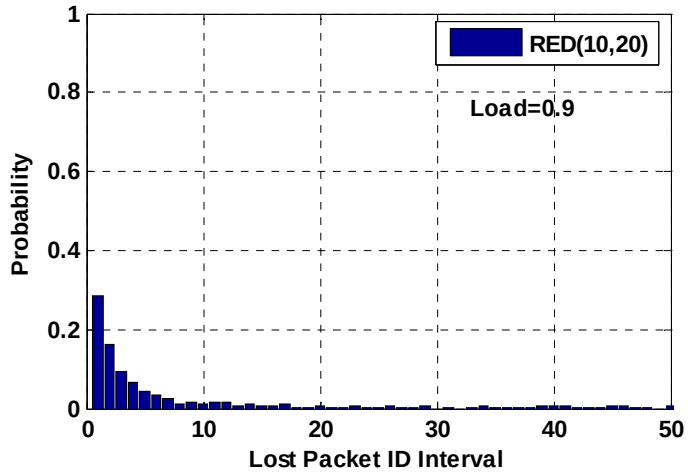

Figure 6-13 Lost packet ID interval distribution for RED (load=0.9)

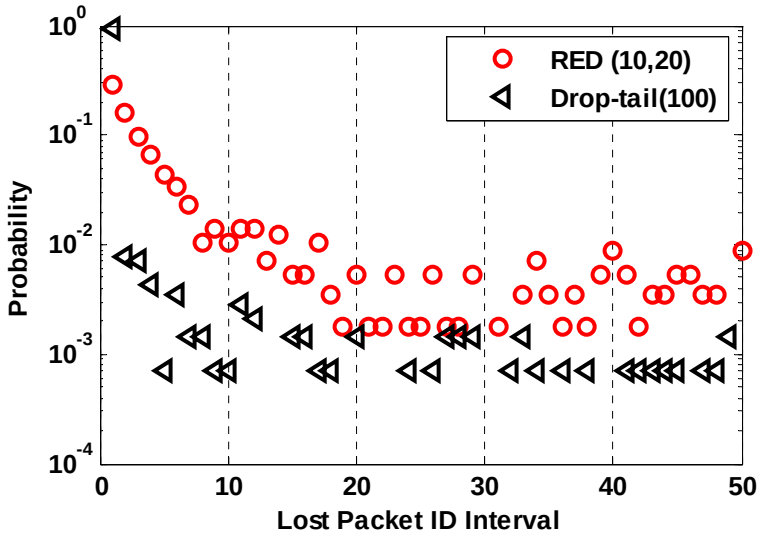

Figure 6-14 Lost packet ID interval distribution (log-linear scale, load=0.9)

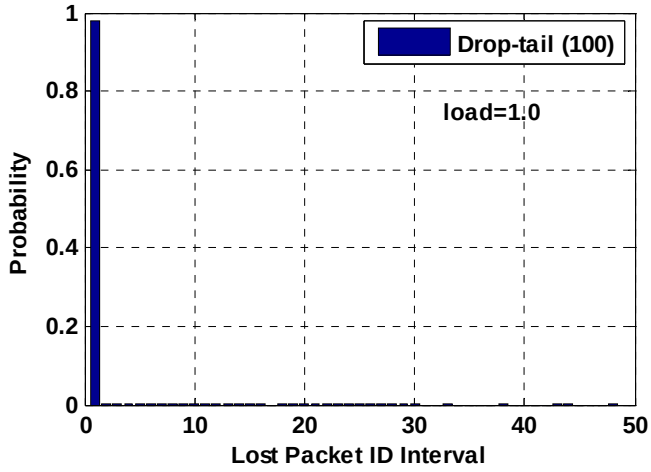

Figure 6-15 Lost packet ID interval distribution for drop-tail (load=1.0)

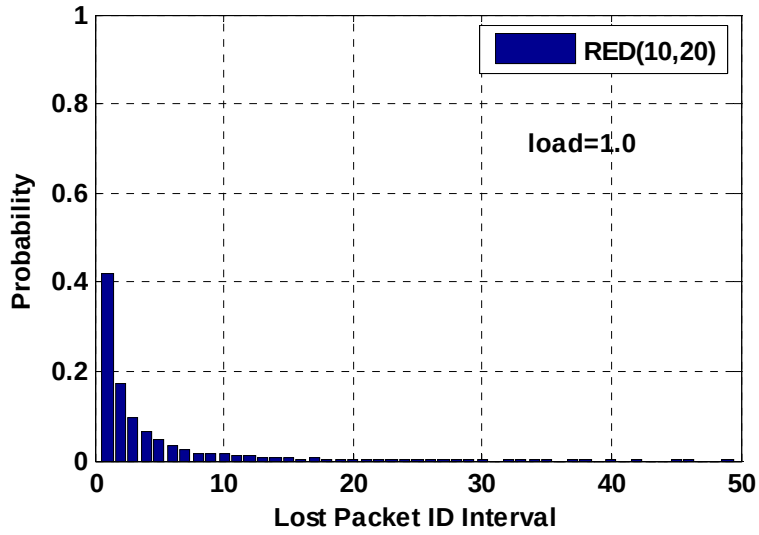

Figure 6-16 Lost packet ID interval distribution for RED (load=1.0)

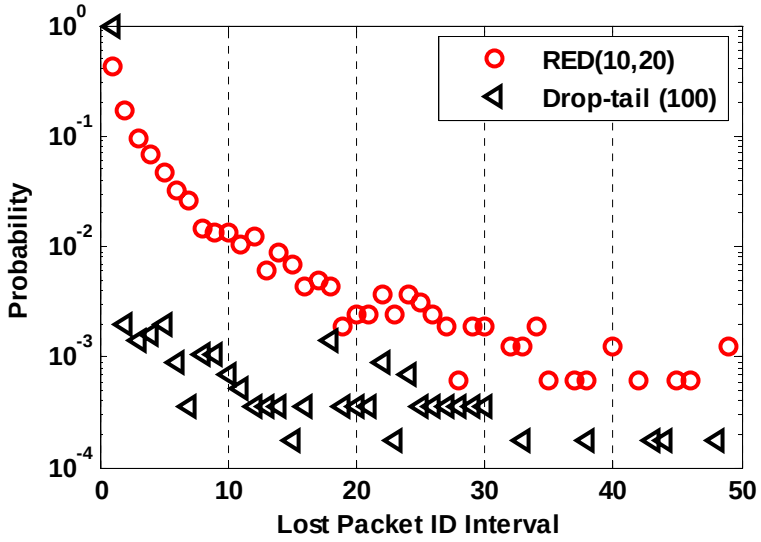

Figure 6-17 Lost packet ID interval distribution (log-linear scale,load=1.0)

In addition, Figure 6-18, Figure 6-19 and Figure 6-20 give the packet loss burst length distribution (in packets) for both drop-tail and RED. These figures indicate that RED can help to decrease the packet loss burst length, which will help not to degrade the quality of traffic so much.

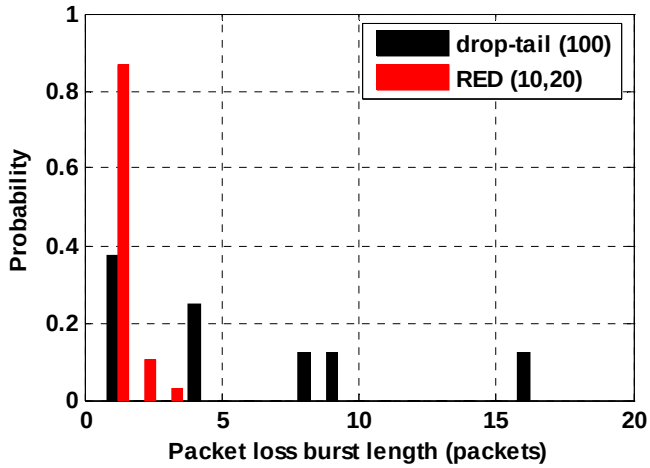

Figure 6-18 Packet loss burst length (load=0.8)

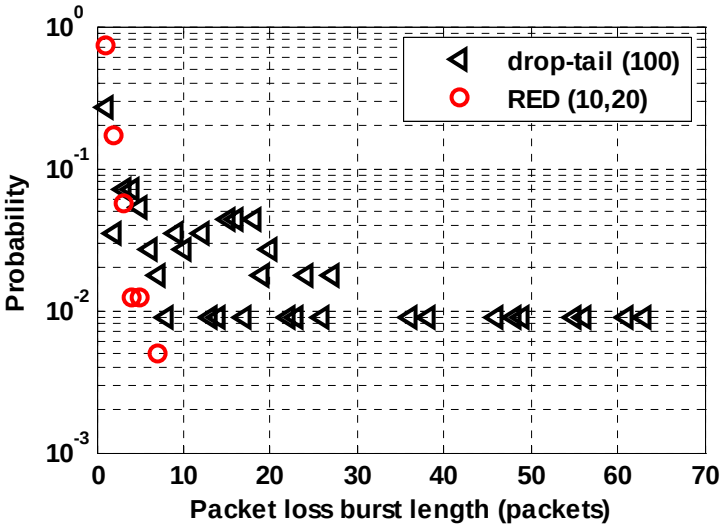

Figure 6-19 Packet loss burst length (load=0.9)

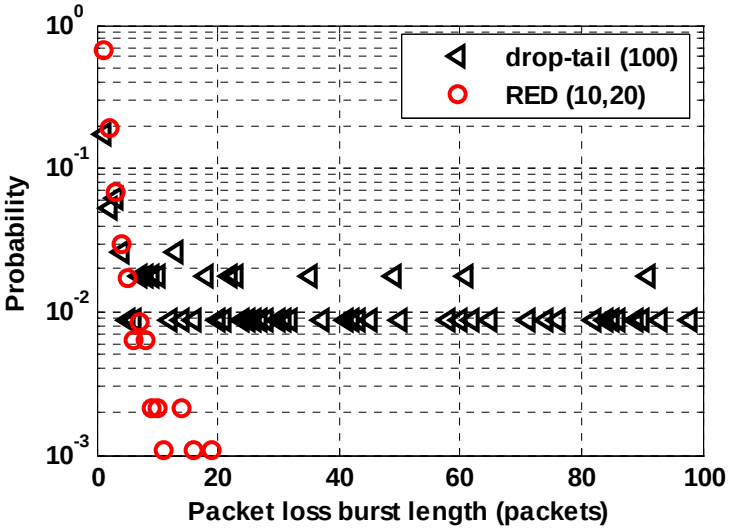

Figure 6-20 Packet loss burst length (load=1.0)

Through the performance comparison between RED and drop-tail in terms of delay, jitter and loss under different loads above, it is clear that RED's performance control can work over a multiple hop topology thus providing end-to-end performance improvements for voice traffic, even for a heavily loaded network path. To achieve these performance improvements, the routers need to use the RED algorithm instead of traditional drop-tail. However this needs no changes to existing network architecture.

6.1.4 Verification for the Proposed Bounds

We can verify the queuing delay and network loss rate bounds proposed in Chapter 5 against results for this 4 hop path. Since all the link propagation delays have been set as 0 ms in our simulations, the sum of the mean end-to-end queuing delay within each hop is equal to the mean end-to-end delay here. If the RED parameters for each router along the path are known, based on the proposed formulas, the bound for the AQS within each hop can be calculated, and thus the queuing delay bound within each hop is known. So the mean end-to-end queuing delay bound is simply the sum of the queuing delay bound within each hop, and the total loss rate is no more than the sum of the loss rate bound within each hop. If we adopt the parameters and scenarios used in the previous sections, the RED minimum and maximum thresholds are set as 10 and 20 respectively, and voice traffic named as scenario 1 is used as the traffic source. We can compare the actual end-to-end delay and loss obtained from simulations with the calculated bound. The comparison between the estimated bound and the actual values is shown in Table 6-2 below.

Load	Bound for AQS under RED (10, 20)	Estimated queuing delay (s)	Simulated e2e delay (s)	Estimated loss rate	Simulated network loss rate
0.7	3	0.0133	0.01059	<0.008	0.00024
0.8	5	0.02217	0.0208	<0.032	0.016
0.9	10	0.04435	0.0388	<0.12	0.0806
1.0	15	0.06653	0.057	<0.28	0.1567
1.1	20	0.0887	0.0705	<0.4	0.22

Table 6-2 Comparison between estimated bound and actual values

This comparison indicates that our proposed bounds for queuing delay and loss rate function well as limits on the actual queuing delay and loss rate, and thus provide a predictable service.

