Performance Evaluation of Internet Telephony

Katsuyoshi Iida

Preface

Internet Telephony is a new communication infrastructure, which covers a diverse range of services. The network will convey voice, video, and data traffic in the same way, i.e., in a packet switching basis. This integration will enable new communication services and lead to cost-effective network management.

There are many advantages of Internet Telephony. First, since only one network is needed for all services, administrative cost of networks can be reduced. Second, voice codec with a high compression ratio can be used easily, so that bandwidth consumption can be reduced compared with the current telephone service. Next, there is a possibility of creating new services. For example, in case of teleconferencing, presentation related data can be sent. Furthermore, multiparty phone can be easily implemented by means of multicasting. Finally, new functions can be easily added because of software's flexibility.

The major objective of this dissertation is to provide a design principle of Internet Telephony Network. The principle is different from that of Public Switched Telephone Network (PSTN) and also different from that of the current Internet. Thus, theories for designing Internet Telephony Network are needed.

Moreover, Internet Telephony Network should provide a real-time communication service by means of packet switching, which has a poor ability of providing such a service. Here, Quality-of-Service (QoS) is an important term, which means communication quality provided by the network. PSTN can provide a QoS of voice communication because it uses circuit switching, which dedicates network resources (e.g., bandwidth or time slots) to each voice communication. On the other hand, Internet Telephony Network needs a different way of providing QoS.

The major QoS related parameters for voice communication are delay jitter, delay and packet loss. Delay jitter or variance of delay time can be reduced by Real-Time Protocol (RTP) on the Internet. In order to improve delay performance, priority scheduling, resource allocation and admission control mechanisms are needed. These methods are also useful for reducing the probability of packet dropping.

Thus, to implement Internet Telephony Network, we need RTP, priority scheduling, resource allocation and admission control methods. Currently, there are few theories to design Internet Telephony Network employing these methods. Therefore, I try to construct design principles of Internet Telephony Network for end-to-end communications in this dissertation.

First in Chapter 2, I focus on the whole of the Internet to give the big picture of end-to-end communications. My design architecture mainly consists of three types of networks, i.e., the domain networks, the access networks, and the stub networks. Among them, I point out issues arising in the former two networks for end-to-end QoS provisioning. First, the access networks cause large transmission delay because of small transmission capacity. On the other hand, the domain networks have relatively large transmission capacity, therefore transmission delay time is insignificant. However, in this kind of networks, a scalability problem in terms of the number of flows is crucial because such networks often accommodate a huge number of flows. In Chapter 2, I would like to emphasize that the Internet Telephony network should be divided into sub-networks for designing end-to-end QoS aware networks. Each sub-network has a different performance bottleneck for providing QoS.

Next, in Chapter 3 and 4, I see the access networks, in which the most crucial issue is large delay time of a packet transmission due to relatively small capacity of links. It is known that large delay time heavily degrades the QoS. Therefore, limiting the maximum number of voice flows, which can keep delay time under a value, becomes an important issue. To evaluate such delay time in access network, I construct two models of access networks; homogeneous and heterogeneous flows. The homogeneous flows mean that their packet transmission rates and their packet lengths are the constants. This represents a multiplexing of a number of voice flows of the same characteristic. In my model, there are a number of voice flows and also

data flows. Data flows are obstacles of voice flows. This is a preliminary but an important case, because widely spread dial-up access lines have very small capacity, which can accept only flows of small bandwidth. In the homogeneous case (Chapter 3), I obtain an exact and closed form solution to the delay performance, so that we do no need any approximations or numerical Laplace inversions. Schemes to reduce delay are proposed and their performance is evaluated.

Furthermore, in Chapter 4, the heterogeneous flow case treats three types of traffic; voice, video and data traffic. The model allows a multiplexing of different transmission rates and different packet lengths, e.g., accommodating both voice and video traffic. Similar to the homogeneous case, data flows are the obstacles to voice and video flows. My numerical results show the large packet length of video packets has a large influence on voice delay.

Next, in Chapter 5, I try to provide a design architecture of backbone networks. The backbone networks have a large transmission capacity, so that delay experienced by real-time packets, like voice and video, will be negligible because of its high speed packet forwarding capability. What is the bottleneck of backbone networks? That is state requirements in routers. Real-time flows request to allocate their dedicated bandwidth in an end-to-end region. Consequently, the network nodes (or routers) should maintain a number of states in terms of flow establishment, and are desired to exchange the signaling messages for allocating some dedicated bandwidth to the real-time flows. For example, a high-speed link of 10 Gb/s can accommodate one million voice flows of 10 kb/s. Consequently, the state requirements will be of major concern in future high-speed networks. Furthermore, call blocking probability, which describes a network utilization from user points of view, is also investigated. Through the numerical results, I examine tradeoffs between call blocking probability and the state requirements.

Acknowledgements

First and foremost, I wish to express my sincere appreciation to Professor Yuji Oie of Kyushu Institute of Technology for his careful advice and heartily supports on the whole of this research in a number of years. Without his guidance, none of this work has been accomplished. I also thank him for his carefully reading of this dissertation.

I would like to thank Professor Kyoki Imamura, Professor Hitoshi Fujii, and Professor Toshiaki Ejima of Kyushu Institute of Technology for their supports and invaluable comments. I am deeply indebted to Associate Professor Kenji Kawahara of Kyushu Institute of Technology, for his stimulating discussions and untiring supports. His steady supports always helped me.

I am also highly indebted to Associate Professor Tetsuya Takine of Kyoto University for his encouragements and invaluable advice, in particular, concerning analysis of communication systems. He supervised me with patient encouragements though my understanding of analytical part was too bad.

I am very grateful to Professor Hideki Sunahara of Nara Institute of Science and Technology for his encouragements and supports.

I would like to express my gratitude to Associate Professor Fumio Ishizaki of Nanzan University, Dr. Duan-Shin Lee of C&C Research Laboratories, NEC USA Inc., and Dr. Bhaskar Senguputa of C&C Research Lab. for their supporting the work in Chapter 4. I also highly appreciate Dr. Dirceu Cabendish of C&C Research Lab. for his insightful comments of the work in Chapter 2 and 3.

There are a number of individuals I wish to thank: Assistant Professor Takeshi Ikenaga of Kyushu Institute of Technology, Assistant Professor Yoshiaki Hori of Kyushu Institute of Design, Assistant Professor Manabu Kato of Ariake National College of Technology, Mr. Hiroyuki Koga, and the rest of the members of Oie's Laboratory and Kawahara's Laboratory for their kindly supports and suggestions.

Great thanks go to Professor Suguru Yamaguchi, Associate Professor Youki Kadobayashi, and Assistant Professor Takeshi Okuda of Nara Institute of Science and Technology for their encouragements and untiring supports. I thank the rest of members in Yamaguchi's Laboratory for their encouragements.

Finally, I would never have been able to finish this work without the support and understanding of my family. Many thanks goes to my wife, Sachiyo, for her supports and patients. The greatest appreciation goes to my parents for their constant support and understanding.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview of Internet Telephony

Internet Telephony is a new communication infrastructure, which covers a diverse range of services. The network will convey voice, video, and data traffic in the same way, i.e., in a packet switching basis. This integration will enable new communication services and lead to a cost-effective network management.

Recently, numbers of related vendors and service providers enter a market of Internet Telephony [Hel98, Dou00, HGP00, KR01]. Furthermore, there is a forecast that commercial telephone networks will be replaced by Internet Telephony instead of the current circuit switched network[KR01, p. ix]. Accordingly, some sort of voice applications (e.g., Internet Telephony) would be the next killer application to the world wide web[KR01, p. 486].

Here are advantages of Internet Telephony.

- Only one network is needed for all services.
  - Administrative cost of networks can be reduced.
- Voice codec with a high compression ratio can be used easily.
  - Bandwidth consumption can be reduced compared with the current telephone service.

- There is a possibility of creating future services.
  - In case of teleconferencing, presentation related data can be sent.
  - Multiparty phone can be easily implemented by means of multicasting.
- New functions can be easily added because of software's flexibility.

The major objective of this dissertation is to provide a design principle of Internet Telephony Network. The principle is different from that of Public Switched Telephone Network (PSTN) and also different from that of the current Internet. Thus, theories for designing Internet Telephony Network are needed.

Moreover, Internet Telephony Network should provide a real-time communication service by means of packet switching, which has a poor ability of providing such a service. Here, Quality-of-Service (QoS) is an important term, which means communication quality provided by the network. PSTN can provide a QoS of voice communication because it uses circuit switching, which dedicates network resources (e.g., bandwidth or time slots) to each voice communication. On the other hand, how can Internet Telephone Network provide a QoS of voice communication? To answer this question, there are many researches and developments about QoS of the Internet. In the next section, I discuss QoS of the Internet.

1.2 QoS: What are the problems for voice communication on the Internet?

As described in the former section, voice communication such as Internet Telephony requires network to keep some degrees of QoS. In order to investigate the impact of QoS of network, a quantitative scale of the voice quality is needed. Usually, we use Mean Opinion Score (MOS) test for evaluating voice quality [HGP00, Hel98, Dou00]. The MOS test is a subjective measurement method with the absolute category rating. In the MOS test, each of 24 listeners reports five levels of the absolute quality of speech samples without comparison to any reference. The five levels are listed in Tab. 1.1. Result of the MOS test is an average value of 24 listeners. In Tab. 1.2, I

describe a list of MOS values for the major voice codecs. G.729.A and G.723.1 are usually employed as codec of Internet Telephony. These codecs require bandwidth of less than 10 Kb/s, and their quality is higher than codec of cellular phone.

Table 1.1: Listening quality scale of MOS test

Excellent	5
Good	4
Fair	3
Poor	2
Bad	1

Table 1.2: MOS values of major voice codec

G.711	Telephone	64Kb/s; PCM	4.2
G.726	Personal Handyphone System (PHS)	32Kb/s; ADPCM	4.0
G.729.A	Internet Telephony	8Kb/s; CS-ACELP	4.0
G.723.1	Internet Telephony	6.4Kb/s; MP-MLQ	3.9
GSM6.20	European Cellular Phone	5.6K/s; VSELP	3.5–3.7
PDC 2	Japanese Cellular Phone	3.6Kb/s; PSI-CELP	3.4–3.6

1.2.1 The impact of packet loss

In Internet Telephony Network, packet loss degrades the voice quality. For example, in a typically used voice codec of G.723.1, the MOS value is 3.9 if there is no packet loss, as described in Tab. 1.2. We can say that the MOS value of 3.9 is near quality of Personal Handy-phone System (PHS), which is popular in Japan. In a case of 3% of packet loss, the MOS value decreases to 3.43, which is near quality of cellular phone [HGP00, p. 86]. In a case of 10% packet loss, the speech quality would be degraded badly. Therefore, percentages of the packet loss should be bounded to a value, such as 1%.

However, the current Internet cannot provide such a bound of packet loss. Figure 1.1 shows examples of packet loss on the current Internet. In this figure, I have investigated transition of packet loss using "ping" program. The source host and the

destination host is in Nara institute of science and technology, Japan, and in University of Melbourne, Australia, respectively. And the tested time is beginning from midnight, January 27, 2001 in Japanese local time. Each plotted point is an average value of 60 seconds. In the figure, there are some minutes of high percentages of packet loss. Such packet losses is not tolerable for Internet Telephony.

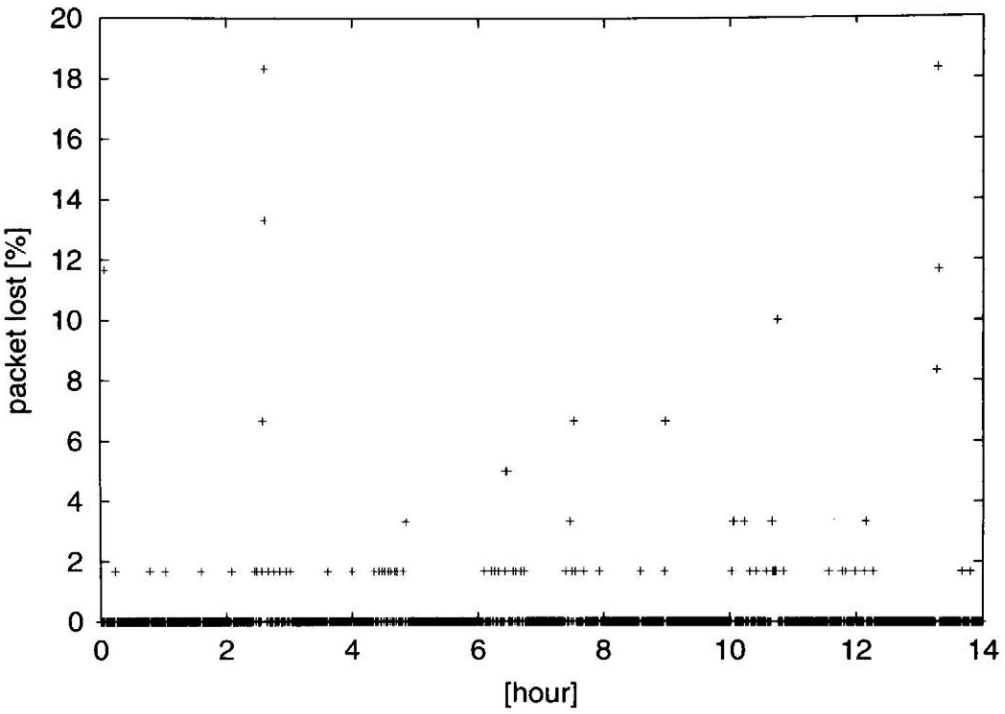


Figure 1.1: Examples of packet loss on the current Internet

1.2.2 The impact of delay time

Furthermore, large delay and delay jitter (or variance of delay time) degrade voice quality. The impact of delay jitter can be compensated at receiver hosts. Further discussion about delay jitter appears in the next section. In this section I discuss delay time, which is one way transmission delay time of the speaker side and the listener side.

I describe a list of standard of delay time [HGP00, Hel98] in Tab. 1.3. This table shows that one way delay of 400ms or 500ms is too high for voice communication. Note that large delay further causes the "echo" problem. If such a problem occurs, we have to prepare a facility called "echo canceler" to solve the problem.

Table 1.3: Standard of delay time in voice communication

ITU-T G.114	Toll quality: < 100ms	Good: < 150ms	
ESTI TIPHON	Best: < 150ms	High: < 250ms	Medium: < 450ms

In Fig. 1.2, I give an example of delay time in the current Internet Telephony Network[Goo99]. The total delay time is over 500ms, which is too large for voice communication. The large delay is caused by the gateway, IP network, access network and the PC client. We should reduce delay time to the acceptable value somehow. The large delay problems at the gateway and at the PC client seem to be caused by an early stage of hardware implementation, which can be improved by further developments. The delay time of IP network, which is indicated as 96ms in the figure, is caused by network of long distance between U.S. coast to coast. This type of delay cannot be reduced. Lastly, the delay time of access network, which is given as 150ms in the figure, is also large. In Chapter 3 and 4, I would like to reduce this type of large delay.

1.3 Mechanisms providing QoS

In the former section, I have discussed what kind of QoS is needed for voice communication. The major QoS related parameters are packet loss, delay and delay jitter. In this section, I introduce the mechanisms for reducing the impact of these QoS related parameters. For convenience of explanation, I first explain mechanisms for delay jitter, then mechanisms for delay, and finally mechanisms for packet loss.