6.2 Congestion Control for Video Traffic

We have mentioned in chapter 2 that real-time interactive video traffic has similar QoS requirements to VoIP, but has distinctly different traffic characteristics from VoIP. For example, voice packets have short and constant sizes, while video packets have large and variable sizes. Video traffic does not exhibit on/off characteristics but works as a Variable Bit Rate (VBR) source, which always has bits to send but with variable bit rates. Since RED can provide a satisfactory QoS control for VoIP, is it possible to extend RED's application to real-time video traffic? In this section, research results for video traffic and the RED algorithm are given.

As introduced in section 3.1.2, the performance study of real-time traffic often depends on video traces which are captured from real video streams; our study uses traffic selected from [Lib] as well. Figure 6-21A shows three example profiles of the video traffic used in [Lib], which have different rates of around 16Kbps, 64Kbps and 256Kbps respectively, and different packet size of around 80 bytes, 320 bytes and 1278 bytes. Figure 6-21B describes the packet size of video traffic with 64Kbps against its packet arrival time within a period. These figures show that video traffic

is always transmitting at some variable rate and variable packet size instead of exhibiting the on/off patterns characteristic of voice traffic. For the feasibility of simulation, we just select the video traffic with 64Kbps in our simulations to avoid too many or too few numbers of traffic source.

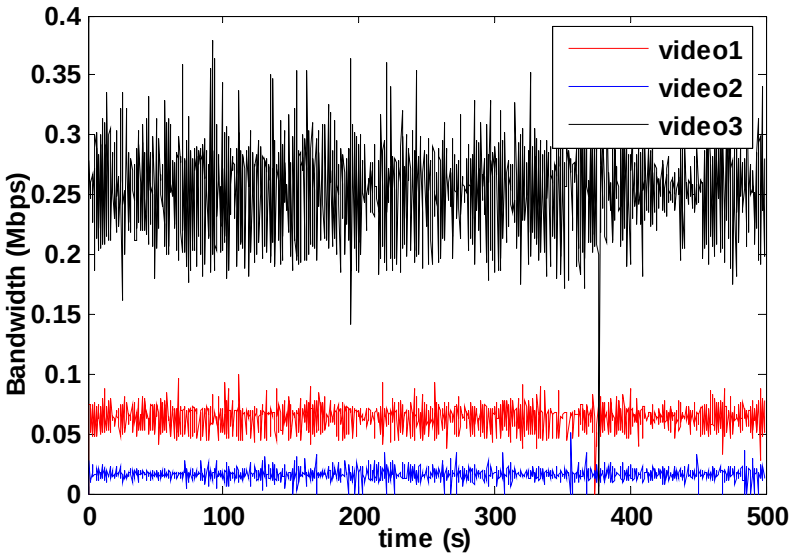

Figure 6-21A Example profiles of three video traffic source types

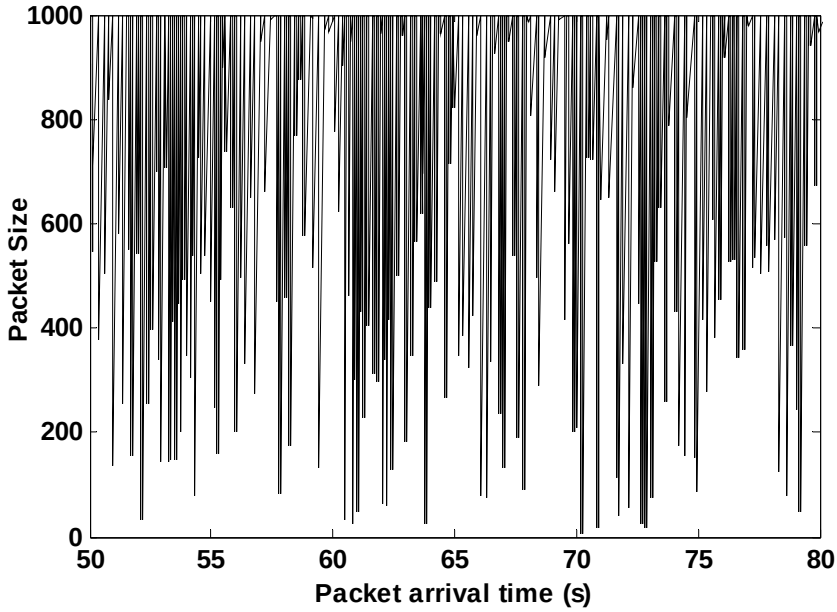

Figure 6-21B Packet size vs packet arrival time for video traffic with 64Kbps

The same single bottleneck topology shown in chapter 4 is used, and the propagation delay of each link is set as 0 ms, so the end-to-end delay is equal to the sum of per-hop queuing delay. First, we proceed as with the voice traffic, varying the number of video sources (to form different loads, e.g. a load of 0.8 requires 19

video traffic sources of 64Kbps) which start at randomly selected times and applying them to a simple FIFO drop-tail algorithm (whose buffer size is set as 100 packets) in order to observe their original end-to-end delay distribution (queuing delay distribution) where no QoS control is provided. Here we observe the end-to-end delay distribution instead of the queue state distribution due to the variable packet size of video traffic. Secondly, the RED algorithm is used for the same workload, and we compare the end-to-end distribution under RED with that under drop-tail.

The end-to-end delay distributions under drop-tail for different loads are shown in Figure 6-22. Clearly, the end-to-end delay distribution of video traffic is rather different from that of voice traffic. When the load of video traffic is no larger than or close to its allocated bandwidth, the end-to-end delay distributions under different loads are similar, and the increasing load does not affect the performance of this video traffic greatly. But once the network is overloaded, each of the link queues fills up and the end-to-end delay distribution concentrates around these nearly full queues. This greatly delays the interactive video packets and harms the quality of the video.

These delay distributions indicate that there is little or no burst scale behaviour, and that the video streams are behaving like constant rate sources. Under these circumstances, RED's ability to filter the decay rate is not as evident as with on/off voice traffic. As Figure 6-23 shows, even when the RED minimum and maximum thresholds are set as small as 1 and 5 respectively, there is little difference in the delay distributions between RED and drop-tail when the network is not overloaded, e.g. load=0.8, and 0.9. However, once the network is overloaded, the delay for video traffic under drop-tail becomes very large, whereas the RED algorithm provides a very effective control of the delay.

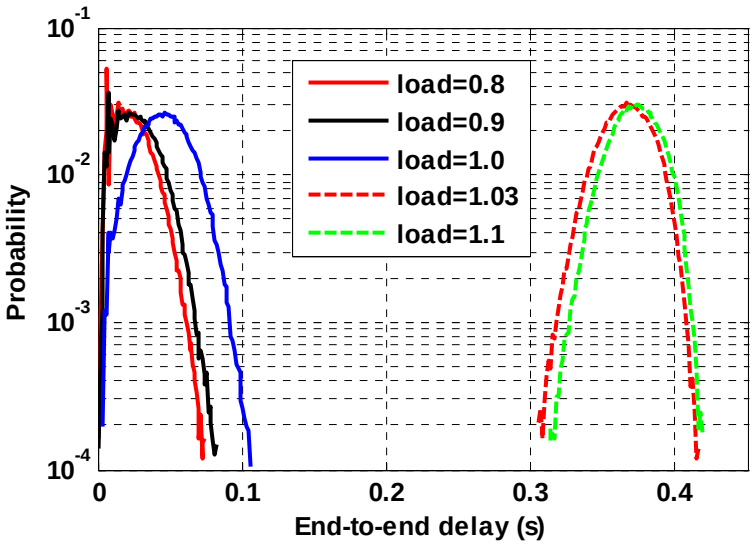

Figure 6-22 End-to-end distribution of video traffic under drop-tail

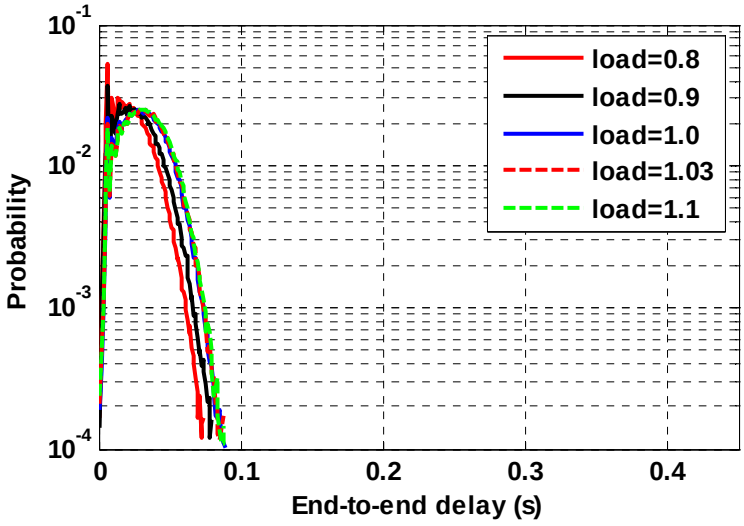

Figure 6-23 End-to-end distribution of video traffic under RED

In addition, as seen from Figure 6-22 and Figure 6-23, for each end-to-end delay distribution, although the delays are different, the range between the minimum and maximum delay, an indicator of jitter as introduced in chapter 2, does not vary greatly. This stable jitter is another characteristic associated with the multiplexing of constant bit rate streams.

A clearer view about the delay performance is given by the complementary cumulative delay distribution for video traffic under different loads in Figure 6-24. When the link is overloaded, such as a load of 1.03, all the packets under drop-tail are beyond the acceptable maximum delay (e.g. 0.15 s), whereas RED can provide

effective control of the end-to-end delay (actually queuing delay) under this overloaded situation and limit the delay to a much smaller range.

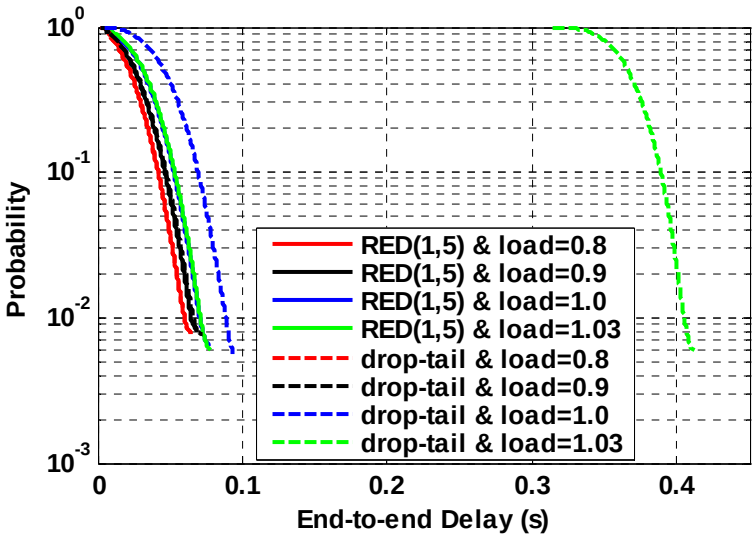

Figure 6-24 Complementary Cumulative end-to-end delay distribution

When the link is overloaded, RED can maintain the queuing delay within a controllable range, and this can be adjusted by changing the RED thresholds. As shown is Figure 6-25, when the link is slightly overloaded, such as a load of 1.03, for drop-tail, all the arriving video packets are beyond the maximum acceptable delay (0.15s), whereas RED can achieve different delay by setting different RED thresholds.

Figure 6-26 gives the probability of being out of contract for video traffic under different RED thresholds and drop-tail when the load is 1.03. With a delay bound of 150ms, the value of the out of contract probability is the value on the Y axis at the end of each curve, as shown by the ellipse in Figure 6-26. We can see that for RED this is between 0.04 and 0.05, which is much better than the corresponding out of contract probability under drop-tail (almost 1.0).