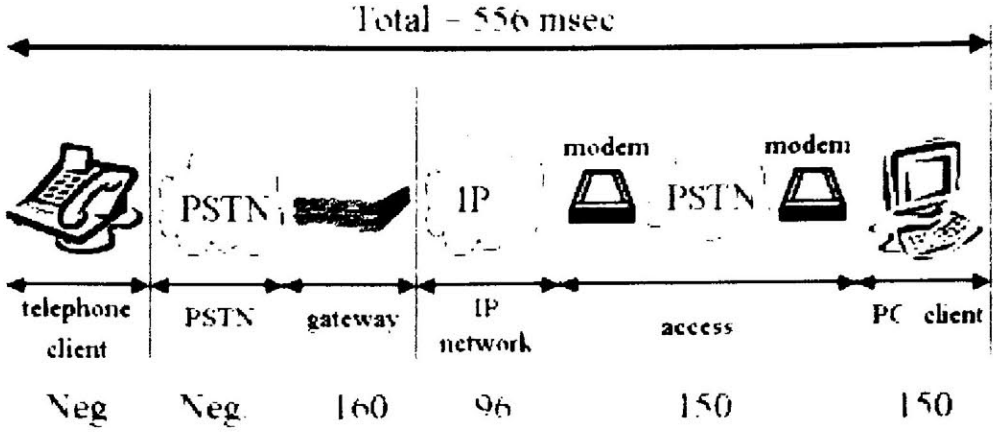


Figure 1.2: Typical delay time in the current Internet Telephony

1.3.1 Mechanisms for delay jitter

Delay jitter is a variance of delivering time of packets in a flow. In packet-switched networks, like the Internet, each intermediate node (or router) stores packets while the forward line is not available for that packet. Packets are forwarded in a First-Come First-Serve (FCFS) basis. If the forward line is empty empty when the packet arrives at every intermediate node, it will experience the smallest delivering time. On the other hand, if there are many packets waiting in buffer when a packet arrives, it will experience larger delay that the smallest case.

To compensate delay jitter, the receiver host attempts to provide synchronous playout of voice packets in the presence of random delay jitter[KR01, p. 503]. If the playout time of each voice packet is not synchronized, the voice quality will be bad. In order to smooth the playout time, Real Time Protocol (RTP) [SCFJ96],[Hel98, pp. 59–62],[HGP00, pp. 10–14] is usually used for voice communication. RTP provides two additional information for each voice packet: sequence number and timestamp.

In the Internet, there are packet losses and out of order delivering because of packet-by-packet routing. By means of sequence number, the receiver host can detect

both packet loss and out of order delivering.

The timestamp is used for synchronizing playout. First, the receiver host decides fixed playout time (q ms). For Internet Telephony, a value between 150ms and 400ms for the fixed playout time is usually employed. When a voice packet arrives at the receiver host, the host checks timestamp of the packet. The timestamp was written by the sender host of the packet. Therefore, packets are scheduled to be played out q ms behind the timestamp. In this way, we can compensate delay jitter for voice communication in most of cases.

However, if many packets arrive behind the playout schedule, the voice quality will be degraded significantly. This might be a problem involved in "delay time".

1.3.2 Mechanisms for delay time

As described in the section 1.2.2, large delay results in bad quality of voice communication, therefore we have to limit the end-to-end delivery time of packets to a value. There are two types of delay bounds: hard guarantee and soft guarantee[Par93],[Kes97, p. 219].

In the hard guarantee, also called deterministic guarantee, we say "the network will deliver every packet in T ms". On the other hand, in the soft guarantee, also called statistical guarantee, we say "the network will deliver p percent of packets in T ms".

Generally, hard guarantee can provide only large T because of providing 100% reliability. Therefore, the soft guarantee with very high percentage of p is preferred to the hard guarantee. I will demonstrate this results in Chapter 3 and 4.

Next, large delay is caused by congestion of networks. There are two reasons of causing the congestion: burst packet arrival and existence of too many sources.

The first one, burst packet arrival causes instantaneous overloads of networks. Basically, the packet switched networks are designed for handling bursty traffic such as file transfer, e-mail, bank transaction, and so on. Since bursty traffic might create instantaneous overloads of packet arriving at routers, the packets have to wait serving previously arriving packets in such a case. In order to remove the effect of burst packet

arrival, priority scheduling can be used in router buffer. Since real-time packets such as voice packets require to be delivered at their destinations as fast as possible, we give a priority to real-time packets over the other traffic. As a result, voice packets experience lower delay than that without priority scheduling.

Note that a well-known problem of *starvation* arises from the ordinary static priority scheduling. Starvation prevents lower priority traffic from utilizing the network. This problem can be solved by Class-Based Queueing (CBQ) [FJ95], which is a variation of the ordinary priority scheduling.

Furthermore, shaping of traffic sources also has an effect of reducing burstiness of packet arrival process. Shaping methods, e.g., stop and go, leaky bucket, and token bucket [Par93, Kes97, KR01], enlarge interval time of consecutive packets, thus burstiness of packet arrival process can be reduced.

The second reason of congestion, existence of too many sources are due to lack of resource allocation and admission control in the current Internet. Priority scheduling, described above, can suppress the effect of bursty non-real-time traffic, but does not care of too many sources of voice traffic. Therefore, we need these two methods for limiting the number of incoming flows to the networks.

For resource allocation and admission control, two methods are standardized for the Internet: Resource reSerVation Protocol (RSVP) [BZB+97] and Common Open Policy Service (COPS) [BCD+00]. However, these two methods are now in deployment for wide use in commercial networks.

RSVP is a resource allocation protocol in a per-flow basis. The objective of RSVP is to implement a similar service to the circuit switched networks. Admission control of RSVP performs in a way of a distributed fashion, thus requested flow can be accepted if every intermediate node allows to accept QoS of the requested flow.

RSVP has two practical problems: scalability and manageability of network operators [Wan01]. The scalability problem is caused by the per-flow resource allocation. In backbone networks, there may exist a large number of flows, therefore state management of routers in terms of connection establishment is very hard job. Moreover, signaling load of RSVP processing increases too high. The manageability problem is that RSVP do not have a resource management mechanism for network operators

who have to monitor, control and enforce use of network resources.

For these reasons, recently, there has been standardized a state-full policy management protocol, called common open policy service (COPS) protocol, which performs network resource management in a centralized fashion. In COPS protocol, network operators can enforce their policy for admission control. In most implementations of COPS, DiffServ is used with COPS [BBC+98]. DiffServ is not a per-flow protocol but a per-class protocol, so that routers do not have to manage states in per-flow. However, another scalability problem arises because the centralized fashion can bring a huge number of admission requests to the server. In Chapter 5, I will try to solve this type of scalability problem.

1.3.3 Mechanisms for packet loss

Like the reason of causing additional delay, the packet loss is caused by network congestion. Therefore, priority scheduling is useful because it can ignore the effect of burst packet arrival. Moreover, resource allocation and admission control mechanism likely RSVP and COPS, is needed for bounding packet loss probability.

Usually, real-time communication like voice communication consumes bandwidth constantly and continually*, therefore the resource allocation mechanism assigns a constant amount of bandwidth. Since there is no burstiness in constant bandwidth consumption, the resource management of bandwidth is required to provide only very small percentages of packet loss.

1.4 Overview of this dissertation

So far, I have introduced Internet Telephony and related QoS mechanisms for realizing a cost-effective multi-service network. Recall Internet Telephony Network would be the next killer application to the world wide web. However, there are few theories to design Internet Telephony network. Therefore, I try to construct design principles of Internet Telephony Network for end-to-end communications in

*For this reason, real-time traffic is also called continues-media traffic.

this dissertation.

First in Chapter 2, I focus on the whole of the Internet to give the big picture of end-to-end communications. My design architecture mainly consists of three types of networks, i.e., the domain networks, the access networks, and the stub networks as illustrated in Fig. 2.1. Among them, I point out issues arising in the former two networks for end-to-end QoS provisioning. The access networks cause large transmission delay because of small transmission capacity. On the other hand, the domain networks have relatively large transmission capacity, so that transmission delay time is insignificant. But, a scalability problem in terms of number of flows is crucial because such networks often accommodate huge number of flows. In Chapter 2, I would like to emphasize that the Internet Telephony Network should be divided into sub-networks for designing end-to-end QoS aware networks. Each sub-network has a different performance bottleneck for providing QoS.

Next, in Chapter 3 and 4, I then see the access networks, in which the most crucial issue is large delay time of a packet transmission due to relatively small capacity of links. It is known that large delay time heavily degrades the QoS. Therefore, limiting the maximum number of voice flows, which can keep delay time under a value, becomes an important issue. To evaluate such delay time in access network, I construct two models of access networks; homogeneous and heterogeneous flows. The homogeneous flows mean that their packet transmission rates and their packet lengths are the constants. This represents a multiplexing of a number of voice flows of the same characteristic. In my model, there are a number of voice flows and also data flows. Data flows are obstacles to voice flows. This is a preliminary but an important case, because widely spread dial-up access lines have very small capacity, which can accept only flows of small bandwidth. In the homogeneous case (Chapter 3), I obtain an exact and closed form solution to the delay performance, so that we do no need any approximations or numerical Laplace inversions. Schemes to reduce delay are proposed and their performance is evaluated.

Furthermore, in Chapter 4, the heterogeneous flow case treats three types of traffic; voice, video and data traffic. The model allows a multiplexing of different transmission rates and different packet lengths, e.g., accommodating both voice and video traffic.

Similar to the homogeneous case, data flows are the obstacles to voice and video flows. My numerical results show the large packet length of video packet has a large influence on voice delay.

Next, in Chapter 5, I try to provide a design architecture of backbone networks. The backbone networks have a large transmission capacity, so that delay experienced by real-time packets, like voice and video, will be negligible because of its high speed packet forwarding capability. What is the bottleneck of backbone networks? That is state requirements in routers. Real-time flows request to allocate their dedicated bandwidth in an end-to-end region. Consequently, the network nodes (or routers) should maintain a number of states in terms of flow establishment, and are desired to exchange the signaling messages for allocating some dedicated bandwidth to the real-time flows. For example, a high-speed link of 10 Gb/s can accommodate one million voice flows of 10 kb/s. Consequently, the state requirements will be of major concern in future high-speed networks. Furthermore, call blocking probability, which describes a network utilization from user points of view, is also investigated. Through the numerical results, I examine trade-offs between call blocking probability and the state requirements.

The results discussed in Chapter 2 are mainly taken from [IKTO00], Chapter 3 from [ITSO01b], Chapter 4 from [ITSO01a] and Chapter 5 from [IKTO99].

Chapter 2

Design Issues Arising in Internet Telephony Network

2.1 Introduction

The Internet has been significantly changed in terms of its qualitative feature as well as its quantitative one such as its transmission capacity. In particular, multimedia real-time communication services over the Internet requires research on their quantitative aspects and new architectures for quality of service (QoS) provisioning. The Internet telephony is one of the promising services because voice communication has been widely spread as a traditional real-time communication service and it can further provide a way of voice communication among various devices connected to the Internet.

Voice traffic can be relayed over several networks of different link capacity; there are mainly three types of networks, i.e., the domain networks, the access networks, and the stub networks as illustrated in Fig. 2.1. The Internet service providers' (ISP) domain networks, which are the backbone part of the Internet, must be of high transmission capacity, e.g., using asynchronous transfer mode (ATM) and/or wavelength division multiplexing (WDM) technologies, whereas the access networks provide access to the domain networks. Some of the access networks will be of still relatively low transmission capacity simply because low capacity links are good enough for voice

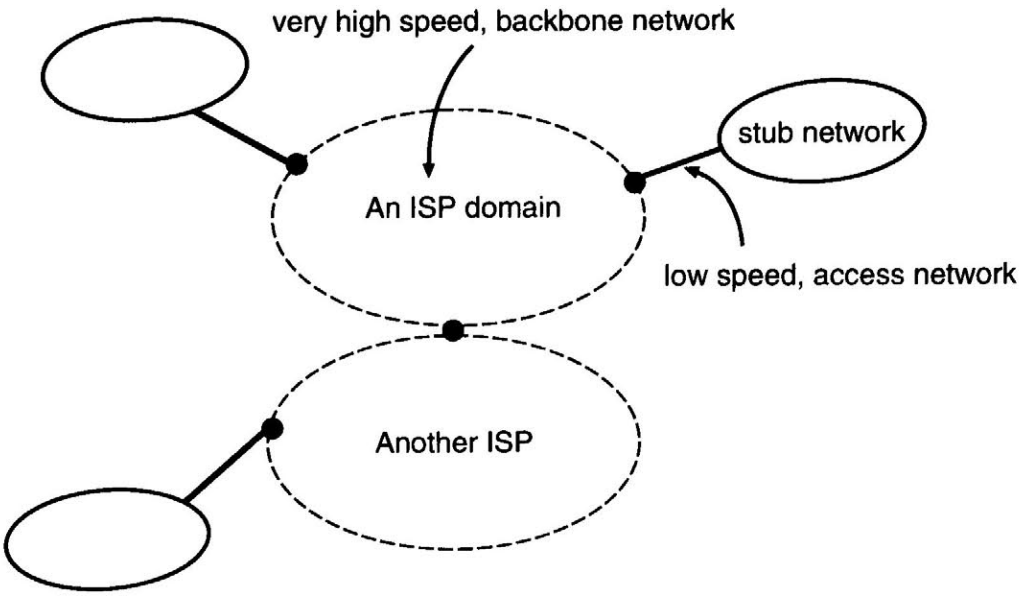


Figure 2.1: An architectural model of the Internet

traffic. On the other hand, the stub networks represent networks of relatively small size and relatively large transmission capacity, such as internal networks of a building or a campus.

The fundamental mechanism to provide QoS there is standardized by Internet Engineering Task Force (IETF) integrated services (IntServ), RSVP, and other working groups. However, there are still two types of issues remaining unsolved in this context. One is how to handle a vast number of voice flows carried over the domain network of high transmission capacity. For example, a link of 10 Gb/s can accommodate as many as one million voice flows of 10 kb/s. The delay experienced by the voice packets there will be negligible due to its high speed packet forwarding capability, and thus only the bandwidth assignment will be of major concern. Another issue is how to guarantee the delay time requirement over the access network of low transmission capacity, in which the delay experienced could be greater than the requirement without any adequate control, due to the networks' low transmission

capacity. Since, the number of voice flows handled there is very small, complicated management for guaranteeing the delay will be acceptable. Consequently, as QoS parameters, bandwidth and delay time are of major concern in the domain networks and the access networks, respectively.

From the above discussion, we will attack two issues in what follows. First, the performance of delay time experienced by voice packets over a link of relatively low transmission capacity will be examined by means of our exact analysis. Our analysis provides an explicit expression for the delay time *distribution*, and thus allows call admission control (CAC) based upon the *statistical* delay bound, instead of deterministic delay bound. In our analysis, voice packets have priority over other packets such as transmission control protocol (TCP) packets. The impact of the TCP packet length on the delay performance of voice packets will be also investigated. Next, we will study an efficient way of handling many voice flows. That is a flow aggregation technique in which the routers manage the flows as trunks, consisting of many flows. This successfully reduces the number of trunks to be treated, but can cause high blocking probability due to inefficient bandwidth assignment compared with the per-flow management. For keeping low and acceptable blocking probability, we will examine a bandwidth management scheme called *provisioned capacity*, which refers to the capacity actually provided to real-time traffic class and must be greater than that required on an average basis. We will give numerical results on the blocking probability by our exact analysis, and show how the flow aggregation can be done effectively while avoiding inefficient bandwidth assignment.

Through our examinations, we provide an architecture for end-to-end QoS provisioning. The following sections are as follows. In Section 2, we treat the low speed access networks. In Section 3, we pay our attention to the high speed backbone networks. In Section 4, we conclude this chapter.

2.2 Delay Performance in Access Networks

Relatively low speed links, such as T1 links, are likely to be still used even in future access networks, as mentioned earlier. Packets can experience very large delay

over those links, hence it is of practical importance to reduce the delay time there. Thus, the related issues are discussed in IETF integrated services specific link layers (ISSLL) working group.

As solutions to reducing the delay time, two technologies have been proposed in IETF: technologies of header compression and segmentation [CJ99]. The header compression is applied to voice packets, which reduces the overhead of voice packets, leads to efficient use of network resources such as intermediate routers and links, and can reduce the delay time as a result. On the other hand, the segmentation is applied to TCP packets sharing the network resources together with voice packets, which makes TCP packet transmission time smaller and in turn waiting time of voice packets smaller as mentioned below.

First, the header compression is effective in particular for low bit rate voice codec. For example, the G.723.1 codec generates a frame of 24 bytes at intervals of 30 ms [Cox97], [Hel98, p. 110]. Each frame is conveyed over the Internet on the payload of an internet protocol (IP) packet, which is usually called IP datagram. It is very likely that each of the IP packets includes at least a header of 40 bytes in IP version 4, which consists of IP, user datagram protocol (UDP), and real-time protocol (RTP) headers, as shown in Fig. 2.2. This results in inefficient use of network resources: the effective utilization is only 0.375. The header size can be significantly reduced to only two bytes without header cyclic redundancy check (CRC) or four bytes with CRC by use of the compression scheme developed so far.

Next, even if voice packets have priority over non-real-time packets, they will be forced to wait until the transmission of non-real-time packets currently being served ends. Therefore, large non-real-time packets or TCP packets should be segmented in this context [CJ99]. This enables voice packets to be interleaved with segmented TCP packets of small size, resulting in small waiting time for the voice packets.

In what follows, considering the impact of the above features, we examine the performance of queueing delay time of voice packets in a router or a *multiplexer* by means of numerical results derived from our exact analysis of the following model.

Voice packets, called CBR packets below, have priority over non-real-time packets, called TCP packets from now on only for convenience. More precisely, N independent

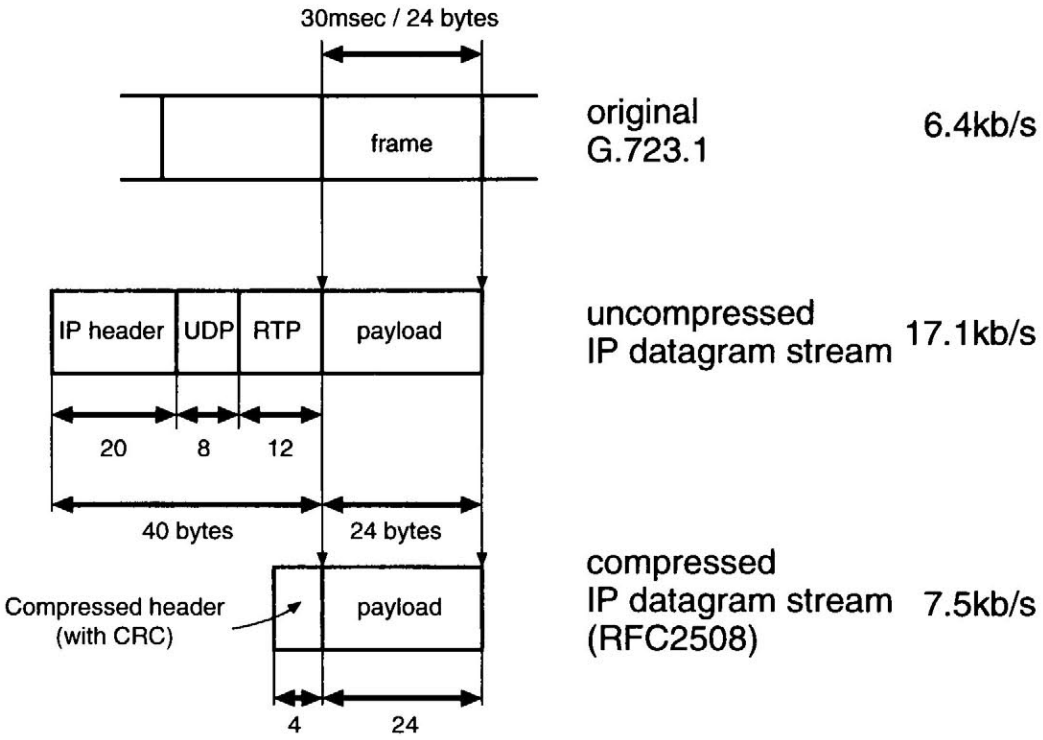


Figure 2.2: Encapsulation of IP datagram

CBR flows and TCP flows share the multiplexer. If there are any CBR packets waiting, the multiplexer continues to transmit them. Otherwise, TCP packets will be transmitted. Non-preemptive priority service discipline is employed so that CBR packets will be enforced to wait until the transmission of TCP packets currently being served ends. Furthermore, we deal with a case where at least one TCP packet is waiting in the buffer equipped. This scenario is very likely to happen due to a greedy flow control mechanism used by TCP, trying to get as much link capacity as possible. This case must be the worst one for CBR packets in terms of their queueing time because at least one TCP packet will be always interleaved between two consecutive CBR packets if the preceding CBR packet leaves before the following one arrives.

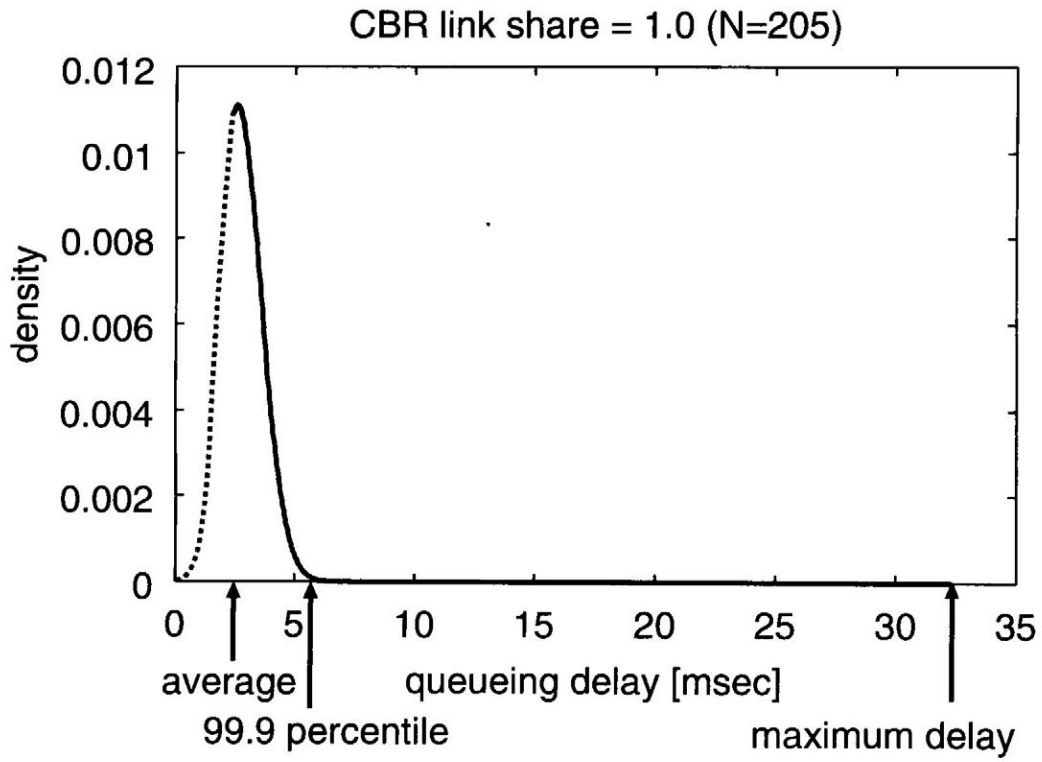


Figure 2.3: A queueing delay density function

The model described above is equivalent to a queueing system called the $nD/D/1$

with vacations. The waiting time distribution can be obtained for more general system, i.e., the $nD/G/1$ with vacations by combining already known results. However, this needs the numerical Laplace inversion, which can lead to inaccurate numerical outputs and further needs computation of a large amount of time particularly for large N. On the other hand, when the service time is constant, we can obtain the explicit expression for the waiting time distribution function in Chapter 3. The equations described in Chapter 3 are quite complicated, so we only show the preliminary numerical results. The detailed discussions will appear in Chapter 3.

From Chapter 3, Fig. 3.4, we can see that the maximum delay is heavily affected by the CBR link shares, while the 99.9-percentile delay is almost insensitive to it. To explain this, we give a queueing delay density function in Fig. 2.3.* The density function, which has been obtained in a link share equal to 1.0, is seriously biased toward a low value of the queueing delay. This is because it is very unlikely that packets from the most of CBR flows arrive simultaneously since packet arrival process from CBR flows is independent and identically distributed. The figure shows that there is a significant difference between the deterministic delay bound and the statistical one, in particular for a large CBR link share or for a large number of CBR flows N Thus, the statistical delay bound is strongly recommended for efficient CAC in the access networks, instead of the deterministic one.

We have found a number of other important results in terms of voice communications in the Internet using this analysis. However, the discussion will be repeated in Chapter 3. For this reason, we only summarize results obtained by the analysis at this point: 1) At large link capacity, unlike the 100-percentile delay, the 99.9-percentile delay is small enough to be negligible. 2) The 99.9-percentile delay bound can accept a larger number of streams than the 100-percentile delay bound when CBR payload size is large. 3) Header compression scheme can reduce delay to half in the N-ISDN networks. 4) Large TCP packets should be segmented into small size, such as 50 or 100 bytes, in extremely slow networks.

So far, we have been restricted to the homogeneous CBR flows. Please refer to

*By our analysis, we can obtain only the solid line in Fig. 2.3 due to a numerical stability condition. The dotted line is added for reference.

Chapter 4 for the performance in heterogeneous environments, in which there are CBR flows of different rates and of different packet sizes.

2.3 Flow Management in the High Speed Backbone Networks

Recently, the traffic conveyed over the Internet has been growing explosively, thereby requiring backbone networks of huge transmission capacity. The ATM and WDM technologies enable such networks to be built and provide various alternatives to the real-time communication over the Internet.

The Internet community has been intensively discussing ways to achieve the real-time communication over the Internet. First, the IntServ architecture and RSVP signaling scheme have been standardized. However, RSVP is based upon the per-flow management, and this causes a scaling problem [FH98, p. 156]. As a solution to that, IETF has discussed a new architecture, called *differentiated services* (DiffServ) architecture [BBC+98]. The DiffServ architecture has introduced the edge concept, in which traffic marking and shaping is done at edges of the network and intermediate routers constituting high speed backbone networks process each packet based upon its mark; the related processing is called *per hop behavior* (PHB). This architecture allows flow aggregation, i.e., a number of traffic flows can be handled as one unit. Ingress edge routers can enforce the flow aggregation, and egress edge routers perform the flow segregation.

A high speed link, e.g., 10 Gb/s, can accommodate one million voice flows of 10 kb/s, as mentioned earlier. The per-flow management is a costly way in this context, while the flow aggregation can reduce the number of units managed independently to acceptable levels. Nevertheless, since it assigns an amount of the bandwidth required by one managed unit even if only one flow of the unit is accepted, it will lead to an inefficient use of the link. As a result, it is expected to suffer a high call blocking probability in CAC, compared with the per-flow management as long as available bandwidth is fixed.

In what follows, we study the performance of the flow aggregation by analyzing the call blocking probability of voice flow. We show features related to the flow aggregation, and propose a way of keeping the blocking probability less than some acceptable one. The analysis is quite comprehensive, so please refer to Chapter 5.

In our method, we use a term of *trunk*. (This is also illustrated in Figs. 5.4, 5.5, and 5.6.) A *trunk* is defined as a unit consisting of a number of flows, and each intermediate router recognizes only trunks, instead of flows. First, when a voice flow setup request arrives at an ingress router, it will establish a trunk for an egress router if no trunk for that egress router is available, and the required bandwidth is available on the output link of the ingress router. If the available bandwidth on the link is not large enough, the flow setup request will be rejected. The link can accept a maximum of C trunks. Furthermore, suppose that there is a trunk already established. A trunk can accommodate a maximum of say m_k flows, which is the trunk capacity (see Fig. 5.5). If the trunk already includes m_k accepted flows, the new flow setup request will be rejected. Otherwise, it will be accepted. Thus, call blocking can occur in establishing a flow within the already established trunk as well as a new trunk. This leads to a two-stage model, as shown in the figure. In the first stage, the arrivals of flows in a trunk are characterized by a Poisson process, and service times of flows are independent and identically distributed. The first stage can be modeled by an $M/G/m/m$ type queueing system, and the second stage can be analyzed by the generalized Engset model.

Through this analysis, we have found flow aggregation can dramatically reduce the number of states to be managed by the routers. This is very important thing to implement high speed routers, as appeared in Chapter 5.

2.4 Conclusions

We have investigated two types of issues for realizing end-to-end QoS provisioning. One is related to delay performance in the access networks of rather low speed links, and another is an effective flow management scheme in the domain networks of very high speed links.

First, we have obtained some features of the statistical delay bound by means of our explicit expression for the delay time distribution function, and shown that the statistical delay bound is effective in accommodating voice flows efficiently on low speed links. We have treated some ways of reducing the delay time such as header compression and segmentation schemes, and investigated their effect. Next, we have dealt with flow aggregation as a cost-effective way of flow management on high speed links. We have evaluated its blocking probability and the amount of extra bandwidth required to keep the blocking less than some acceptable value.

End-to-end QoS provisioning needs different approaches in different types of sub-networks. In this chapter, we have investigated effective ways of QoS provisioning for both low speed access and high seed domain networks.

Chapter 3

Delay Analysis of Voice Traffic in Access Network

3.1 Introduction

The future Internet should support both traditional non-real-time and real-time services. Examples of the real-time service are interactive video, voice, and other time stringent applications. Such applications require the network to meet a deadline for delivering packets to destinations. If many packets arrive too late, the quality of the real-time service will deteriorate significantly. Thus, QoS control based upon packet delivery time should be supported by the network.

In this chapter, we examine the delay performance of packets from constant bit rate (CBR) traffic as real-time traffic whose delay can be affected by non-real-time traffic. More specifically, our model has two types of traffic: superposition of N independent CBR streams and non-real-time streams. At a multiplexer, multiple CBR streams share a single buffer, whereas non-real-time streams share another distinct buffer. The CBR buffer has non-preemptive priority over the non-real-time buffer. This strategy can be easily implemented and can minimize the effects of non-real-time traffic on the delay time of packets from CBR streams. We analyze the delay for CBR packets by solving the $nD/D/1$ queue with vacations. The vacation time represents service time of a packet belonging to non-real-time streams. We obtain an exact and

closed form solution of the delay, hence obviating the need of any approximations or numerical Laplace inversions.