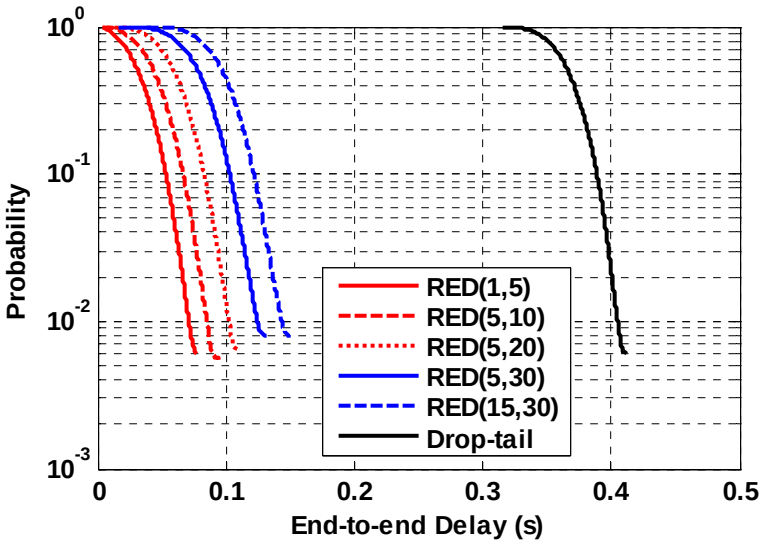

Figure 6-25 Complementary Cumulative end-to-end delay distribution (load=1.03)

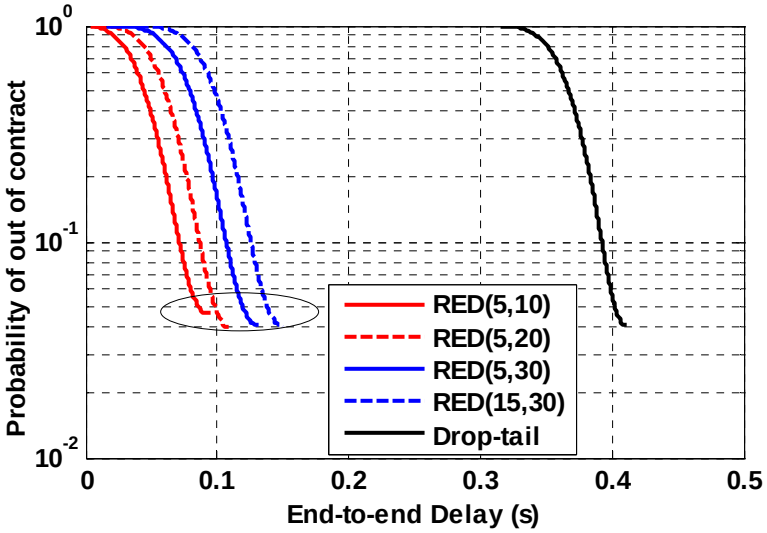

Figure 6-26 Out of contract probability (load=1.03)

From the analysis above, a conclusion can be drawn: when the link is not overloaded, RED can provide similar QoS for video traffic as with drop-tail. Once the link is overloaded, the QoS under drop-tail will deteriorate to unacceptable levels, but RED can provide excellent performance control by limiting both the queuing delay and out of contract probability over the overloaded link, thus providing a graceful degradation when network resources are limited.

6.3 RED's Performance Improvements under DiffServ

Performance improvements obtained by applying RED to voice and video traffic

have been studied and the corresponding configuration guidelines for voice have been derived. These guidelines are based on the study of a single hop bottle-neck topology with one class of service. As introduced in chapter 1, DiffServ is an architecture proposed to provide QoS in IP networks. Can RED's performance improvement for UDP traffic and the configuration guidelines work in a DiffServ architecture? In the following section, we evaluate the performance control obtained by applying the RED algorithm in a single hop DiffServ environment.

DiffServ [RFC2474] [RFC 2475] aggregates traffic flows by using some standard mappings. It is normally recommended to map multimedia/real-time applications (voice, video, etc) into the EF (Expedited Forwarding) class and to use PQ (Priority Queue) to guarantee the QoS of these services with small loss, low delay and assured bandwidth. However, as studied in [Ali04] [Yil01] [Kar00], those EF classes of traffic may have quite diverse QoS requirements and traffic characteristics, e.g. voice has smaller constant packet size, and video has larger and variable packet size. Such standard mapping leads to heterogeneous traffic multiplexing within the same queues in the aggregating routers, and this can deteriorate the service delivery. Thus, authors in [Ali04] [Yil01] [Kar00] suggest that voice and video need to be segregated into separate queues and to use WFQ to provide better QoS control.

WFQ, as a widely employed scheduling discipline in DiffServ, is used to separate different traffic flows and allocate the link bandwidth to different classes of traffic. The weight of WFQ represents the fraction of the total link capacity given to each individual flow (in IntServ) or each flow aggregate (in DiffServ). So we can separate UDP traffic from TCP traffic, or separate voice traffic from video traffic and allocate each class a fair fraction of the total bandwidth. But this allocation under DiffServ is slightly different from the single queue under IntServ. Firstly, for any work conserving scheduler, the spare bandwidth will be given to the other non-empty queues, which is also known as the bandwidth stealing effect [Kuz01]. Due to the bandwidth stealing effect, the prior assumption on the received bandwidth can be different from the actual bandwidth used. Secondly, because the essence of WFQ is to send packets from each queue within each cycle time, packets in each queue have to wait when packets in other queues are being served. This will bring some extra queuing delays. However, our configuration guidelines rely on the knowledge of the

load of each queue which is related to the allocated bandwidth, and the estimated bound is based on a single service queue. So we need more simulations to verify the estimated bound in the DiffServ architecture. In the following section, different simulations are run to demonstrate the proposed bound for voice traffic in a DiffServ architecture.

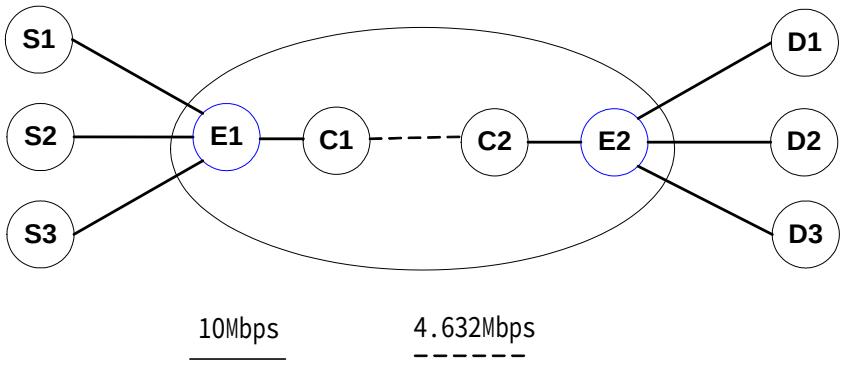

Figure 6-27 DiffServ Topology

The simulation topology is depicted in Figure 6-27. The DS domain consists of two core routers (C1, C2) and two edge routers (E1, E2). S1, S2, and S3 are connected with different classes of sources respectively and transmit packets to the corresponding destinations via three different queues in C1. A bottleneck link of 4.632Mbps connects C1 and C2, and all the other links have a bandwidth of 10Mbps. The propagation delay over each link is set to 0 ms. WFQ scheduling is used at the bottleneck (C1-C2) with pre-defined scheduler weights to provide equal share of the bottleneck bandwidth, i.e. 1.544Mbps for each type of traffic. The physical queue size of both RED queue and drop-tail queue is set as 500 (in terms of packets). The RED parameters of p_{max} and w_q are set as 0.1 and 0.002 individually as before, and the RED thresholds are varied over a range in order to observe the corresponding performance control by RED. By setting link propagation delays to 0 ms, we can express the end-to-end delay by using the queuing delay on the bottleneck link, where the queuing process occurs in our scenario. We consider different traffic workloads by adjusting the numbers of voice sources.

During the simulation design, because NS2 does not provide a built-in queue

monitor to monitor the queue size of each queue under DiffServ, we need to measure the end-to-end delay of the packets in each queue instead of the queue size. Then we can transform the estimated bound given by AQS into end-to-end delay via Equation 5-1. To verify this transform action is credible, we compare the actual end-to-end delay distribution with the one transformed from the queue state distribution for scenario 1 in a single class scenario.

As shown in Figure 6-28, the end-to-end delay distribution transformed from the queue state distribution accords well with the actual measured end-to-end delay distribution. Thus we can measure the end-to-end delay distribution, calculate the mean delay, and compare the mean delay with the proposed delay bound which has been transformed from AQS.

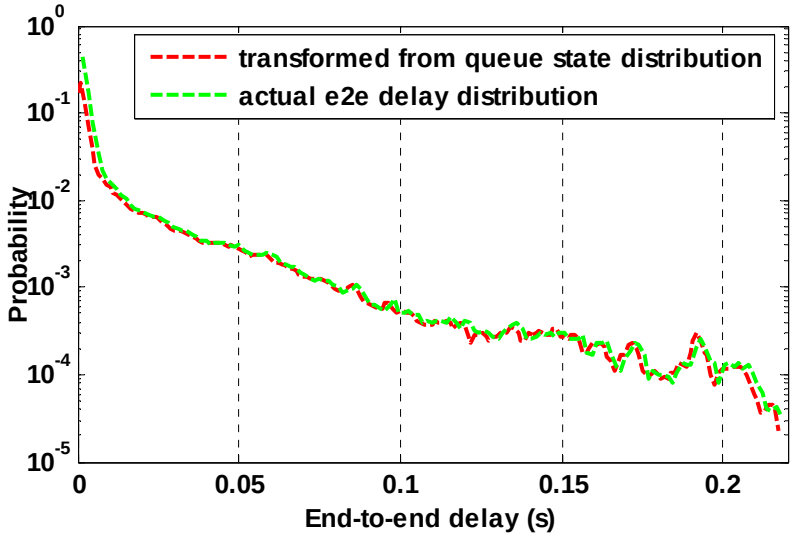

Figure 6-28 Comparison between e2e delay distributions (scenario 1)

Scenario 1 and scenario 4 (in Table 5-1), which have different characteristics in terms of on/off time, peak rate and packet size, are set as workload attaching to source class 1 and source class 2 respectively. Ftp traffic is attached to source class 3, and this is configured so that it always has packets to send to occupy its share of bandwidth. We record the end-to-end delay of class 1 and class 2, compute the mean delay and loss rate from simulations, and compare with the proposed bounds for mean delay and loss rate. When we record the end-to-end distribution of class 1 under different loads, the number of sources in class 2 is set as 132 to occupy all the share of its allowed bandwidth (4.632/3=1.544Mbps). Similarly, when we measure

the end-to-end delay of class 2, the number of sources in class 1 is set as 51 to achieve a load of 1.0 as well. The details of the settings are illustrated in Table 6-3.

To measure source 1			To measure source 2		
Num of source 1		Num of source 2	Num of source 1	Num of source 2	
Load=0.8	41	132	51	Load=0.8	106
Load=0.9	46	132	51	Load=0.9	120
Load=1.0	51	132	51	Load=1.0	132
Load=1.1	57	132	51	Load=1.1	146

Table 6-3 Numbers of sources

As shown in Figure 6-29 to Figure 6-32, the end-to-end delay distributions for class 1 (voice scenario 1) and class 2 (voice scenario 4) under different loads and different RED parameters are drawn and compared with the drop-tail algorithm. It is obvious that the end-to-end delay control by RED does work in a DiffServ enviroment. As found in chapter 4, the end-to-end delay is controlled by the maximum threshold; the minimum threshold has little impact on the end-to-end delay distribution. We only give the figures for scenario 1 and 4 under load 0.8 and 1.0. The full set of results can be found in Appendix D. Here, when we observe the end-to-end delay distribution of class 1, we use 132 class 2 traffic sources to occupy all its share of bandwidth. Actually, if we use fewer class 2 traffic sources, class 1 traffic can steal the left-over bandwidth and obtain a shorter delay.