Future backbone networks are operated at a very high speed, e.g., Gb/s. On such networks, the delay time experienced by packets is very low. On the other hand, low speed links are likely to remain in access networks, for example 33.6Kb/s links or T1 links. The delay experienced by CBR packets on such links is very large so that reducing the delay on the links to an acceptable value is of practical importance to provide the real-time communication service. Our analysis enables us to easily calculate a statistical delay bound, which is lighter than a deterministic bound. The difference of those two bounds is oftentimes an order of magnitude. Thus, a connection admission control (CAC) based on a statistical bound can achieve efficient use of the bandwidth. Furthermore, our numerical results show that the size of packets from both types of streams has a great impact on CBR packet delay; longer packets cause longer delay. Conversely, shorter packets degrade the throughput performance because it raises a ratio of the header size to the payload size. Therefore, there might exist an optimal size of packets due to the tradeoff between these factors. We examine this issue in the body of the chapter.

Several past works exist about analyzing delay time of a superposition of CBR streams. This problem can be represented by finding the waiting time distribution of $nD/D/1$ queue. Roberts and Virtamo have developed a closed form formula [RV91]. Dron and others have proposed an asymptotic formula whose time complexity is independent of the system size N [DRS91]. The formula works well over large N Ramamurthy and Sengupta have suggested an approximate solution method for $N \leq 100$ [RS91]. More recently, studies for slotted networks have appeared in the context of ATM networks. Humblet and others have presented an analytical method based on the Ballot theorems [HBH93]. The method provides steady-state delay distribution as well as transient behavior of the system. Privalov and Sohraby have provided an exact analysis of the jitter process [PS98]. Their results indicate that jitter variance is bounded and never exceeds the constant 2/3.

Our study has two differences from the above past researches: 1) we consider non-real-time traffic that can affect CBR delay time and obtain an exact and closed

form solution in that context, and 2) we discuss an adequate size for packets based on the tradeoff stated above. These evaluations are for multiple classes of service on low speed links such as in access networks.

The rest of the chapter is organized as follows. In Section 2, we describe the system treated here and its mathematical model. In Section 3, we analyze the model. In Section 4, we give some numerical results to show a quantitative evaluation of the statistical delay bound. Especially, we focus on the impact of the packet sizes on the delay performance. In Section 5, we conclude the chapter.

3.2 System and Model Description

In this section, we provide the system description and the mathematical model used for the system analysis.

3.2.1 System Description

At a multiplexer, multiple CBR streams and non-real-time streams share a bottleneck link as shown in Fig. 3.1. The packets from CBR streams are required to be transmitted as fast as possible. Therefore, we use priority scheduling to meet this requirement. The unused capacity left by CBR streams should be utilized effectively by non-real-time streams. Multiple CBR streams share a single buffer, whereas non-real-time streams share a distinct buffer. The CBR buffer has non-preemptive priority over the non-real-time buffer.

Note that a well-known problem of *starvation* arises from the ordinary static priority scheduling. Starvation prevents lower priority traffic from utilizing the network. This problem can be solved by Class-Based Queueing (CBQ) [FJ95], which is a variation of the ordinary priority scheduling. While CBQ is of practical interest, we do not go into the details of CBQ to get the nature of the simple non-preemptive priority scheduler.

One of our main objectives is to describe a delay bound for CBR that is tight and suitable for admission control. Therefore, we assume the CBR buffer to be of infinite

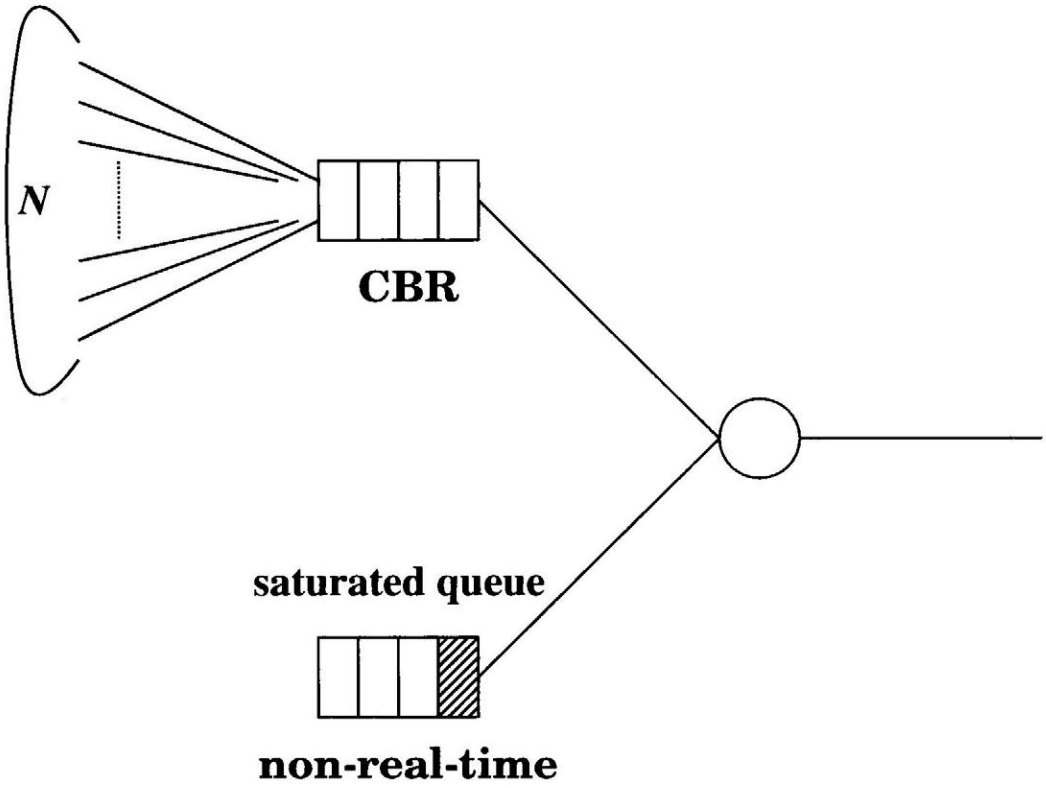


Figure 3.1: System overview

size in order to get the worst-case bound.

Most of the non-real-time traffic in the Internet is carried by transmission control protocol (TCP). The congestion control mechanism of TCP is placed at the end-nodes so that TCP sources try to get as much link capacity as possible. Generally speaking, a number of packets from TCP sources are in transit at intermediate nodes. For this reason, we assume the non-real-time buffer to be saturated in our system so that at least one packet always exists in the buffer. This is the worst-case scenario for the CBR delay, considering the non-preemptive priority.

On the CBR side, there are N streams. We assume they are homogeneous in terms of rates and packet sizes.

3.2.2 Mathematical Model

We consider a single server queue with a superposition of N independent input streams, where arrivals from each input stream are characterized by an equilibrium renewal process with an interarrival time of D (deterministic). This means that the first arrival from each input stream occurs uniformly in the interval $(0, D)$ and is independent of the other streams. Service times are independent and identically distributed (i.i.d.) with a distribution function $B(x)$. We assume that each service time is less than D/N and the mean service time is denoted by b. Arrivals receive service on a first-come first-serve basis. We call this queue the ordinary queue, where the server is idle when the queue is empty.

Next, we consider a queue with vacations, where arrivals and services have the same characteristics as in the ordinary queue. In the queue with vacations, the server takes a vacation when the queue becomes empty. Vacation times are i.i.d. according to a distribution function $U(x)$ with finite mean u. If the server finds packets waiting in the queue upon returning from a vacation, it serves the packets continuously until the queue becomes empty, and then it takes the next vacation. This service discipline is called exhaustive. If the server finds no packets upon returning from a vacation, it takes another vacation. This vacation discipline is called multiple vacations.

Remark Each input stream corresponds to a CBR stream. Vacation times corre-

spond to the services of non-real-time streams. To investigate the worst-case delay scenario for the CBR streams that has (non-preemptive) priority over the non-real-time streams, we assume that the service of a non-real-time packet starts every time the server becomes idle.

3.3 Analysis

3.3.1 General Service Time

We can obtain the waiting time distribution for the queue with vacations by combining known results, stated in this section. In [Sen90], Sengupta analyzed the ordinary queue, where the mean service time b is normalized to one. Suppose the queue is empty at time zero. Let $V_k^*(s)$ and $W_k^*(s)$ denote the Laplace-Stieltjes Transforms (LST) of the virtual and actual waiting time distributions, respectively in the queue without vacations, when the number of arrival streams is equal to k. Following the same argument as in [Sen90], we have for $k = 1, \ldots, N$ and $Nb < D$,

$$V_k^*(s) = 1 - kb/D + (kb/D)\frac{1 - B^*(s)}{sb} W_k^*(s),$$

and

$$W_k^*(s) = V_{k-1}^*(s).$$

It then follows from these equations and $V_0^*(s) = 1$ that

$$W_N^*(s) = \sum_{k=0}^{N-1} \frac{(N-1)!}{(N-1-k)!} \left[\frac{1 - B^*(s)}{sD}\right]^k \left(1 - \frac{(N-1-k)b}{D}\right), \quad Nb < D. \tag{3.1}$$

Let $W^*(s)$ denote the LST of the actual waiting time distribution in the queue with vacations. Owing to the stochastic decomposition property of the $G/G/1$ queue with exhaustive and multiple vacations [Dos90], we have

$$W^*(s) = \frac{1 - U^*(s)}{su} W_N^*(s), \tag{3.2}$$

where $U^*(s)$ denotes the LST of $U(x)$.

3.3.2 Constant Service Time

In order to obtain the explicit expression of the distribution function $W(x)$, we consider the case of constant service time, i.e., $B^*(s) = e^{-sb}$ and constant vacation time, i.e., $U^*(s) = e^{-su}$ Note that the forward recurrence time of the constant service time b has a uniform distribution over $[0, b]$, whose distribution function $\tilde{B}(x)$ is given by

$$\tilde{B}(x) = \begin{cases} \dfrac{b - \alpha_b(x)}{b}, & x \geq 0, \\ 0, & x < 0, \end{cases}$$

where

$$\alpha_y(x) = (y - x)^+$$

Let $\tilde{B}^{(k)}(x)$ denote the k-fold convolution of $\tilde{B}(x)$ with itself.

Lemma 1

$$\tilde{B}^{(k)}(x) = 1 - \frac{1}{k!b^k} \sum_{n=1}^{k} \binom{k}{n} (-1)^{k-n} \alpha_{nb}^k(x), \qquad k = 1, 2, \ldots, \qquad x \geq 0. \tag{3.3}$$

The proof of Lemma 1 is given in Appendix 3.A.

Let $F^{(k)}(x)$ denote the convolution of $\tilde{B}^{(k)}(x)$ and the forward recurrence time distribution $\tilde{U}(x) = (u - \alpha_u(x))/u$ of constant vacation time u. We then have

$$F^{(k)}(x) = \int_{x-u+\alpha_u(x)}^{x} \tilde{B}^{(k)}(y) \frac{dy}{u} \tag{3.4}$$

Similar to the proof of Lemma 1, we can show that

$$F^{(k)}(x) = 1 - \frac{1}{(k+1)!b^k u} \sum_{n=0}^{k} \binom{k}{n} (-1)^{k+1-n} \left\{ \alpha_{nb}^{k+1}(x) - \alpha_{nb+u}^{k+1}(x) \right\}, \qquad k = 1, 2, \ldots,$$

which comes from (3.3) and (3.4).

From (3.1) and (3.2). $W^*(s)$ is given by

$$W^*(s) = \sum_{k=0}^{N-1} \frac{(N-1)!}{(N-1-k)!} \left(\frac{b}{D}\right)^k \left(1 - \frac{(N-1-k)b}{D}\right) \left[\frac{1 - B^*(s)}{sb}\right]^k \frac{1 - U^*(s)}{su} \tag{3.5}$$

We note here that the facter $\left[\frac{1-B^*(s)}{sb}\right]^k \frac{1-U^*(s)}{su}$ $(k = 1, 2, \ldots)$ is the LST of $F^{(k)}(x)$. For simplicity, we define $F^{(0)}(x)$ as

$$F^{(0)}(x) = \frac{u - \alpha_u(x)}{u}$$

It then follows from (3.5) that

$$W(x) = \sum_{k=0}^{N-1} \frac{(N-1)!}{(N-1-k)!} \left(\frac{b}{D}\right)^k \left(1 - \frac{(N-1-k)b}{D}\right) F^{(k)}(x), \qquad x \geq 0. \tag{3.6}$$

Further we note that

$$\sum_{k=0}^{N-1} \frac{(N-1)!}{(N-1-k)!} \left(\frac{b}{D}\right)^k \left(1 - \frac{(N-1-k)b}{D}\right) = 1.$$

Thus (3.6) is rewritten to be

$$W(x) = 1 - \frac{1}{Nu} \sum_{k=0}^{N-1} \binom{N}{k+1} \left(1 - \frac{(N-1-k)b}{D}\right) \left(\frac{1}{D}\right)^k \cdot \sum_{n=0}^{k} \binom{k}{n} (-1)^{k+1-n} \left\{\alpha_{nb}^{k+1}(x) - \alpha_{nb+u}^{k+1}(x)\right\}, \quad x \geq 0. \tag{3.7}$$

Note that (3.7) consists of terms that alternate in sign and therefore is not numerically stable.

To obtain a numerically stable expression, we rewrite (3.7) as

$$1 - W(x) = \frac{D}{Nu} \sum_{k=0}^{N-1} \binom{N}{k+1} \sum_{n=0}^{k} \binom{k}{n} (-1)^n \left\{\left(\frac{-\alpha_{nb}(x)}{D}\right)^{k+1} - \left(\frac{-\alpha_{nb+u}(x)}{D}\right)^{k+1}\right\} - \frac{b}{u} \sum_{k=0}^{N-2} \binom{N-1}{k+1} \sum_{n=0}^{k} \binom{k}{n} (-1)^n \left\{\left(\frac{-\alpha_{nb}(x)}{D}\right)^{k+1} - \left(\frac{-\alpha_{nb+u}(x)}{D}\right)^{k+1}\right\} \tag{3.8}$$

Lemma 2

$$\sum_{k=0}^{N-1} \binom{N}{k+1} \sum_{n=0}^{k} \binom{k}{n} x_n^{k+1} y_n^n = \sum_{n=0}^{N-1} x_n (x_n y_n)^n \sum_{k=0}^{N-n-k} \binom{k+n}{n} (1 + x_n)^k \tag{3.9}$$

The proof of Lemma 2 is given in Appendix 3.B. Applying Lemma 2 to (3.8), we obtain

$$1 - W(x) = \frac{D}{Nu} \left[\sum_{n=0}^{N-1} \left(\frac{\alpha_{nb+u}(x)}{D}\right)^{n+1} \sum_{k=0}^{N-n-1} \binom{k+n}{n} \left(1 - \frac{\alpha_{nb+u}(x)}{D}\right)^k \right.$$

$$
\left. - \sum_{n=0}^{N-1} \left(\frac{\alpha_{nb}(x)}{D} \right)^{n+1} \sum_{k=0}^{N-n-1} \binom{k+n}{n} \left(1 - \frac{\alpha_{nb}(x)}{D} \right)^{k} \right]
$$

$$
- \frac{b}{u} \left[\sum_{n=0}^{N-2} \left(\frac{\alpha_{nb+u}(x)}{D} \right)^{n+1} \sum_{k=0}^{N-n-2} \binom{k+n}{n} \left(1 - \frac{\alpha_{nb+u}(x)}{D} \right)^{k} \right.
$$

$$
\left. \left. - \sum_{n=0}^{N-2} \left(\frac{\alpha_{nb}(x)}{D} \right)^{n+1} \sum_{k=0}^{N-n-2} \binom{k+n}{n} \left(1 - \frac{\alpha_{nb}(x)}{D} \right)^{k} \right] \right] .
$$

The above expression is numerically stable when $\max(0, (N-1)b + u - D) \leq x \leq (N-1)b + u$.

3.4 Numerical Results

In this section, we show the numerical results obtained through our exact analysis of the delay time distribution. First, we focus on a low bit rate voice coding algorithm, called G.723.1 [Cox97, K+98, Hel98]. Although its bit rate is 5.3Kb/s or 6.4Kb/s, its quality is rated at as high as 3.98 on Mean Opinion Score (MOS) scale, while that of the traditional 64Kb/s voice coder is rated at 4.0 in the MOS tests [Hel98, pp. 110–111]. Therefore, as far as MOS test is concerned, G.723.1 can be considered an algorithm suitable for low bit rate networks. However, G.723.1 suffers from large algorithmic delay: lookahead delay of 7.5 ms, frame delay of 30 ms and processing delay similar to frame delay. According to ITU-T G.114, the communication quality is good if the total end-to-end delay is less than 150 ms. Therefore, the transmission delay should be minimized to satisfy this end-to-end delay requirement, in particular if G.723.1 is used.

Secondly, we consider the effect of the packet sizes. In case of CBR traffic, each frame in G.723.1 is usually generated at an interval of 30 ms, so that payload size is 24 bytes. On the other hand, the header size of Internet Protocol/User Datagram Protocol/Real-Time Protocol (IP/UDP/RTP) in IP version 4 is 40 bytes or more [Ste94, p. 34]. Therefore, the size of G.723.1 encoded CBR packets is at least 64 bytes over the Internet. On low speed networks, the header overhead can severely affect the delay performance. Recently an IP/UDP/RTP header compression scheme has been developed to reduce this large overhead [CJ99]. The compression can reduce the

header size to two (without header Cyclic Redundancy Check (CRC)) or four bytes (with header CRC). On the other hand, TCP packet size is typically around 40 bytes, 500 bytes, or 1500 bytes[CMT98]. Packets of around 40 bytes are acknowledgments. Packets of around 1500 bytes are usually used as data packets when path Maximum Transfer Unit (MTU) discovery function is implemented [Ste94, p. 237]. Otherwise, data packets are usually of size around 500 bytes. In this dissertation, we assume TCP packets to be of 500 bytes unless stated otherwise. We also discuss the effect of varying the TCP packet size at the end of this section.

For these reasons, CBR packet size, TCP packet size and others are parameterized as follows.

- CBR packet size : 28 bytes (4 byte header + 24 byte payload), 64 bytes (40 byte header + 24 byte payload).
- TCP packet size : 500 bytes.
- Link speed : L = 33.6Kb/s, 128Kb/s, 1,536Kb/s and 10Mb/s.
- Payload rate of CBR stream : C = 6.4Kb/s.
- Rate of CBR stream with header overhead : 7.47Kb/s (4 byte header case) or 17.07Kb/s (40 byte header case).

For simplicity, we introduce the notation CBR=(4+24)bytes, which represents a CBR packet consisting of a 4-byte header and a 24-byte payload.

Before showing numerical results regarding the parameters above, we briefly provide a comparison of delay time distribution functions of both the queue with vacations and the queue without vacations in Fig. 3.2. Let the number N of streams be nine, the arrival interval D of each stream be 10, and the constant service time b be one. We choose 0.2 and 1.0 as the constant vacation time u. As mentioned in the previous section, the queue without vacations means the ordinary $nD/D/1$ queue. In this dissertation, we compute the delay time distribution of the ordinary $nD/D/1$ queue using the approach for continuous time arrivals described in [HBH93]. In Fig. 3.2, the delay time of the queue with vacations is always greater than that of

the queue without vacations, and the delay time for small u approaches that of the queue without vacations, as expected.

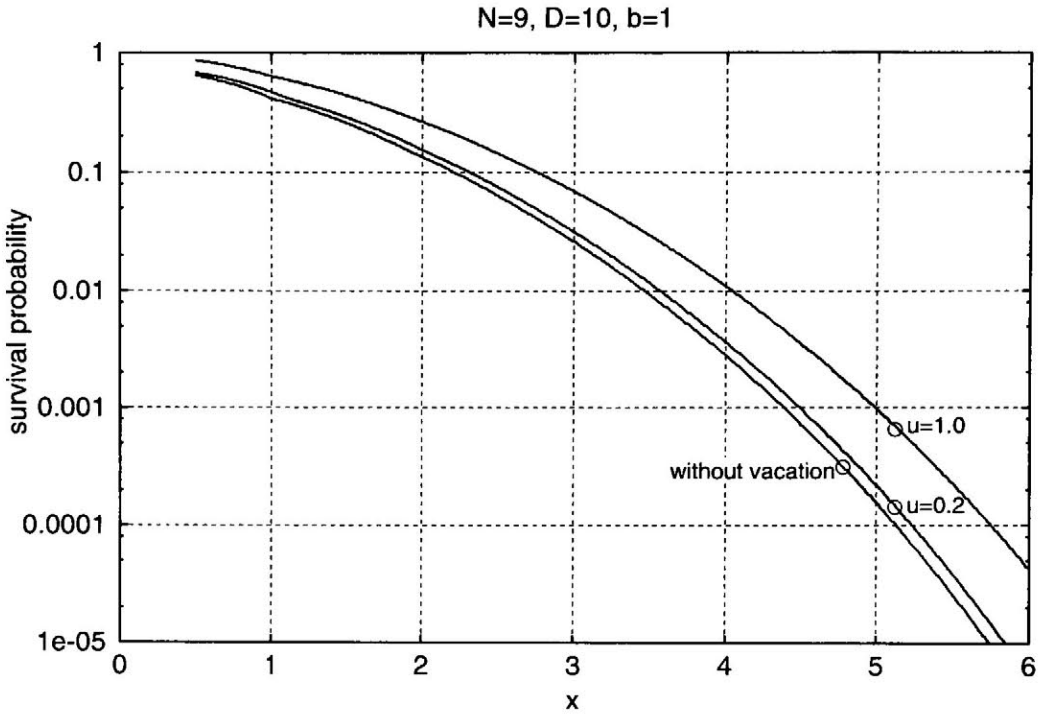


Figure 3.2: Comparison between the queue with vacations and the queue without vacations

In Fig. 3.3, we give an example of the survival function associated with the queueing delay distribution. We assume a link connecting with a modem of link speed L of 33.6Kb/s, and that link is shared by one voice stream of CBR=(40+24)bytes and TCP packets of 500 bytes. A TCP packet transmission takes about 119 ms on the link so that the delay performance of packets from voice streams can be heavily affected by TCP packets, as shown in the figure. This large delay can be reduced by the header compression and the segmentation of TCP packets. We demonstrate the effect of these schemes at the end of this section.

The link share is defined as a ratio of the total bandwidth of all CBR streams to the link capacity. For example, if a link share is 1.0, $N = 205$ streams of

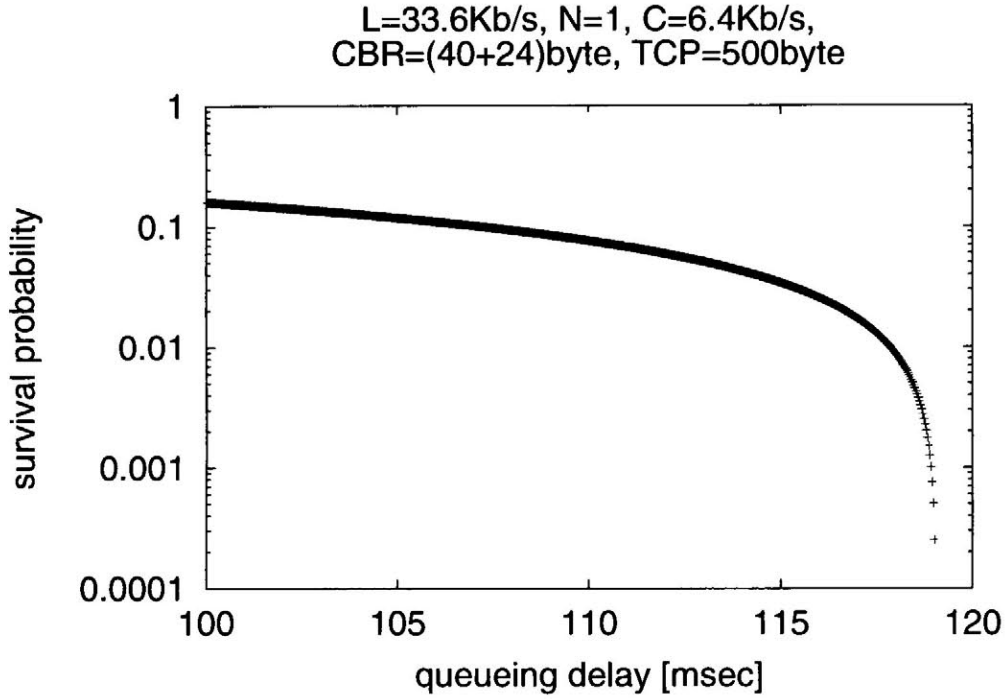


Figure 3.3: Example of the survival function on the queueing delay

CBR=(4+24)bytes are multiplexed because 1,536Kb/s divided by 7.47Kb/s is roughly 205.7. We investigate the impact of the CBR link share on the queuing delay in Fig. 3.4. The 100-percentile delay bound can be calculated in a straightforward way, and that is $(N-1)b+u$; this bound is also called the deterministic delay bound. The 99.9-percentile delay can be calculated by our analysis. This kind of delay performance is called a statistical delay bound here. From the figure, unlike the 99.9-percentile delay, the 100-percentile delay is heavily affected by the CBR link share. This can be explained as follows. It is very unlikely that packets from a large number of CBR streams arrive at the same time because the arrival process from CBR streams is i.i.d. For this reason, by increasing the number of CBR streams, one should not expect a significant impact on the statistical bound.

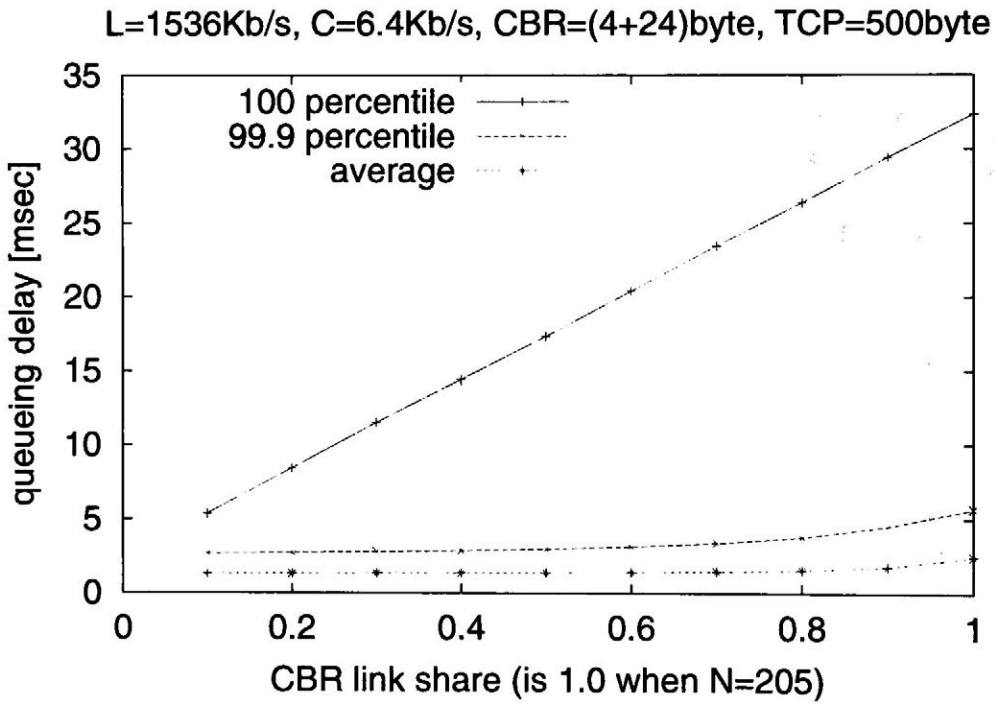


Figure 3.4: The impact of the CBR link share on the queuing delay

Next, we investigate the impact of the link capacity on queueing delay in Fig. 3.5.

Since CBR delay performance is the worst when the CBR link share is equal to 1.0, we have fixed the link share to that value. The number N of CBR streams at each amount of link capacity is given in Table 3.1. In this figure, TCP packet size is 500 bytes, each CBR stream is of CBR=(4+24)bytes, and CBR interarrival time D is 30 ms. Since the link share is approximately 1.0, Nb is approximately 30 ms and the deterministic delay $(N-1)b+u$ is always greater than 30 ms because TCP packet transmission time u is larger than CBR packet transmission time b. On the other hand, the 99.9-percentile delay is very small over a wide range of the link capacity. In particular, when the link capacity L is greater than 5Mb/s, the 99.9-percentile delay is nearly equal to 3 ms, i.e, 1/10 of the interarrival time D. Thus we can say that the queueing delay is small enough to be negligible when the link capacity L is greater than 5Mb/s. In other words, when more than 669 CBR sources are multiplexed (see Table 3.1), the queueing delay is immaterial in terms of the QoS of CBR streams. Note that it has been reported, e.g., in [K+98], which G.723.1 codec can tolerate packet losses of at most 10%. This allows the 90-percentile delay bound to be used for guaranteeing acceptable QoS. Thus, the statistical delay bound is of practical importance. In this dissertation, we employ the 99.9-percentile delay bound as a very strict one.