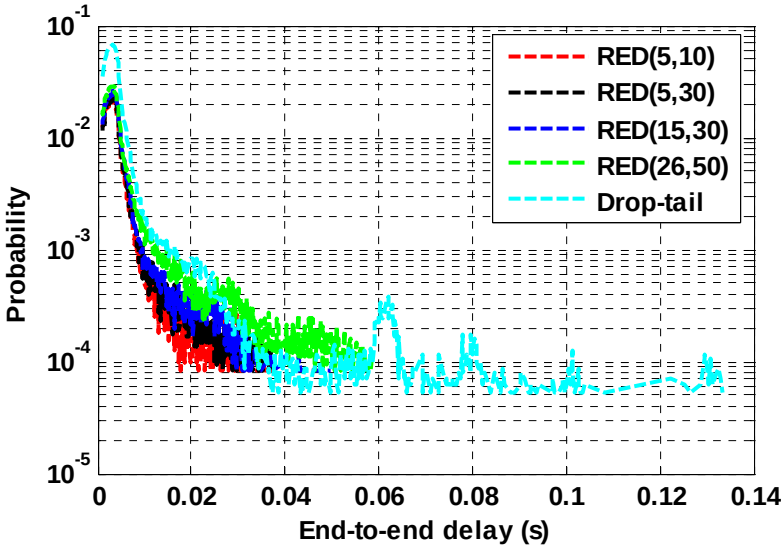

Figure 6-29 End-to-end delay distribution for class 1 (load=0.8)

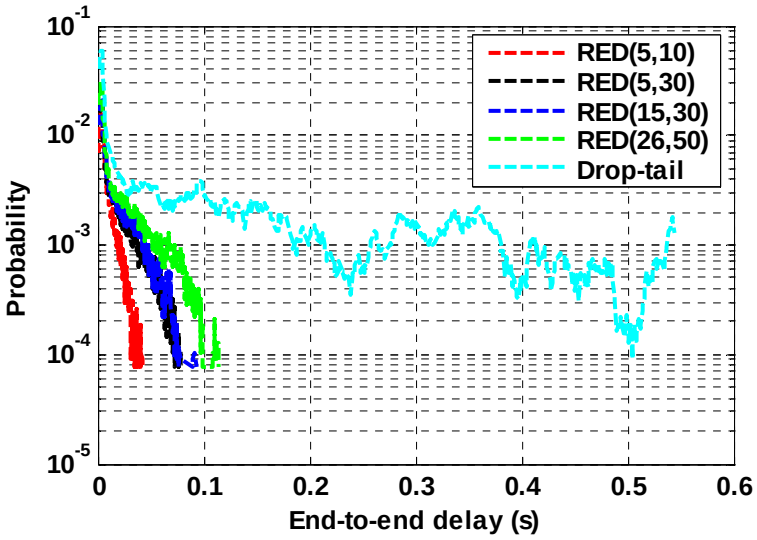

Figure 6-30 End-to-end delay distribution for class 1 (load=1.0)

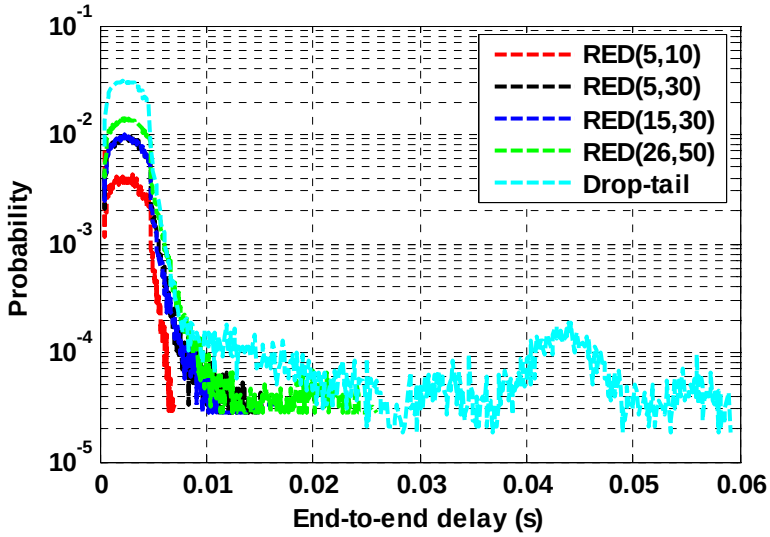

Figure 6-31 End-to-end delay distribution for class 2 (load=0.8)

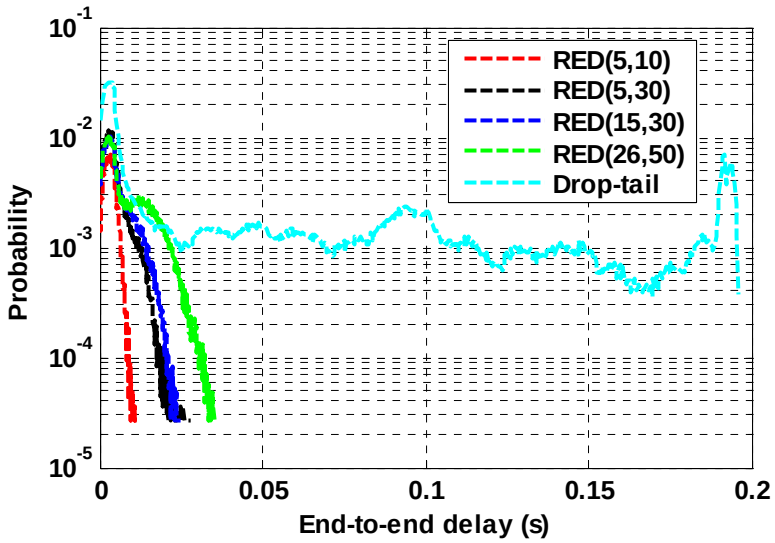

Figure 6-32 End-to-end delay distribution for class 2 (load=1.0)

Jitter, the standard deviation of end-to-end delay, is also controlled by virtue of RED's control of end-to-end delay. Figure 6-33 and Figure 6-34 give the corresponding jitter for each scenario, and the values obtained under RED are quite satisfactory even for heavily loaded conditions.

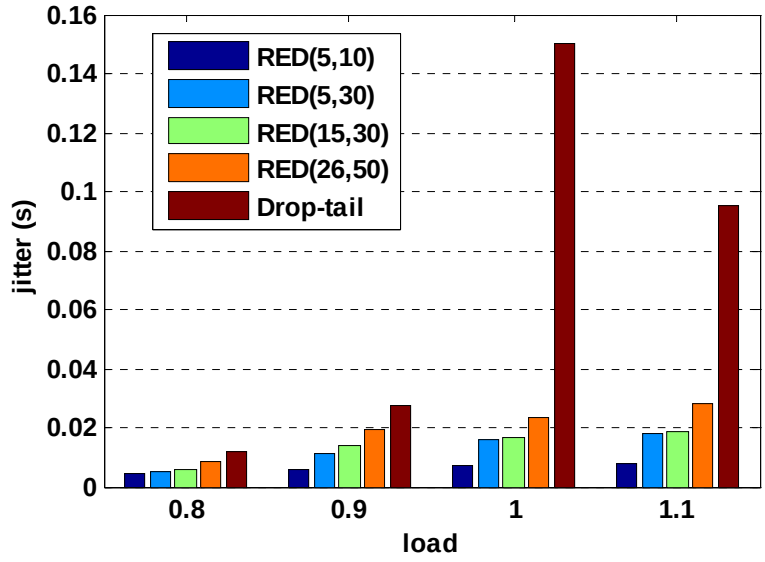

Figure 6-33 Jitter for class 1 traffic under different loads

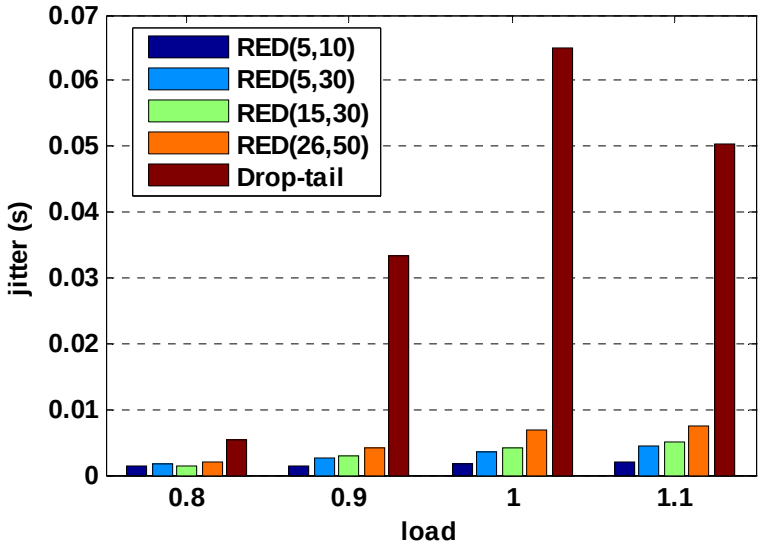

Figure 6-34 Jitter for class 2 traffic under different loads

Finally, Figure 6-35 and Figure 6-36 give the corresponding mean delay under different RED thresholds and loads for class 1 traffic and class 2 traffic. As seen from these figures, the mean delay bounds proposed for single class operation still limit the mean delay under DiffServ. The scales in the two figures are different because the traffic for class 1 (scenario 1) and class 2 (scenario 4) has different packet sizes and hence different service times per packet. Obviously, the smaller the packet size, the shorter the mean delay. In additon, when load is high, e.g. 1.0 and 1.1, the mean delay under DiffServ is lower than the proposed bound. This results

from the bandwidth stealing mentioned above, because when the voice traffic is overloaded, extra bandwidth obtained from other queues helps to reduce the average queuing delay of voice traffic. When the load is below 0.8, the queuing delay and loss rate are both small. Thus for lower loads, the estimated bound at a load of 0.8 can serve as a reasonable limit on the corresponding delay and loss.

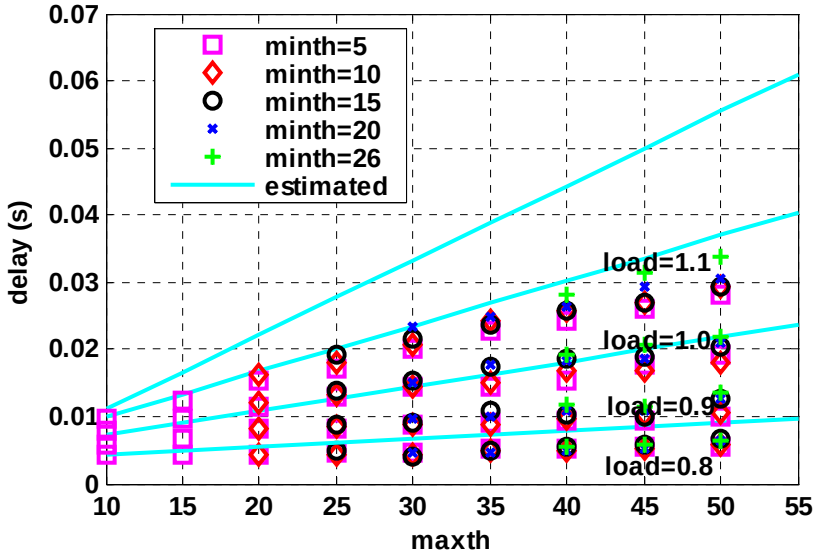

Figure 6-35 End-to-end delay of Scenario 1 under DiffServ

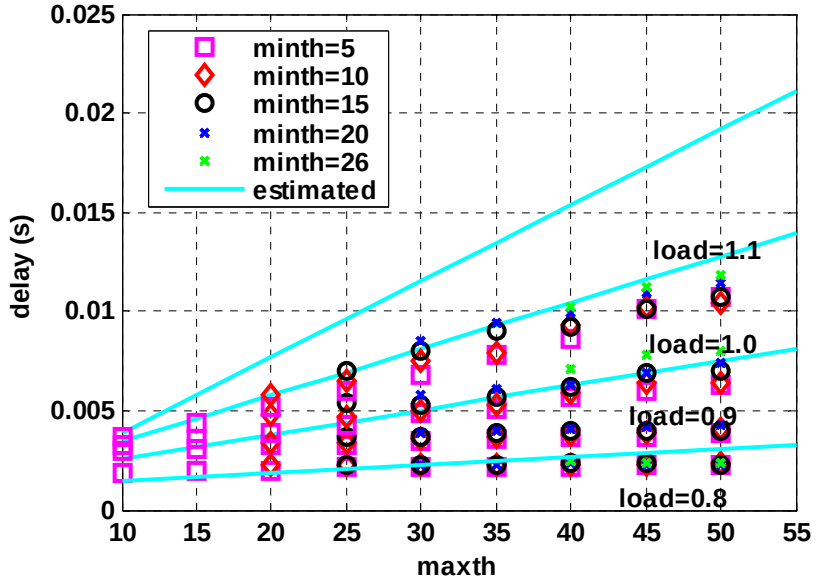

Figure 6-36 End-to-end delay of Scenario 4 under DiffServ

Figure 6-37 and Figure 6-38 give the corresponding network loss rates for scenario 1 and scenario 4. Table 6-4 gives a clearer view of the bound for the loss rate. It is

obvious that the loss rate for scenario 1 and 4 is within the bound summarized in Table 5-3.