Table 3.1: The number of CBR streams in Fig. 4.9

Link capacity	# N of CBR streams
100Kb/s	13
500Kb/s	66
1Mb/s	133
5Mb/s	669
10Mb/s	1339

So far, we have dealt with CBR packets each including a payload of 24 bytes and a header of 4 bytes. TCP packet size was fixed at 500 bytes. In what follows, the impact of these factors is discussed. First, we show the impact of CBR frame size on the delay performance in Fig. 3.6. Each CBR frame is created by a voice encoder at a constant time interval. The frame is then packetized into an IP packet with a

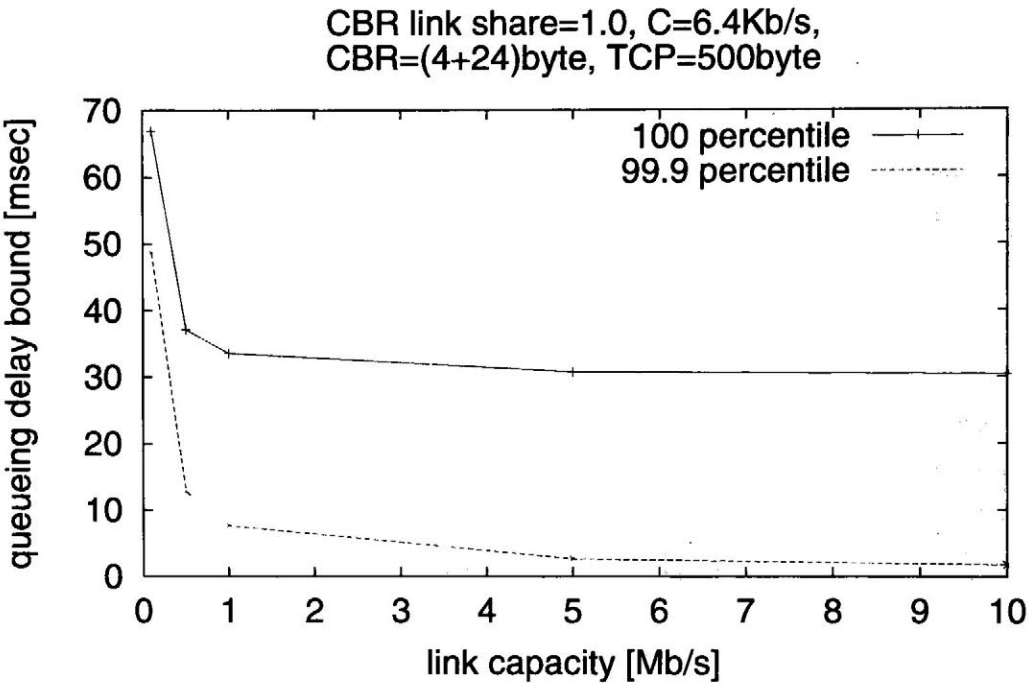


Figure 3.5: The impact of link capacity on the queueing delay

4-byte compressed header. The interarrival time of packets from each CBR stream is strongly related with the CBR payload size. For example, a 24-byte frame leads to 30 ms interarrival time D, or *packetization delay*, due to 6.4Kb/s coding rates. A larger payload size results in a larger packetization delay.

Nevertheless, as shown in Fig. 3.6, the 99.9-percentile queueing delay is almost insensitive to the CBR payload size, whereas the 100-percentile queuing delay linearly increases with the CBR payload size.

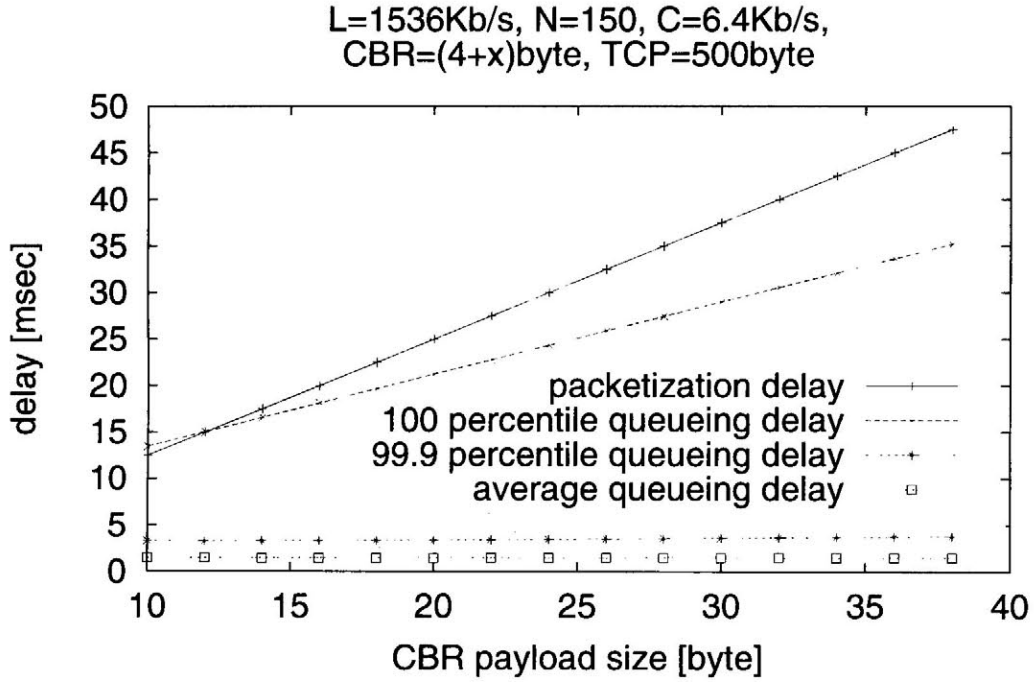


Figure 3.6: The impact of CBR frame size on the delay performance

Next we calculate the CBR payload size so as to minimize the end-to-end delay. Generally, the end-to-end delay consists of several parts: packetization delay, queueing delay, transmission delay, propagation delay and others. The CBR payload size affects the packetization delay, the queueing delay and processing delay. The processing delay is due to the computation time to compress the voice signal. For example, a

coder chip using digital signal processor (DSP) can process the voice signal within the frame delay. Likewise, four complete coder chips can reduce the processing delay to a quarter of the frame delay[Cox97]. In this way, processing delay depends directly on the coder's implementation. For simplicity, we do not consider the processing delay here.

The queueing delay occurs at each of the intermediate nodes. In the future Internet, most of the intermediate nodes have a high speed link of, say, gigabits per second so that the queueing delay is negligible. Nevertheless, access networks still employ low speed links. For this reason, the queueing delay occurring in the ingress node (an entrance point of the network) and the egress node (an exit point of the network) is not negligible, but should be evaluated precisely. In what follows, we focus on the sum of the queueing delay occurring in the network, which can be regarded as an ingress node or an egress node, and the packetization delay. This sum includes only part of the end-to-end delay. Here, we suppose that the above delay should be less than 30 ms for the required quality, and examine the number of acceptable streams under this condition. In Fig. 3.7, we illustrate the number of acceptable streams as a function of the CBR payload. For example, a 20-byte payload and a 24-byte payload lead to a packetization delay of 25 ms and 30 ms, respectively. Thus, a 24-byte payload is not acceptable, and the queueing delay should be less than 5 ms even in the former case. We note that the G.723.1 coder cannot be employed under this condition because the coder uses a 24-byte frame. From the figure, an 11-byte frame is the optimal for the 100-percentile delay bound, which allows 175 CBR streams. On the other hand, a 19-byte frame is optimal for the 99.9-percentile delay bound, which allows 198 streams.

Figure 3.8 illustrates the impact of CBR payload sizes on the number of acceptable streams under various delay bounds. A payload of 24 bytes cannot be employed in the G.723.1 standard with 30 ms delay bound, while 40 ms or more delay bounds allow a 24-byte payload, as shown in this figure. Note that the maximum number of acceptable streams is limited to 240 even with large payload size because 1536Kb/s : 6.4Kb/s= 240 : 1.

Next, the impact of the header length on the 99.9-percentile delay bound is illus-

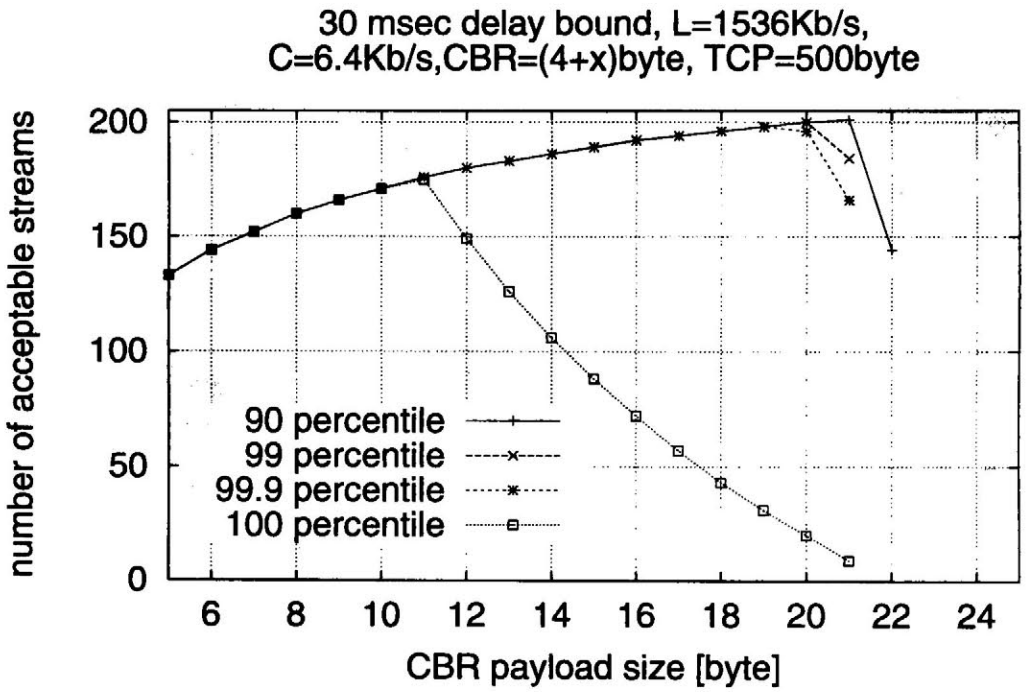


Figure 3.7: The impact of the payload size on the number of acceptable streams

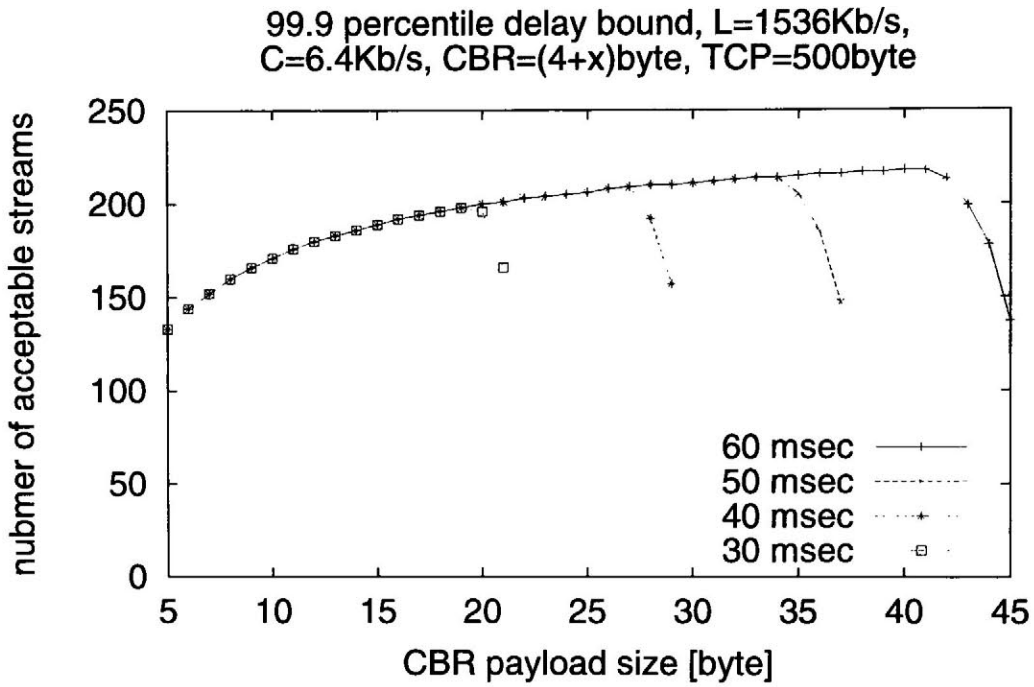


Figure 3.8: The impact of the delay bound on the maximum number of acceptable streams

trated in Fig. 3.9. The x-axis indicates the size of TCP packets. Seven voice streams share a link of 128 Kb/s, which can be provided by N-ISDN (Narrowband-Integrated Service Digital Network). Note that the delay time performance in the case of TCP packet of zero size is equivalent to that in the queue without vacations, which was computed by [HBH93] and in a similar way in Fig. 3.2. As we mentioned earlier, TCP packets of about 500 bytes are very common, and the uncompressed header is now dominant. As seen in the figure, this results in the delay time of more than 40 ms. Thus, in this case, TCP packet is better to be segmented into some shorter packets. We can see that segmentation and header compression techniques are effective to improve delay performance, successfully reducing the delay time within the acceptable level. Notice that there is a need for related processing capabilities in the routers, hence the effective throughput of TCP degrades with the decrease of their packet size due to segmentation. Nevertheless, Internet telephony can be operated at these performance levels even in an access network of low transmission capacity.

Finally, we show that header compression and segmentation of TCP packets can be an effective strategy to reduce the impact of TCP packet size on queueing delays on slow links. In Fig. 3.10, we provide the impact of TCP packet size on the queuing delay. We suppose L =33.6Kb/s modem link with some voice streams. For TCP packet size of 500 bytes, the delay is too long. On the other hand, small TCP packets or segmented TCP packets, for example 100 byte packets, significantly improve the delay performance. On this link, we can see that the number of voice streams has little impact on the queueing delay. Furthermore, the network can accept four voice streams by using segmentation and header compression, whereas it cannot accept even one voice stream without them (see Fig. 3.3).

3.5 Conclusions

We have investigated the delay performance of CBR traffic whose delay is affected by TCP traffic. The delay performance has been analyzed by solving the $nD/D/1$ queue with vacations. An exact and closed form solution has been obtained.

We have presented a number of numerical results. 1) CBR link utilization affects

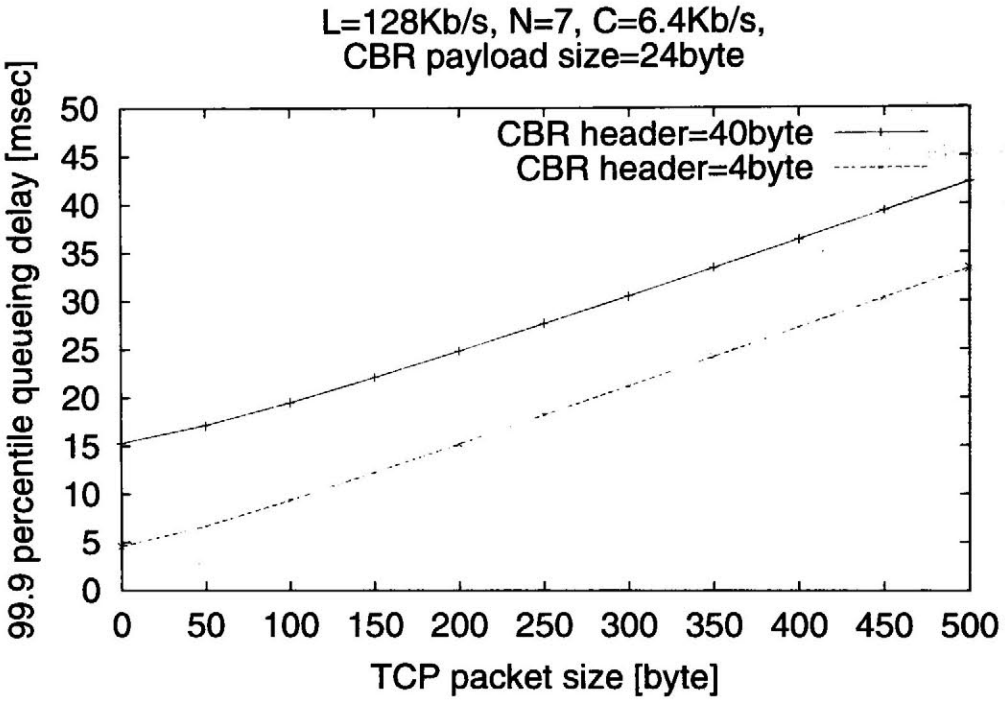


Figure 3.9: The impact of the header overhead on the 99.9-percentile delay bound

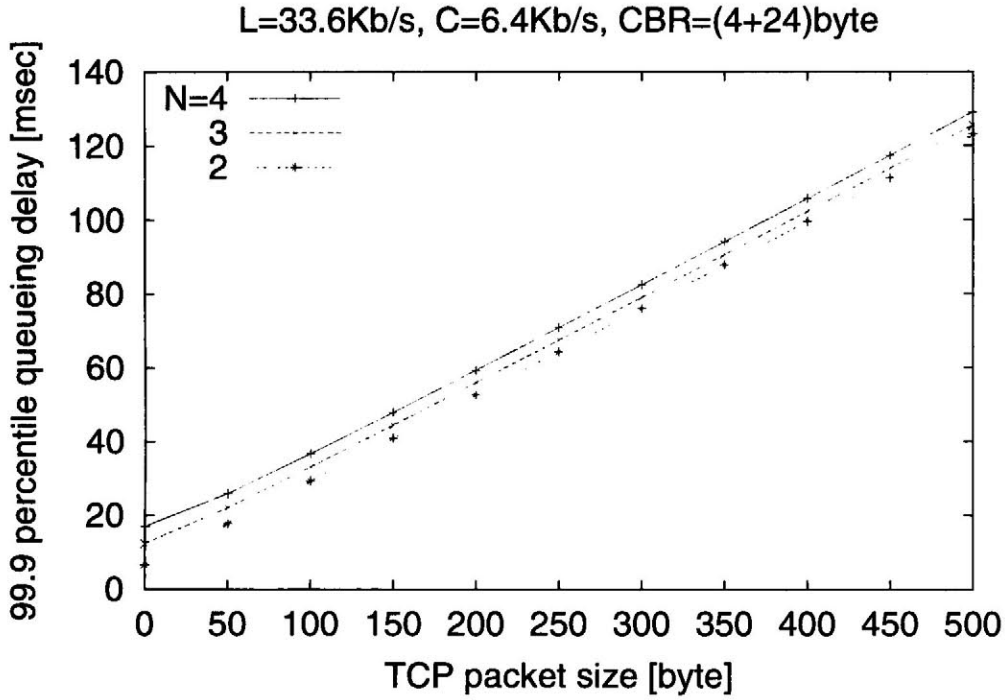


Figure 3.10: The impact of TCP packet size on the queuing delay

only slightly the 99.9-percentile delay bound while it heavily affects the 100-percentile delay bound. 2) At large link capacity, unlike the 100-percentile delay, the 99.9-percentile delay is small enough to be negligible. 3) The 99.9-percentile delay bound can accept larger number of streams than the 100-percentile delay bound when CBR payload size is large. 4) Header compression scheme can reduce delay to half in the N-ISDN networks. 5) Large TCP packets should be segmented into small size, such as 50 or 100 bytes, in extremely slow networks.

Appendix

3.A Proof of Lemma 1

We prove (3.3) by induction. Clearly, (3.3) holds for $k = 1$. Suppose the equation holds for some $k = k^*$ We then have

$$
\begin{aligned}
\tilde{B}^{(k^*+1)}(x) &= \int_0^{\min(x,b)} \tilde{B}^{(k^*)}(x-y)\frac{dy}{b} \\
&= \int_{x-b+\alpha_b(x)}^{x} \tilde{B}^{(k^*)}(y)\frac{dy}{b} \\
&= \frac{b-\alpha_b(x)}{b} \\
&\quad + \frac{1}{(k^*+1)!b^{k^*+1}} \sum_{n=1}^{k^*} \binom{k^*}{n} (-1)^{k^*-n} \left[\alpha_{nb}^{k^*+1}(x) - \{\alpha_{(n+1)b}(x) - \alpha_b(x)\}^{k^*+1}\right] \\
&= 1 - \frac{1}{(k^*+1)!b^{k^*+1}} \left[(k^*+1)!b^{k^*}\alpha_b(x) + \sum_{n=1}^{k^*} \binom{k^*}{n} (-1)^{k^*+1-n}\alpha_{nb}^{k^*+1}(x)\right. \\
&\qquad\qquad + \sum_{n=1}^{k^*} \binom{k^*}{n} (-1)^{k^*-n}\alpha_{(n+1)b}^{k^*+1}(x) \\
&\qquad \left. + \sum_{n=1}^{k^*} \binom{k^*}{n} (-1)^{k^*-n} \sum_{m=0}^{k^*} \binom{k^*+1}{m} (-1)^{k^*+1-m}\alpha_{(n+1)b}^{m}(x)\alpha_b^{k^*+1-m}(x)\right] \\
&= 1 - \frac{1}{(k^*+1)!b^{k^*+1}} \left[\sum_{n=1}^{k^*+1} \binom{k^*+1}{n} (-1)^{k^*+1-n}\alpha_{nb}^{k^*+1}(x)\right. \\
&\qquad\qquad + (k^*+1)!b^{k^*}\alpha_b(x) - (-1)^{k^*}\alpha_b^{k^*+1}(x) \\
&\qquad \left. + \sum_{n=1}^{k^*} \binom{k^*}{n} (-1)^{k^*-n} \sum_{m=0}^{k^*} \binom{k^*+1}{m} (-1)^{k^*+1-m}\alpha_{(n+1)b}^{m}(x)\alpha_b^{k^*+1-m}(x)\right]
\end{aligned}
$$

$$
\begin{aligned}
&= 1 - \frac{1}{(k^*+1)!b^{k^*+1}} \left[\sum_{n=1}^{k^*+1} \binom{k^*+1}{n} (-1)^{k^*+1-n} \alpha_{nb}^{k^*+1}(x) \right. \\
&\qquad + (k^*+1)! b^{k^*} \alpha_b(x) - (-1)^{k^*} \alpha_b^{k^*+1}(x) \\
&\qquad \left. + \sum_{n=1}^{k^*} \binom{k^*}{n} (-1)^n \sum_{m=0}^{k^*} \binom{k^*+1}{m} (-1)^{m+1} \left(\alpha_b(x) + nb\right)^m \alpha_b^{k^*+1-m}(x) \right] \\
&= 1 - \frac{1}{(k^*+1)!b^{k^*+1}} \left[\sum_{n=1}^{k^*+1} \binom{k^*+1}{n} (-1)^{k^*+1-n} \alpha_{nb}^{k^*+1}(x) \right. \\
&\qquad + (k^*+1)! b^{k^*} \alpha_b(x) - (-1)^{k^*} \alpha_b^{k^*+1}(x) \\
&\qquad + \sum_{l=1}^{k^*+1} \left\{ \sum_{n=1}^{k^*} \binom{k^*}{n} (-1)^n n^{k^*+1-l} \right\} \\
&\qquad\qquad \left. \cdot \left\{ \sum_{m=k^*+1-l}^{k^*} \binom{k^*+1}{m} \binom{m}{k^*+1-l} (-1)^{m+1} \right\} b^{k^*+1-l} \alpha_b^l(x) \right] \\
&= 1 - \frac{1}{(k^*+1)!b^{k^*+1}} \left[\sum_{n=1}^{k^*+1} \binom{k^*+1}{n} (-1)^{k^*+1-n} \alpha_{nb}^{k^*+1}(x) \right. \\
&\qquad + (k^*+1)! b^{k^*} \alpha_b(x) - (-1)^{k^*} \alpha_b^{k^*+1}(x) \\
&\qquad \left. + (k^*+1)!(-1)^{2k^*+1} b^{k^*} \alpha_b(x) - \sum_{m=0}^{k^*} \binom{k^*+1}{m} (-1)^{m+1} \alpha_b^{k^*+1}(x) \right] \\
&= 1 - \frac{1}{(k^*+1)!b^{k^*+1}} \sum_{n=1}^{k^*+1} \binom{k^*+1}{n} (-1)^{k^*+1-n} \alpha_{nb}^{k^*+1}(x),
\end{aligned}
$$

which shows (3.3) holds for $k = k^* + 1$, too. Note here that we use the equalities

$$
\binom{m-1}{n-1} + \binom{m-1}{n} = \binom{m}{n}, \tag{3.10}
$$

$$
\alpha_p(x - q + \alpha_q(x)) = \alpha_{p+q}(x) - \alpha_q(x),
$$

$$
\int_p^q \alpha_y^m(x) dx = \frac{-1}{m+1} \left(\alpha_y^{m+1}(q) - \alpha_y^{m+1}(p) \right),
$$

$$
\alpha_{(n+1)b}^m(x) \alpha_b(x) = \left(\alpha_b(x) + nb\right)^m \alpha_b(x),
$$

and for $m = 0, \ldots, k$

$$
\sum_{n=1}^{k} \binom{k}{n} (-1)^n n^m = \begin{cases} -1, & m = 0, \\ 0, & m = 1, \ldots, k-1, \\ k!(-1)^k, & m = k, \end{cases} \tag{3.11}
$$

in the above computation. Note that (3.11) can be shown to hold by

$$\sum_{n=1}^{k} \binom{k}{n} (-1)^n n^m = \lim_{z \to 0} \frac{d^m}{dz^m} \left[(1 - e^z)^k - 1 \right].$$

3.B Proof of Lemma 2

Note first that the right hand side of (3.9) is rewritten to be

$$\sum_{n=0}^{N-1} x_n (x_n y_n)^n \sum_{k=0}^{N-n-k} \binom{k+n}{n} (1 + x_n)^k = \sum_{k=0}^{N-1} \sum_{n=0}^{k} \binom{k}{n} x_n (x_n y_n)^n (1 + x_n)^{k-n}.$$

Thus we shall prove the following equality:

$$\sum_{k=0}^{N-1} \binom{N}{k+1} \sum_{n=0}^{k} \binom{k}{n} x_n^{k+1} y_n^n = \sum_{k=0}^{N-1} \sum_{n=0}^{k} \binom{k}{n} x_n (x_n y_n)^n (1 + x_n)^{k-n}. \tag{3.12}$$

Clearly (3.12) holds for $N = 1$. Suppose it holds for $N = N^*$. Then, by (3.10),

$$\begin{aligned}
&\sum_{k=0}^{N^*} \binom{N^*+1}{k+1} \sum_{n=0}^{k} \binom{k}{n} x_n^{k+1} y_n^n \\
&\quad = \sum_{k=0}^{N^*-1} \binom{N^*}{k+1} \sum_{n=0}^{k} \binom{k}{n} x_n^{k+1} y_n^n + \sum_{k=0}^{N^*} \binom{N^*}{k} \sum_{n=0}^{k} \binom{k}{n} x_n^{k+1} y_n^n \\
&\quad = \sum_{k=0}^{N^*-1} \sum_{n=0}^{k} \binom{k}{n} x_n (x_n y_n)^n (1 + x_n)^{k-n} + \sum_{n=0}^{N^*} \binom{N^*}{n} x_n (x_n y_n)^n (1 + x_n)^{N^*-n} \\
&\quad = \sum_{k=0}^{N^*} \sum_{n=0}^{k} \binom{k}{n} x_n (x_n y_n)^n (1 + x_n)^{k-n}.
\end{aligned}$$

The above equation implies that (3.12) holds for $N = N^* + 1$, too, which completes the proof.

Chapter 4

Delay Analysis of Voice and Video Traffic in Enterprise Access Network

4.1 Introduction

Multimedia communications will soon be established in enterprise networks, which convey voice, video, and data traffic in the same way, i.e., in a packet switching basis. This integration will enable new communication services and lead to cost-effective network management. On the other hand, it needs a way of controlling a quality-of-service (QoS) to meet a stringent requirement of real-time communications such as voice and video communications.

Enterprise networks will access the Internet with a number of technologies; for example, asymmetric digital subscriber line (ADSL), wireless local loop (WLL) and IMT-2000. These access technologies provide a broad range of link capacity from about 500 Kb/s to about 10 Mb/s with low costs. Even with this range of link capacity, low bit-rates video streams (e.g., from 50 Kb/s to 500 Kb/s) can be accommodated.

In this chapter, we examine delay performance of packets from constant bit rate (CBR) traffic like voice and video traffic whose delay characteristics can be affected

by non-real-time traffic. More specifically, our model has three types of traffic: two kinds of CBR streams (i.e., voice and video traffic), and non-real-time streams. At a multiplexer, all the CBR streams share a single buffer, whereas non-real-time streams share another distinct buffer. The CBR buffer has a non-preemptive priority over the non-real-time buffer. This strategy can be easily implemented and can minimize the effects of non-real-time traffic on the delay time of packets from CBR streams. We analyze the delay for both types of CBR packets by solving the $\sum D_i/G/1$ queue with vacations. The vacation time represents service time of a packet belonging to the non-real-time streams.

There are a number of past works analyzing delay time of queues with a superposition of CBR streams. The problem treated there is how to find the waiting time distribution of the $\sum D_i/D/1$ queue. Roberts and Virtamo have derived an exact formula for the queue length distribution [RV91]. Privalov and Sohraby have bounded the jitter distribution and moments in slotted networks such as asynchronous transfer mode (ATM) [PS98]. However, these researches do not deal with a case where service time (or packet length) is heterogeneous.

Moreover, we have already got an exact formula of the delay time distribution of the $nD/D/1$ queue with vacations [ITSO01b], but it was limited to the homogeneous case in terms of the rates of CBR traffic and the packet lengths. However, the enterprise networks handle multimedia traffic, so that transmitted packets can be different in their lengths and transmission rates; e.g., video packets are often larger than audio packets and their transmission rates are very different. Therefore, we will treat the heterogeneous case, in which two types of CBR traffic, i.e., long-interval class and short-interval class, share a buffer. In multimedia communications, traffic from different services share the network resources, so that one traffic can affect the quality of another traffic and vice versa. Through the numerical results, we show the impact of traffic parameters on the queueing delay time of voice and video communications. In summary of the results, 1) voice traffic gives a little impact on the delay time of video traffic, 2) packet length of video traffic has large influence on the delay time of voice traffic because length of video packets is ten or hundred times as large as voice packets, and 3) packet length of non-real-time traffic heavily affects the delay time of

both types of real-time packets.

The rest of this chapter is organized as follows. In Sect. 2, we describe the system treated here and its mathematical model. In Sect. 3, we analyze the model. In Sect. 4, we show some numerical results for the enterprise networks. Especially, we focus on the impact of packet length on the delay performance. In Sect. 5, we conclude the chapter.

4.2 System and Model Description

In this section, we provide the system description and the mathematical model used for the system analysis.

4.2.1 System Description

At a multiplexer, multiple CBR streams of short intervals and long intervals, and non-real-time streams share a bottleneck link as shown in Fig. 4.1. The packets from both kinds of CBR streams are required to be transmitted as fast as possible. Therefore, we use the priority scheduling to meet this requirement. The unused capacity left by CBR streams should be utilized effectively by non-real-time streams. All the CBR streams share a single buffer, whereas non-real-time streams share a distinct buffer. The CBR buffer has a non-preemptive priority over the non-real-time buffer.

One of our main objectives is to derive the delay bound of CBR packets that is tight and suitable for admission control. Therefore, we assume the CBR buffer to be of infinite size in order to get the worst-case bound.