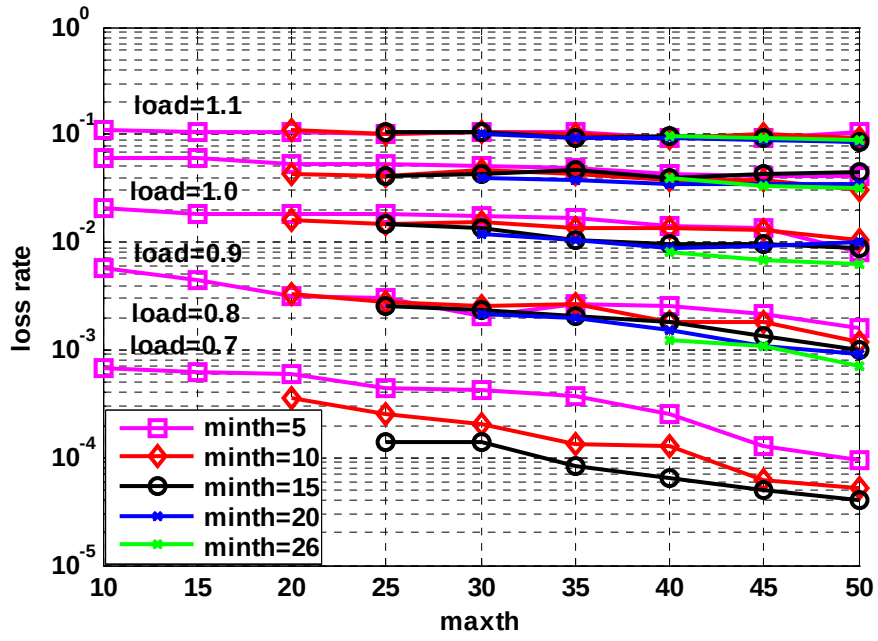

Figure 6-37 Network loss rate of scenario 1 under DiffServ

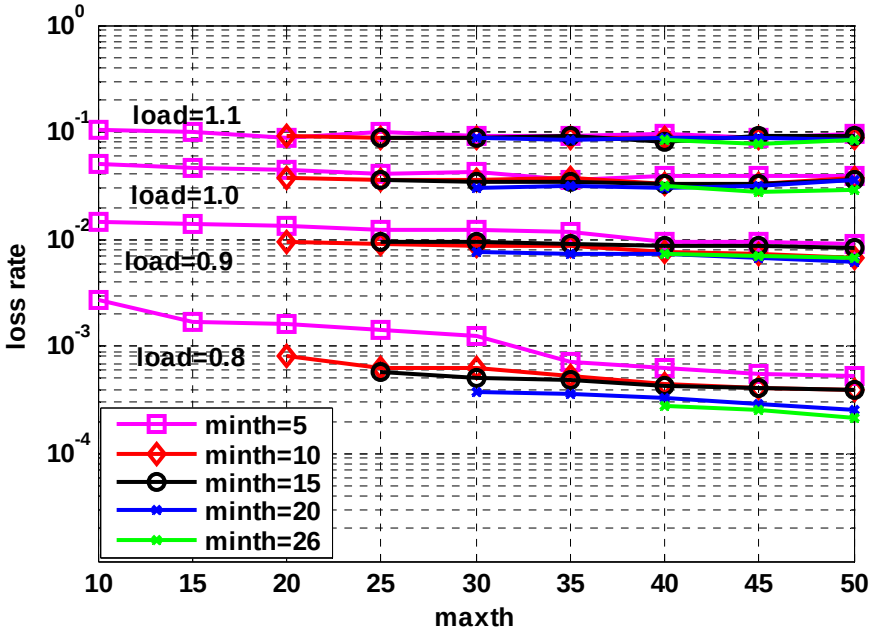

Figure 6-38 Network loss rate of scenario 4 under DiffServ

load	scenario 1	scenario 4	Guideline bound
0.8	<0.01	<0.01	<0.01
0.9	<0.02	<0.02	<0.03
1.0	<0.06	<0.06	<0.07
1.1	Around 0.1	Around 0.1	Around 0.1

Table 6-4 Loss rate for scenario 1 and scenario 4 under DiffServ

6.4 Summary

In chapter 4, we discussed RED's performance improvements for VoIP traffic over a single bottle-neck topology in terms of delay, jitter and loss under loaded and unloaded networks. In this current chapter we have investigated whether these improvements can be extended to more realistic scenarios with multiple hops or multiple classes.

First, a study about RED end-to-end performance control for VoIP traffic is given over a more realistic multiple hop topology. Through the end-to-end performance comparison between RED and drop-tail under different loads, we find that RED can control and adjust the end-to-end delay of voice traffic within an acceptable range under loaded or unloaded conditions. Through this control of delay, the corresponding jitter is bounded as well. Another key performance metric, the effective loss is decreased with respect to drop-tail by reducing the proportion of packets delivered too late to be of use. Also lost packets are distributed more randomly by the RED early drop mechanism thus decreasing consecutive loss. In addition, the proposed configuration guideline for voice traffic over RED is verified to be workable over this multiple hop topology.

Secondly, the performance of video traffic with the RED algorithm is studied. Different from voice traffic, the performance of video traffic is less bursty when the link is not overloaded. Once the link with drop-tail is overloaded, its performance will deteriorate to an unacceptable level. With RED the delay and jitter (and hence late delivery) can be controlled and thus provide a stable performance and graceful

degradation for video traffic whatever the load.

Lastly, RED's effect on VoIP is studied over a more complex architecture: DiffServ. Simulation results show RED can provide good performance control in terms of delay, jitter and loss under this DiffServ environment. In addition, the configuration guideline for VoIP traffic proposed in Chapter 5 is verified in DiffServ as well, indicating the possibility of wider applications in multi-class IP network infrastructure.

Chapter 7 Conclusions and Future Work

7.1 Conclusions

In recent years, real-time audio and video applications over IP networks, e.g. IP telephony, teleconferencing, have grown enormously. These applications enable efficient communications through IP networks. For example, VoIP is similar to the traditional circuit-switched telephone service, but can provide local and long distance telephone service at very low cost. It can also facilitate the deployment of new services which are not easily supported by the traditional circuit-switched networks, such as videoconferencing, which enables a face to face meeting between groups of people at two or more different locations through both speech and sight.

While different from the traditional applications, these real-time audio/video applications are sensitive to end-to-end delay and delay variation, but can tolerate occasional loss. But IP networks today only provide a best-effort service, and such a service does not make any guarantees for delay, jitter or loss of individual packets. Hence it is a challenge to develop successful real-time audio/video networking applications over best effort IP networks. In addition, the quality of real-time traffic will deteriorate to unacceptable levels when it is delivered over a moderately congested link, and this feature conflicts with the goal of fully utilizing the bandwidth in order to save cost. In spite of these challenges, the low cost of services and potential flexible applications for real-time traffic over IP network still attract the attentions of network service providers, and this has motivated our research work.

In recent years there have been numerous attempts to introduce QoS into IP networks, although most have failed more because of economic and legacy reasons than because of technical reasons [Kur05]. So a preferable solution should be with low cost and low complexity. This thesis presents a method of improving the QoS of real-time audio/video over IP network infrastructure at low service cost and making use of existing widely deployed mechanisms.

Random Early Detection (RED) [Flo93], a widely studied queue management mechanism, can help control the average queue size and hence the queuing delay within a network whether the transport protocol is cooperative or uncooperative. This delay control is extremely helpful for delay sensitive real-time traffic. However, RED was widely investigated in the context of TCP congestion collapse and synchronization issues, and the application of RED to UDP traffic has not been studied until now.

We present a thorough study of RED's application to VoIP traffic to improve its QoS in terms of delay, jitter and loss (both effective loss and consecutive loss). First, extensive simulation results show that the RED mechanism is able to regulate both packet-scale and burst-scale decay rates of queuing behaviour for inelastic voice traffic with real-time QoS requirements. This is especially important for the burst scale component, which dominates at larger queue sizes. Reducing the burst scale component can help to decrease the probability of the queue being large and hence lowers both the mean queue size and then mean queuing delay. Secondly, a detailed evaluation is given of how RED acts as a decay rate filter to control the delay, jitter and loss. This study indicates that by reducing the probability of larger queue sizes, RED keeps the mean queuing delay within an acceptable range, and maintains the jitter at a stable level. This delay control can also help to reduce the probability of late packet delivery and hence the total effective loss rate. In addition, consecutive loss, which is harmful to voice traffic and more difficult to recover from when using loss recovery schemes, is decreased with RED due to its randomized early drop behaviour. So extensive simulations have verified that through RED's burst scale decay rate filter behaviour, the performance of VoIP is improved in terms of delay, jitter and loss (both effective loss and consecutive loss).

Having demonstrated that RED is able to improve the QoS of voice traffic, applying it effectively requires some guidelines for configuring RED parameters to achieve a satisfactory performance. So a RED configuration guideline for voice traffic is proposed. This guideline is derived from extensive simulation results, takes into account the targeted performance requirements of voice traffic and the applied network load. Because the average queue size is determined by the queue state distribution, burstier traffic causes a longer queue state tail thus resulting in a longer

AQS. For a fixed buffer size, burstier traffic suffers higher loss due to the fact that it needs more buffer space to hold its longer queue state tail. So based on this analysis for the relationships between buffer size and traffic characteristic, the most bursty traffic was used to derive bounds for the worst case AQS and loss rate. The simulations demonstrated that the AQS of voice traffic increases with RED's maximum threshold, and forms an approximately linear relationship, whose gradient depends on the load. So through linear regression, the corresponding functions which describe the relationship between AQS and maximum thresholds under different loads were derived. These functions describe the AQS for the most bursty voice traffic, and so can be used as a bound for other less bursty traffic as well. As for loss rate, simulations show that the loss rate decreases as the maximum threshold increases, but this change in loss rate is not as obvious as the increase in AQS. The loss rate, for any given load, is not affected greatly by the maximum threshold. Thus the loss rate of the most bursty voice traffic under a small maximum threshold can be used as the bound for other less bursty traffic. This estimate of loss rate is not as accurate as for the AQS, but because VoIP traffic is more tolerant of loss, this rough estimate is acceptable.

This predicted bound for AQS and loss rate with VoIP traffic under RED will help network service providers to predict the worst case loss rate and delay that the voice traffic will experience per hop given specific RED parameter values and hence the end-to-end performance. If the required QoS cannot be met, the end-to-end performance can be adjusted by changing the admissible load and maximum threshold.

Thus the estimated bound will help service providers to set appropriate RED parameter values within the routers based on target delay and loss requirements for voice traffic. This can be extended to provide appropriate service for voice traffic at different levels, e.g. gold, silver and bronze service. A service provider can allocate voice traffic with different QoS requirements to different classes of queues configured with different RED parameter values. For example, the gold level voice traffic experiences RED queues with smaller maximum thresholds and lower applied load in order to benefit from lower delay and loss.

In addition, the configuration guideline can help to increase bandwidth utilization. Service providers always want to achieve a high utilization, i.e. approaching 1.0, but need to maintain QoS commitments. Based on the configuration guidelines, a service provider can select the highest load under which the delay and loss is acceptable.

Discussion about the configuration guideline's extension to the link with higher bit-rate is also provided. Since the resulting AQS over a link with a higher bit-rate than T1 link is shorter (e.g. over an E1 link), so the guidelines derived for a T1 link continue to apply over these links. Actually T1 link is the lowest link speed worth considering, and if we scale up link bit-rate and load (to keep the normalized load constant, e.g. at 0.8), then the burst scale decay rate, and hence AQS decreases (and the loss decreases, too). Hence the guidelines derived for the T1 link will continue to apply over other higher bit-rate links, but will just not be such a tight bound.

The performance evaluation of VoIP over a single bottleneck topology is also extended to the end-to-end performance over a realistic multiple hop scenario. Foreground traffic traverses a congested network path that consists of 4 hops before reaching the receiving end, and cross traffic is generated at each intermediate router from background traffic sources. This background traffic competes for the resource with the foreground traffic within one router queue and then exits the path. Via extensive simulations, our results demonstrate that, similar to the single bottleneck topology, the end-to-end delay distribution of VoIP over multiple hops under RED is well controlled within an acceptable range. Moreover, this limit on the end-to-end delay distribution helps to maintain stable jitter and decreases loss due to late delivery. In addition, this end-to-end performance study verifies that the configuration bounds for delay and loss are workable across multiple hops.

To verify that RED's control of VoIP is flexible and compatible and that the proposed bound works under DiffServ, RED was also investigated in a DiffServ architecture with WFQ scheduling discipline to allocate the link bandwidth to different classes of traffic. VoIP traffic with distinctly different characteristics (peak rate and packet size) was allocated to different queues with different RED thresholds. Measurement of the end-to-end delay and loss rate of these different

VoIP sources confirmed that the proposed configuration guidelines can also be applied to the end-to-end performance of VoIP under DiffServ.

The use of RED in managing VoIP traffic can control delay and jitter, and minimize the loss whatever the load. When the link is congested and overloaded, the control of delay, jitter and effective loss is especially obvious when compared with drop-tail. This provides a good solution to quality degradation of real-time traffic under congested network conditions.

Interactive video traffic has similar QoS requirements to VoIP, but has distinctly different traffic characteristics, such as larger and variable packet size and continuously variable bit rate instead of on/off patterns. So RED's performance control behaves in subtly different ways. When the link is not overloaded, the performance of video traffic is stable. However, once a drop-tail link is overloaded, the delay of video traffic becomes very long, resulting in a high probability of late delivery and hence effective loss. The QoS of video traffic with such long delays deteriorates rapidly. Applying the RED algorithm can provide a strong control for the delay through its preset thresholds, thus decreasing the probability of late delivery as well. Simulations indicate that an appropriately configured RED algorithm benefits video traffic under overloaded situations by greatly improving the delay and effective loss performance, thus providing a good solution to quality degradation of video traffic under congested network conditions.

7.2 Future work

This thorough evaluation of RED's performance improvements for VoIP can be considered as the first successful step in the whole development of improving QoS by active queue management to inelastic UDP traffic. There are many ways to extend this application further, and some of these possibilities are reviewed in this section.

The VoIP models used in the analysis and simulations are modeled by the conventional multiplexing exponential on/off sources, and this exponential model can be used for a first hand performance estimate. As mentioned in [Chua02], these classical on/off models describe the digitized voice in two-way telephone

conversation very well, but for other newly emerging multimedia applications, e.g. distance learning, multimedia conferencing and interactive games, a newly proposed Weibull model is a better matching statistical model and achieves lower error than the conventional exponential model. Due to the fact that these multimedia applications have similar delay-sensitive requirements, the benefits gained by controlling queue behaviour with the RED algorithm can help to improve the QoS of these multimedia applications. So a study about RED's potential performance control for traffic represented by this Weibull model (and hence other multimedia applications) would be a valuable extension. This can widen RED's application into more real-time traffic scenarios rather than voice traffic only. A future study about the Weibull model for interactive traffic and its behaviour with the RED algorithm can provide the performance analysis (in terms of delay, jitter and loss) for other packet audio streams over IP networks.

In addition, when these per-hop behaviours are concatenated into source-destination paths across a more realistic network topology (which may consist of multiple core routers and edge routers), VoIP's end-to-end performance improvement by RED can be explored. At present, end-to-end QoS solutions mainly involve greater complexity, (e.g. searching out alternative paths that currently meet the specified QoS constraints), while a combination of the RED algorithm, class based schedulers and OSPF routing could obtain not only service capacity, but also statistical bounds on delay and jitter performance for VoIP traffic. This could lead to integrating QoS into the default shortest paths by configuring the hops to meet predefined statistical delay and loss bounds based on the configuration guideline described in this thesis. Working together with admission control at the edge of the domain, the per-hop bounds for delay and loss under a certain maximum load will ensure that the end-to-end performance of delay and loss along the shortest path can be predicted.

With these two steps complete, the QoS improvement for VoIP and other multimedia traffic could be verified to be robust enough to apply in reality, thus greatly increasing the QoS of real-time traffic over IP networks with low cost and little added complexity.

Appendix

A: Decay Rate Analysis

Here is a brief introduction of the excess-rate (ER) method for decay rate analysis as used in section 3.2.4. The ER theory [Pit01] gives a more accurate formula for computing the decay rate of burst scale queuing, and ER theory can be briefly outlined as follows:

The following symbols are used:

Ton = mean duration of the ON time periods of a single on/off source

Toff = mean duration of the OFF time periods of a single on/off source

α = activity factor i.e. Ton / (Ton+Toff)

R = packet arrival rate of the single on/off source during the ON period

C = bandwidth allocated to these aggregate on/off sources

In ER Analysis, N on/off sources are parameterized into a single equivalent on/off source with the parameters $T_{(on)}$, $T_{(off)}$, Ron and Roff as shown below. The definitions of these parameters are as follows:

$T_{(on)}$ = mean ON time of the equivalent single on/off source

$T_{(off)}$ = mean OFF time of the equivalent single on/off source

R_{on} = transmission rate of the equivalent single on/off source during ON period

R_{off}= transmission rate of the equivalent single on/off source during OFF period

The ON and OFF time of the single equivalent on/off source are still assumed to be accurately modeled by a negative exponential distribution as described in section 3.1.1. During the ON period of the equivalent single aggregated source, the transmission rate Ron is greater than the service rate. Hence, the queue is built up and

causes burst-scale queuing. During an OFF period, the transmission rate Roff is smaller than the service rate, and the queue built up during ON periods will be gradually reduced.

The calculation of these parameters of a single equivalent aggregate on/off source can be found in [Pit01], and the formulae are listed in Table A:

i. $T_{(ON)} = \frac{R \cdot T_{ON}}{C - A_p}$	mean ON time (aggregate model)
ii. $T_{(OFF)} = T_{(ON)} \frac{1-D}{D}$	mean OFF time (aggregate model)
iii. $R_{ON} = C + R \cdot \frac{A_p}{C - A_p}$	mean rate in the ON state (aggregate model)
iv. $R_{OFF} = \frac{A_p - D \cdot R_{ON}}{1-D}$	mean rate in the OFF state (aggregate model)
v. $A_p = F \cdot T_{ON} \cdot R$	mean load in packets/sec
vi. $F = \frac{N}{T_{ON} + T_{OFF}}$	the rate of flow arrivals
vii. $A = F \cdot T_{ON}$	the offered traffic in Erlangs
viii. $N_o = \frac{C}{R}$	minimum number of active sources for burst-scale queuing
ix. $B = \frac{\frac{A^{N0}}{N0!}}{\sum_{r=0}^{N0} \frac{A^r}{r!}}$	

$D = \frac{\left\{\frac{A^{N_o}}{N_o!}\cdot\left(\frac{N_o}{N_o - A}\right)\right\}}{\left\{\sum_{r=0}^{N_o-1}\frac{A^r}{r!} + \frac{A^{N_o}}{N_o!}\left(\frac{N_o}{N_o - A}\right)\right\}}$	the probability a flow is delayed
$p_{er} = \frac{h \cdot D}{C - A_p}$	Probability that a packet is excess rate arrival
$a = 1 - \frac{1}{T_{(on)}(R_{on} - C)}$	
$s = 1 - \frac{1}{T_{(off)}(C - R_{off})}$	
$\eta_b = \frac{a}{s}$	Decay rate of burst scale

Table A Formula for burst scale decay rate

B: Calculated Decay Rate

	Calculated Packet-scale decay rate	Calculated Burst-scale decay rate
RED (1, 2)	0.6177	None
RED (1, 3)	0.6206	0.7541
RED (1, 5)	0.6215	0.8415
RED (1, 10)	0.6278	0.8703
RED (5, 10)	0.6321	0.9015
RED (5, 20)	0.6331	0.9251
RED (26, 50)	0.6434	0.9503
Drop Tail	0.6514	0.9760
Theoretical	0.6587	0.9798

Table B Calculated decay rate (load=0.8, voice model 1)

C: Linear Regression

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. The most common method for fitting a regression line is the method of least-squares. This method calculates the best-fitting line for the observed data by minimizing the sum of the squares of the vertical deviations from each data point to the line. The technique of least squares curve fitting can be easily be implemented in MATLAB. As introduced in [Pra02], most of the curve fits we do are either polynomial curve fits or exponential curve fits. If we want to fit a polynomial of order n through our data, we can use the MATLAB's built-in function *polyfit* which performs a least squares curve fit. If we want to fit a non-polynomial function, the two most commonly used functions are $y=ae^{bx}$ or $y=cx^{d}$, we can convert these exponential curve fits into polynomial curve fits (actually a linear one) by taking log of both sides of the equations, i.e.

1. $\ln(y) = \ln(a) + bx$ or $\bar{y} = a_0 + a_1 x$, where $\bar{y} = \ln(y), a_1 = b, a_0 = \ln(a)$

2. $\ln(y) = \ln(c) + d\ln(x)$ or $\bar{y} = a_0 + a_1 \bar{x}$, where $\bar{y} = \ln(y), \bar{x} = \ln(x)$, a_1=d, and a_0=ln(c)

Now we can use *polyfit* in both cases with just first order polynomials to determine the unknown constants, the steps involved are the following.

Step-1: Prepare new data: prepare new data vectors $\bar{y}, \bar{x}$ by taking the ln of the original data.

Step-2: Do a linear fit: Use polyfit to find the coefficients a_0 and a_1 for a linear curve fit.

Step-3: Plot the curve: From the curve fit coefficients, calculate the values of the original constants (e.g. a, b, etc.). Recompute the values of y at the given bin's according to the relationship obtained and plot the curve along with the original data.

D: End-to-end Delay Distributions under DiffServ

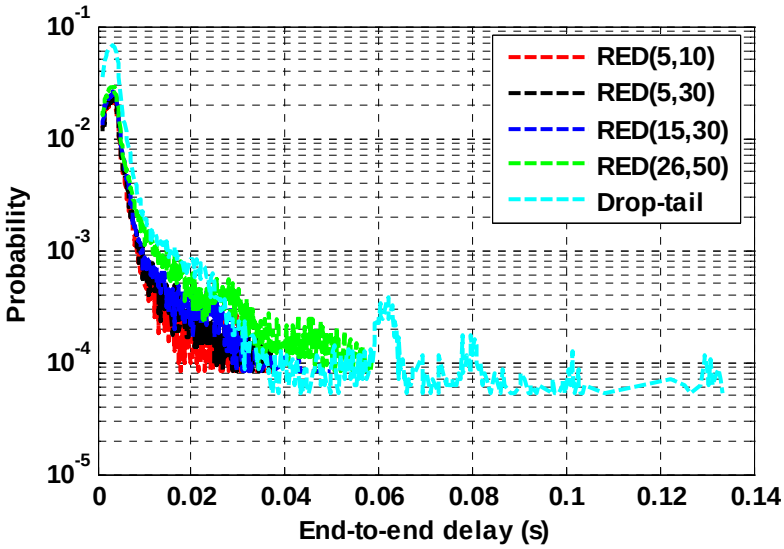

Figure D-1 End-to-end delay distribution for class 1 (load=0.8)

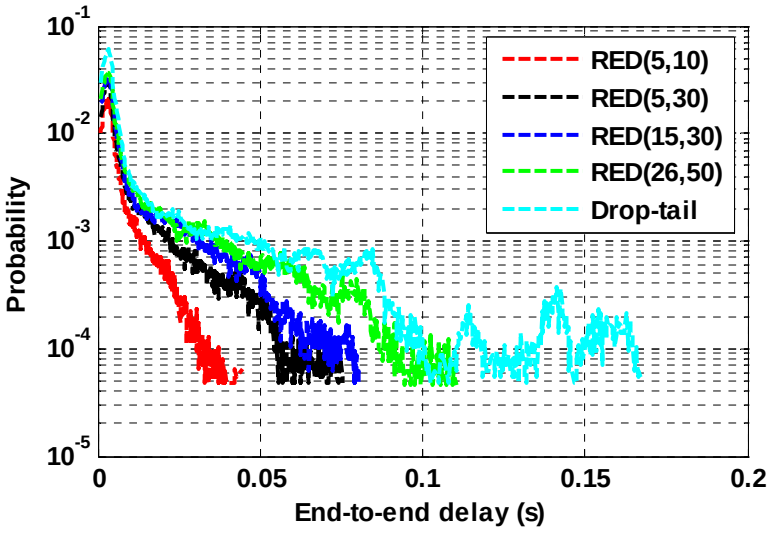

Figure D-2 End-to-end delay distribution for class 1 (load=0.9)

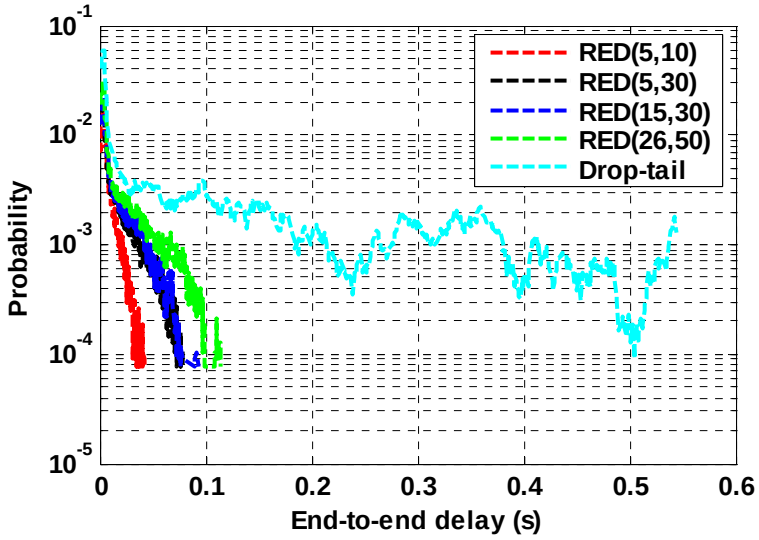

Figure D-3 End-to-end delay distribution for class 1 (load=1.0)

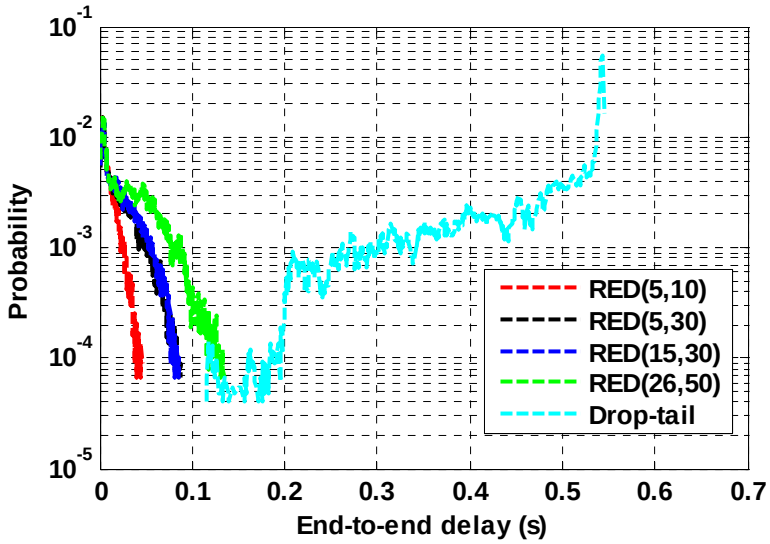

Figure D-4 End-to-end delay distribution for class 1 (load=1.1)

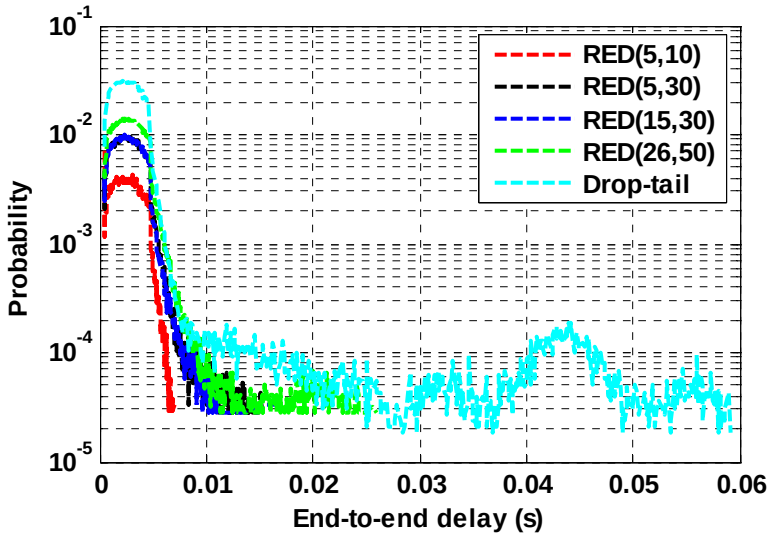

Figure D-5 End-to-end delay distribution for class 2 (load=0.8)

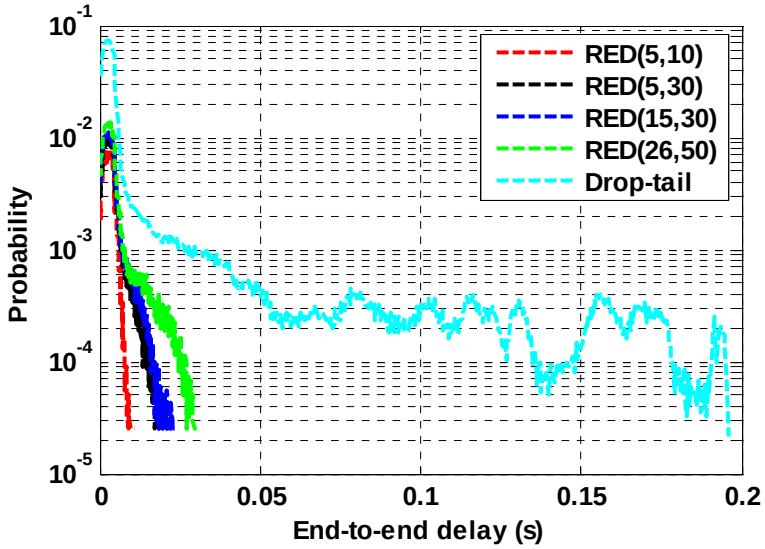

Figure D-6 End-to-end delay distribution for class 2 (load=0.9)

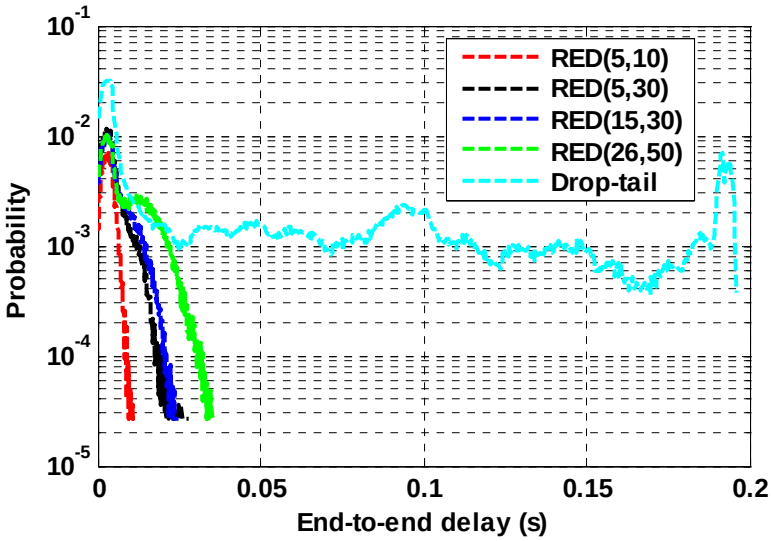

Figure D-7 End-to-end delay distribution for class 2 (load=1.0)

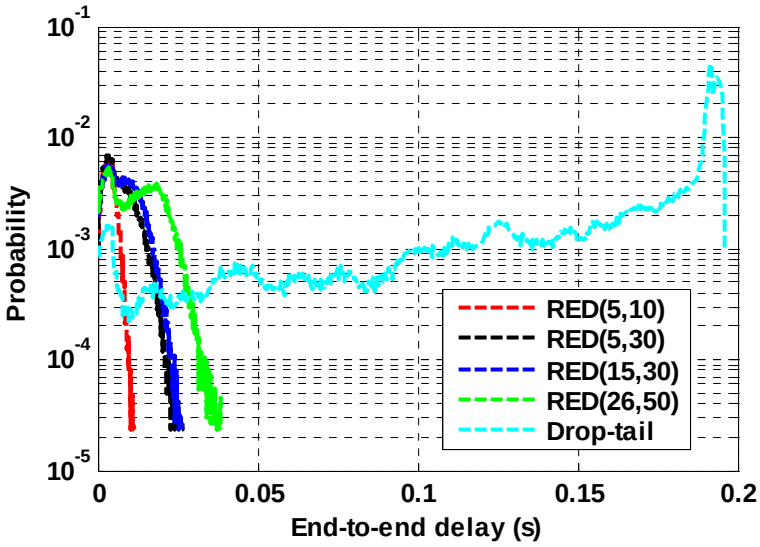

Figure D-8 End-to-end delay distribution for class 2 (load=1.1)

E: Throughputs under Different Loads and RED Thresholds

Max_{th}\load	0.7	0.8	0.9	1.0	1.1
Offered load	1.0808	1.235	1.389	1.544	1.698
Drop-tail	1.085	1.232	1.383	1.528	1.543
50	1.084	1.230	1.366	1.458	1.507
45	1.083	1.229	1.362	1.453	1.505
40	1.082	1.228	1.358	1.450	1.503
35	1.081	1.228	1.355	1.448	1.499
30	1.081	1.227	1.354	1.445	1.497
25	1.080	1.227	1.354	1.444	1.496
20	1.080	1.227	1.353	1.442	1.494

Table E-1 Throughput (in Mbps) for min_{th}=10 under different loads and max_{th}

Max_{th}\load	0.7	0.8	0.9	1.0	1.1
Offered load	1.0808	1.235	1.389	1.544	1.698
Drop-tail	1.085	1.232	1.383	1.5283	1.543
50	1.082	1.231	1.365	1.457	1.512
45	1.080	1.231	1.363	1.455	1.510
40	1.079	1.230	1.362	1.453	1.504
35	1.078	1.229	1.361	1.452	1.501
30	1.078	1.229	1.361	1.449	1.498

Table E-2 Throughput (in Mbps) for min_{th}=20 under different loads and max_{th}

Max_{th}\load	0.7	0.8	0.9	1.0	1.1
Offered load	1.0808	1.235	1.389	1.544	1.698
Drop-tail	1.085	1.232	1.383	1.5283	1.543
50	1.084	1.231	1.374	1.465	1.511
45	1.083	1.230	1.371	1.463	1.510
40	1.082	1.230	1.370	1.462	1.509

Table E-3 Throughput (in Mbps) for min_{th}=26 under different loads and max_{th}

Reference

[Aba95] J. Abate, G. L. Choudhury, W. Whitt, “Exponential Approximations for Tail Probabilities in Queues, I: Waiting Times”, Operations Research, vol. 43, pp. 885-901, 1995

[Ala05] O. Alanen, M. Paakkonen, M. Ketola, T. Hamalainen, J. Joutsensalo, “Enhanced Admission Control Solution for Multicasting with Diffserv”, Next Generation Internet Networks, pp.268 – 273, 18-20 Apr. 2005

[Ali04] R. B. Ali, S. Pierre, Y. Lemieux, “Diffserv Qos Performance Evaluation of Multimedia Telephony”, Electrical and Computer Engineering, Canadian Conference, vol. 4, pp. 2115 - 2118, 2-5 May. 2004

[Bai91] A. Baiocchi, N. Melazzi, M. Listanti, A. Roveri, R. Winkler, “Loss performance analysis of an ATM multiplexer loaded with high-speed ON-OFF sources,” IEEE Journal on Selected Areas in Communications, vo1.9, no.3, pp. 388-393, Apr. 1991.

[Bol93] J. C. Bolot, “End-to-End packet delay and loss behavior in the Internet”, Proc. of ACM SIGCOMM’93, pp. 289-298, 1993

[Bon00] T. Bonald, M. May, “Analytic evaluation of RED performance”, IEEE INFOCOM 2000, vol. 3, pp 1415 - 1424, 26-30 Mar. 2000

[Bor98] M. S. Borella, D. Swider, S. Uludag, and G. B. Brewster, “Internet Packet Loss: Measurement and Implications for End-to-End QoS,” Proc. of International Conference on Parallel Processing, Aug. 1998

[Bou02] C. Bouras, Dimitrios Primpas, Afrodite Sevasti and Andreas Varnavas, “Enhancing the diffserv architecture of a simulation environment”, Distributed Simulation and Real-Time Applications, Proc. Sixth IEEE International Workshop, pp.108 – 115, 11-13 Oct. 2002

[Bra69] P. T. Brady “A Model for Generating ON-OFF Speech Patterns in Two-Way

Conversations". Bell System Technology Journal, vol.48, pp. 2445-2472, Sep. 1969

[Cha00] C. S. Chang, "Performance guarantees in communication networks", Springer-Verlag, 2000

[Chun00] J. Chung, M. Claypool, "Dynamic-CBT and chips-Router support for improved multimedia performance on the Internet", Proc. of ACM Multimedia Conference, Nov. 2000

[Chua01] C. N. Chuah, "A Scalable Framework for IP-Network Resource Provisioning Through Aggregation and Hierachical Control", PhD Thesis, University of California at Berkeley, 2001

[Chua02] C. N. Chuah, "Characterizing Packet Audio Streams from Internet Multimedia Applications", Proc. of International Communications Conference, New York, 2002

[Cis01] "Traffic analysis for voice over IP", http://www.cisco.com/univercd/cc/td/doc/cisintwk/intsolns/voipsol/ta_isd.pdf, 2001

[Den95] S. Deng "Traffic Characteristics of Packet Voice". IEEE International Conference on Communications, vol.3, pp. 1369-1374, 1995

[Dra00] A. Dracinschi, S. Fdida, "Efficient congestion avoidance mechanism", Local Computer Networks, 25th Annual IEEE Conference, pp.228-237, Nov. 2000

[Duf98] N. Duffield, "Applications of Large Deviations to performance analysis with long-range dependent traffic", Workshop on stochastic modeling and analysis of communication networks, 1998

[Flo93] S. Floyd , V. Jacobson, "Random early detection gateways for congestion avoidance", Networking, IEEE/ACM Transactions, vol.1 no.4, pp.397-413, Aug. 1993

[Flo97] S. Floyd, "RED: Discussions of Setting Parameters", http://www.icir.org/floyd/red.html, November 1997

[Flo02] S. Floyd, E. Kohler, "Internet research needs better models",

http://www.icir.org/models/, 2002

[Fit01]F. H. P. Fitzek, M. Reisslein, “MPEG-4 and H.263 video traces for network performance evaluation”, IEEE Network Magazine, vol.15, no.6, pp.44-54, Nov. 2001

[Fur01] B. Furht., S. W. Smoliar, “Video and Image Processing in Multimedia Systems”, Springer, 2001

[Gao00] J. B. Gao, “Multiplicative Multifractal Modeling of Long-Range-Dependent (LRD) Traffic in Computer Communications Networks”, PhD thesis, University of California, 2000

[Gev01] P. Gevros, J. Crowcroft, P. Kirstein, S. Bhatti, “Congestion control mechanisms and the best effort service model”, Network, IEEE, vol. 5, no.3, pp.16 – 26, May-June 2001

[Han94] W. Y. Han, S. J. Lee, C. M. Han, S. H. Kim, “Queuing analysis for an ATM multiplexer loaded by CBR and On/Off traffic sources”, Singapore International Conference on Communication Systems ’94, pp. 760-764, November 1994

[Har01] W. C. Hardy, “QoS Measurement and Evaluation of Telecommunications Quality of Service”, John Wiley & Sons, ISBN 0-471-49957-9, 2001

[Has00] M. Hassan, A. Nayandoro, M. Atiquzzaman, “Internet telephony: services, technical challenges, and products”, Communications Magazine, IEEE, vol.38, no.4, pp.96-103, Apr. 2000

[Hef86] H. Heffes, D. M. Lucantoni, “A Markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance”, IEEE Journal on Selected Areas in Communications, vol.4, no.6, pp.856-868, Sep. 1986

[Hol01] C. V. Hollot, V. Misra, D. Towsley, W. B. Gong, “A control theoretic analysis of RED”, Proc. of IEEE INFOCOM, pp. 1510-1519, Apr. 2001

[Hus00] G. Huston, “Internet Performance Survival Guide: QoS strategies for

multiservice networks", John Wiley & Sons, Inc, 2000

[ITU P.59] International Telecommunication Union, Telephone transmission quality objective measuring apparatus: Artificial conversation speech. Recommendation P.59, Telecommunication Standardization Sector of ITU, Geneva, Mar. 1993

[ITU G.114] ITU-T Recommendation G.114. General Characteristics of International Telephone Connections and International Telephone Circuits: One-Way Transmission Time, Feb. 1998

[Jel95] P.R. Jelenkovic, A.A. Lazar, "On the dependence of the queue tail distribution on multiple time scales of ATM multiplexers", Proc. of the 29th Annual Conference on Information Sciences and Systems, Baltimore, pp.746-752, Mar. 1995

[Kar00] M. Karam, F. Tobagi, "On Traffic Types and Service Classes in the Internet", Proc. of GLOBCOM'2000, Dec. 2000

[Kar01] M. Karam, F. Tobagi, "Analysis of the delay and jitter of voice traffic over the Internet", Proc. of IEEE INFOCOM, vol.2, pp.824--833, 2001

[Kur88] J. F. Kurose, H. T. Mouftah, "Computer-Aided Modeling, Analysis, and Design of Communication Networks", IEEE Journal on Selected Areas in Communications, vol.6, no.1, Jan. 1988

[Kur05] J. F. Kurose, K. W. Ross, "Computer networking: a top-down approach featuring the Internet", third edition, Pearson Education, 2005

[Kuz01] A. Kuzmanovic, E. Knightly, "Measuring Service in Multi-Class Networks", Proc. of IEEE INFOCOM'01, Apr. 2001

[Leu03] C. M. Leung, J. A. Schormancs, J. M. Pitts, A. Woolf, "Accurate decay rate prediction for burst-scale queuing in packet buffers", Electronic Letters, vol.39, no.2, pp.253-254, 23 Jan. 2003

[Liu02] E. Liu, "A Hybrid Queuing Model for Fast Broadband Networking Simulation", PhD Thesis, University of London, 2002

[Lib] video trace library, http://www.tkn.tu-berlin.de/research/trace/trace.html

[Meh01] P. Mehta, and S. Udani, “Voice over IP”, Potentials, IEEE, vol.20, no.4, pp.36 – 40, Oct-Nov. 2001

[Men02] F. Menta, G. Schembra, “Efficient design of RED routers for TCP/RAP fairness optimization”, ICC 2002, IEEE International Conference on Communications, vol.4, pp.2186-2190, 28 April-2 May 2002

[Mic97] H. Michiel, K. Laevens, “Teletraffic engineering in a broad-band era”, Proc. of the IEEE, vol. 85, no.12, pp. 2007-2033, Dec. 1997

[Min98] D. Minoli, and E. Minoli “Delivering Voice over IP Networks”, Wiley, 1998

[Nag91] R. Nagarajan, J. F. Kurose, D. Towsley, “Approximation Techniques for Computing Packet Loss in Finite-Buffered Voice Multiplexers”, IEEE Journal on Selected Areas in Communications, vol.9.no.3, pp.368-377, Apr. 1991

[Nor91] I. Norros, J. W. Roberts, A. Simonian, J. T. Virtamo, “The Superposition of Variable Bit Rate Sources in an ATM Multiplexer”, IEEE Journal on Selected Areas in Communications, vol.9, no.3, Apr. 1991

[Per98] C. Perkins, O. Hodson, V. Hardman, “A survey of packet loss recovery techniques for streaming audio”, IEEE Network Magazine, vol.12, no.5, pp.40-47, Sep/Oct. 1998

[Pit01] J. M. Pitts, J. A. Schormans, “Introduction to IP and ATM design and performance”, Wiley, 2001

[Pit06] J.M. Pitts, X. Wang, Q. Yang, J. A. Schormans, “Excess-rate queuing theory for M/M/1/RED with application to VoIP QoS”, Electronic Letters, vol.42, no.20, 28 Sep. 2006

[Pra02] R. Pratap, “Getting started with MATLAB”, Oxford University Press, 2002

[Ram94] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, “Adaptive playout mechanisms for packetized audio applications in wide-area networks”, IEEE INFORCOM, Toronto, Canada, June 1994

[Rib00] V. Ribeiro, M. Coates, R. Riedi, S. Sarvotham, B. Hendricks, R. Baraniuk,

"Multifractal Cross-Traffic Estimation", Proc. ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management, Monterey, CA, Sep. 2000

[Riz97] L. Rizzo. "Effective erasure codes for reliable computer communication protocols", ACM Computer Communication Review, vol. 27, no2, Apr. 1997

[Rob91] J. W. Roberts, "Variable-Bit-Rate Traffic Control in B-ISDN", IEEE Communications Magazine, vol.29, no.9, pp.50-56, Sep. 1991

[RFC1633] R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview", Jun. 1994

[RFC1889] H. Schulzrinne, S.Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", Jan. 1996

[RGC2198] C. E. Perkins, I. Kouvelas, O. Hodson, V. J. Hardman, M. Handley, J. C.Bolot, A. Vega-Garcia, and S. Fosse-Parisis, "RTP payload for redundant audio data", Internet Engineering Task Force, Sep. 1997

[RFC2309] B. Braden etc., "Recommendations on Queue Management and Congestion Avoidance in the Internet", Apr. 1998

[RFC2474] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", Dec. 1998

[RFC2475] S. Black, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services", Dec. 1998

[RFC2733] J. Rosenberg and H. Schulzrinne, "An RTP payload format for generic forward error correction", RFC2733, Dec. 1999

[RFC3261] J. Rosenberg, etc. "SIP: Session Initiation Protocol", June 2002

[RFC3550] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", July 2003

[Sch96] J. A. Schormans, J. M. Pitts, B. R. Clements, E.M. Sharf, "Approximation to M/D/1 for ATM CAC, buffer dimensioning and cell loss performance", Electronic

Letters, vol.32, no.3, pp.164-165, Feb. 1996

[See04] P. Seeling, M. Reisslein, B. Kulapala, “Network performance evaluation using frame size and quality traces of single-layer and two-layer video: a tutorial”, IEEE Communications Surveys & Tutorials, Third Quarter, 2004

[Sri86] K. Sriram and W. Whitt, “Characterizing Superposition arrival processes in packet multiplexers for voice and data”, IEEE Journal on Selected Areas in Communication, vol.4, no.6, pp.833-846, Sep. 1986

[Ste02] R. A. Stewart, “End-to-end delay analysis for small/medium scale IP networks”, PhD thesis, University of London, 2002

[Sun04] L. Sun, “Speech quality prediction for voice over Internet protocol networks”, PhD thesis, University of Plymouth, 2004

[Xu05] Y. Xu and R. Guerin, “Individual QoS versus aggregate QoS: A loss performance study”, IEEE/ACM Transactions on Networking, vol.13, no.2, Apr. 2005

[Yat93] D. J. Yates, J. F. Kurose, D. Towsley, M. G. Hluchyj, “On Per-session End-to-end Delay Distributions and the Call Admission Problem for Real-time Applications with QOS Requirements”, ACM SIGCOMM Symposium on Communications Architectures and Protocols, pp 2-12, Sep. 1993

[Yil01] S. Yilmaz, I. Matta, “On class-based isolation of UDP, short-lived and long-lived TCP flows”, Proc. of the 9th Annual Meeting of the IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, pp.415-422, 15-18 Aug. 2001

[Zen04] X. Zeng, C.H. Lung, C. Huang, A. Srinivasan, “A bandwidth-efficient scheduler for MPLS DiffServ networks”, Proc. of the 12th IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, Volendam, the Netherlands, vol.4, no.8, pp.251-258, Oct. 2004

[Ziv02] A. Ziviani, J. F. de Rezende, O.C.M.B. Drarte, and S. Fdida, “Improving the

delivery quality of MPEG video streams by using differentiated services", Proc. of the 2nd European Conference on Multi-service Networks, pp.107-115, 2002