Most of the non-real-time traffic is carried by transmission control protocol (TCP) on the Internet. A congestion control mechanism of TCP is placed at the end-nodes, so that TCP sources try to get as much link capacity as possible. Generally speaking, a number of packets from TCP sources are in transit at intermediate-nodes. For this reason, we assume the non-real-time buffer to be saturated in our system so that at least one packet always exists in the buffer. This is the worst-case scenario for the

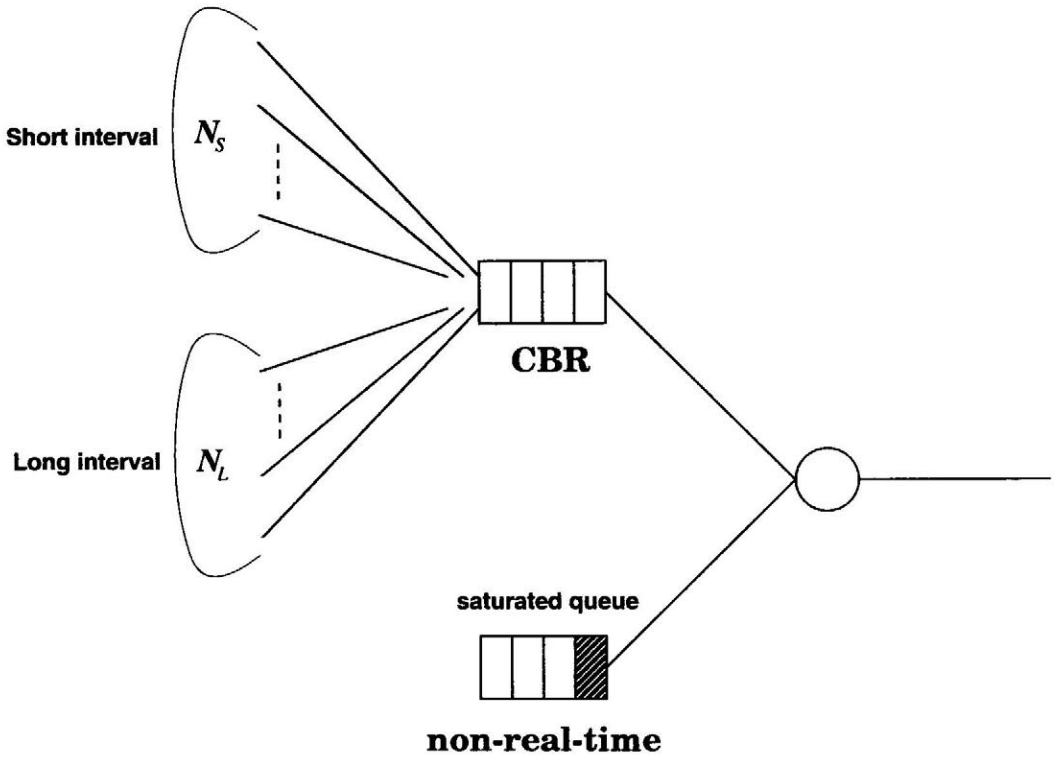


Figure 4.1: System overview.

CBR delay, considering the non-preemptive priority.

On the CBR side, there are N_S streams of short intervals and N_L streams of long intervals. These two types of streams are heterogeneous in terms of interarrival times and packet lengths.

4.2.2 Mathematical Model

We consider a single server queue with a superposition of N independent input streams. Arrival streams are classified into two types: short-interval class and long-interval class. Arrivals from each short-interval (resp. long-interval) class stream are characterized by an equilibrium renewal process with an interarrival time of D_S (resp. D_L) (deterministic). This means that the first arrival from each short-interval (reps. long-interval) class stream occurs uniformly in the interval $(0, D_S)$ (resp. $(0, D_L)$) and is independent of other streams. We assume here that there exists a positive integer M such that $D_L = MD_S$. Let N_S and N_L denote the number of short-interval and long-interval class streams, respectively. Note that $N = N_S + N_L$. Service times of short-interval (reps. long-interval) class are independent and identically distributed with a distribution function $B_S(x)$ (resp. $B_L(x)$). Let b_S and b_L denote the mean service times of short-interval class and long-interval class, respectively. We assume that service times of both classes are bounded, and we denote the maximum service time of short-interval (resp. long-interval) class by $\overline{b_S}$ (resp. $\overline{b_L}$). Further we assume that $\overline{b_S}N_S + \overline{b_L}N_L \leq D_S$. Thus, there exist real numbers R_S and R_L such that each service time of short-interval (resp. long-interval) class is less than R_S/N_S (resp. R_L/N_L) and $R_S + R_L = D_S$. The service is a non-preemptive and the service order of arrivals follows the first-come first-serve. We call the related queue the ordinary queue, where the server is idle when the queue is empty.

Next, we consider the queue with vacations, where arrivals and services have the same characteristics as in the ordinary queue. In the queue with vacations, the server takes a vacation when the queue becomes empty. Vacation times are independent and identically distributed according to a distribution function $U(x)$ with finite mean u. If the server finds waiting customers in the queue on his/her return from a vacation,

he/she serves customers continuously until the queue becomes empty, and then takes the next vacation. This vacation discipline is called exhaustive service. If the server finds no customers on his/her return from a vacation, he/she takes another vacation. This vacation discipline is called multiple vacations.

Remark Two types of input stream correspond to long-interval or short-interval CBR streams. Vacation times correspond to the service times of non-real-time streams. To investigate the worst-case scenario of both types of CBR packets that has a (non-preemptive) priority over the non-real-time packets, we assume the service of a non-real-time packet starts whenever the server becomes idle.

4.3 Analysis

Let $W_S^*(s)$ (resp. $W_L^*(s)$) denote the Laplace-Stieltjes transforms (LST) of the actual waiting time distribution of short-interval (resp. long-interval) class in the queue with vacations. Owing to the stochastic decomposition property of $G/G/1$ queue with exhaustive and multiple vacations [Dos90, Tak96], we have

$$\begin{aligned} W_S^*(s) &= \frac{1-U^*(s)}{su} W^{(S)}(s), \\ W_L^*(s) &= \frac{1-U^*(s)}{su} W^{(L)}(s), \end{aligned} \tag{4.1}$$

where $U^*(s)$ denotes the LST of $U(x)$ and $W^{(S)}(s)$ (resp. $W^{(L)}(s)$) denotes the LST of the actual waiting time distribution of short-interval (resp. long-interval) class in the ordinary queue. In what follows, we consider $W^{(S)}(s)$ and $W^{(L)}(s)$.

If the ordinary queue has only short-interval class streams, it follows from the result in [Sen90] that the virtual and the actual waiting time processes have the regenerative cycles of length D_S. Since we assume $D_L = MD_S$ with a positive integer M and each service time of short-interval (reps. long-interval) class is less than R_S/N_S (resp. R_L/N_L) with $R_S + R_L = D_S$, the virtual and the actual waiting time processes in the ordinary queue have the regenerative cycles of length D_L. We divide each regenerative cycle into M sub-cycles of length D_S and call the mth ($m = 1, \ldots M$) sub-cycle in the kth ($k = 1, 2, \ldots$) regenerative cycle the (k, m) sub-cycle. Since

$R_S + R_L = D_S$, it can be shown that all the virtual and actual waiting time processes in the (k, m) sub-cycles $(k = 1, 2, \ldots)$ are identical. Namely, the sample paths of the virtual and actual waiting times in the $(k+1, m)$ sub-cycle are identical to those in the (k, m) sub-cycle, where $k = 1, 2, \ldots$.

Let $N_L(m)$ $(m = 1, \ldots, M)$ denote the number of long-interval class streams that arrives in the (k, m) $(k = 1, 2, \ldots)$ sub-cycle. Recall that long-interval class streams are characterized by equilibrium renewal processes with an interarrival time of $D_L = MD_S$. Therefore we have

$$\Pr\left(N_L(1) = n_1, \ldots, N_L(M) = n_M\right) = \frac{N_L!}{M^{N_L} n_1! \cdots n_M!},$$

where $n_m \geq 0$ $(m = 1, \ldots, M)$ and $n_1 + \cdots + n_M = N_L$. Further, given $N_L(m) = n_m$, the arrival epoch of each of n_m long-interval class streams is uniformly over the interval in the (k, m) sub-cycle and independent of others. Thus the waiting times of short-class and long-class streams can be analyzed by the queueing model having N_S arrival streams with service time distribution $B_S(x)$ and n_m arrival streams with $B_L(x)$, each of which is characterized by an equilibrium renewal process with deterministic renewal interval of length D_S. Let $W_n^{(S)}(s)$ (resp. $W_n^{(L)}(s)$) denote the LST of the actual waiting time distribution of an arrival stream with $B_S(x)$ (resp. with $B_L(x)$), given N_S arrival streams with $B_S(x)$ and n arrival streams with $B_L(x)$. We then have

$$W^{(S)}(s) = \sum_{\substack{n_1 + \cdots n_M = N_L \\ n_m \geq 0}} \frac{N_L!}{M^{N_L} n_1! \cdots n_M!} \sum_{m=1}^{M} \frac{W_{n_m}^{(S)}(s)}{M},$$

$$W^{(L)}(s) = \sum_{\substack{n_1 + \cdots n_M = N_L \\ n_m \geq 0}} \frac{N_L!}{M^{N_L} n_1! \cdots n_M!} \sum_{m=1}^{M} \frac{n_m W_{n_m}^{(L)}(s)}{N_L}.$$

With the result in the Appendix, $W_n^{(S)}(s)$ and $W_n^{(L)}$ are given by

$$W_n^{(S)}(s) = V_{N_S - 1, n}^*(s) \quad (n = 0, \ldots, N_L),$$

$$W_n^{(L)}(s) = \begin{cases} 1 & (n = 0), \\ V_{N_S, n-1}^*(s) & (n = 1, \ldots, N_L), \end{cases}$$

where $V_{i,j}^*(s)$ $(i, j = 0, 1, 2, \ldots)$ is recursively obtained by

$$V_{0,0}^*(s) = 1,$$

$$
\begin{aligned}
V^*_{i,0}(s) &= 1 - \frac{ib_S}{D_S} + \frac{ib_S}{D_S}\frac{1-B^*_S(s)}{sb_S}V^*_{i-1,0}(s) \\
&= \sum_{k=0}^{i}\frac{i!}{(i-k)!}\left(\frac{1-B^*_S(s)}{sD_S}\right)^k\left(1-\frac{(i-k)b_S}{D_S}\right) \quad (i=1,2,\ldots), \\
V^*_{0,j}(s) &= 1 - \frac{jb_L}{D_S} + \frac{jb_L}{D_S}\frac{1-B^*_L(s)}{sb_L}V^*_{0,j-1}(s) \\
&= \sum_{k=0}^{j}\frac{j!}{(j-k)!}\left(\frac{1-B^*_L(s)}{sD_S}\right)^k\left(1-\frac{(j-k)b_L}{D_S}\right) \quad (j=1,2,\ldots), \\
V^*_{i,j}(s) &= 1 - \frac{ib_S}{D_S} - \frac{jb_L}{D_S} + i\frac{1-B^*_S(s)}{sD_S}V^*_{i-1,j}(s) \\
&\qquad + j\frac{1-B^*_L(s)}{sD_S}V^*_{i,j-1}(s) \quad (i,j=1,2,\ldots).
\end{aligned}
$$

4.4 Numerical Results

In this section, we will show our numerical results obtained through our analysis of the delay time distribution. The rest of this section is organized as follows. First, we explain low bit-rates voice and video coding algorithms to determine parameters for numerical experiments. Next, we give descriptions regarding both the parameters and the Jagerman method for the numerical Laplace inversion. We then show the interactions between voice streams and video streams, the impact of link capacity on the statistical delay, and effectiveness of a header compression and small TCP packets by numerical results.

4.4.1 Voice and video coding algorithms

Interactive voice and video communications on the Internet, especially in low capacity of transmission links, require some performance demands; error robustness and low bit-rates with acceptable quality. In the packet networks, there are some packet losses and a limit of the link capacity, thus the coding algorithms should endure such inadequate communication environments. In this dissertation, we focus on G.723.1 for voice communication and H.263+ for video communications.

For voice communications or *voice over IP* (VoIP), a number of coding algorithms are standardized. Above all, G.723.1 [ITU95, Cox97] is one of the promising algorithms for VoIP because of its low coding rate (6.4 Kb/s or 5.3 Kb/s) with high quality and its error robustness. The Mean Opinion Score (MOS) of G.723.1 is 3.98 while the MOS of ordinary G.711 (64 Kb/s) is 4.0 [Hel98, p. 111]. Moreover, speech quality of G.723.1 transmission is little affected by random packet loss rates of up to 10 percents [K+98].

Meanwhile, video conferencing and videophone are examples of video communications, which play an important role in business scenes. However, the video communications are not popular currently yet because of their low quality at high cost. Thus, we should reduce costs while keeping the video quality. As mentioned earlier, we keep our eyes on H.263+ [ITU98, Rij96, CEGK98, Sch99]. H.263+ or *"video coding for low bitrate communication"* is formally known as H.263 version 2, which is standardized in 1998 by the International Telecommunication Union (ITU).

According to some researches, there are a number of methods to increase the error resilience of H.263+ (or H.263) under some packet losses. In [WLSC98], Willebeek-LeMair et al. have proposed the robust H.263 video coding, which intelligently and frequently updates macroblocks based on their impacts on future frames. This increases the error resilience because some important macroblocks are frequently updated. Wenger et al. have described two methods for supporting the error resilience [WKOK98, Wen97]. Both of the methods use *reference picture selection mode - Annex N* of H.263+. The reference picture selection (RPS) mode allows the codec to produce and decode the inter-frame compressed pictures based not only on the last transmitted picture but also on earlier transmitted pictures. The first method of using RPS employs the back-channel of the video transmission (for example using Real-Time Control Protocol (RTCP) [SCFJ96]) to notify positive or negative acknowledgments. By using this information, the sender can keep track of the last correctly received picture at the receiver. This method is effective but cannot be applied for point-to-multipoint communications. For the multicast communications, video redundancy coding (VRC) or the second method is applicable. VRC creates two or more threads of I-picture (or inter-frame picture) sequences. This allows us

to keep the picture quality when a packet loss occurs, because a packet loss damages only one thread of I-picture sequences. Using the RPS with the back-channel method or the VRC method, the video quality can be kept even in packet loss rates of up to 10 or 20 percents. In another research of Wenger et al., an encoding method of two packets per one picture, has been evaluated [WC99]. This also allows packet loss rates of up to 20 percents.

4.4.2 Description of parameters and Laplace inversion

Table 4.1: Selected parameters for numerical experiments.

Link capacity	384 Kb/s, 512 Kb/s, 640 Kb/s, 1 Mb/s, 1.536 Mb/s, 2 Mb/s, 4 Mb/s, 6 Mb/s
voice stream	30-ms interval and 24-Byte payload (17.1 Kb/s, without header compression, 7.47 Kb/s with header compression)
video stream	60-ms interval and 250, 500, 750, 1000 and 1500-Byte payload (38.7 Kb/s, 72.0 Kb/s, 105 Kb/s, 139 Kb/s and 205 Kb/s, without compression)

In this subsection, we give parameters for numerical experiments, for example, link capacity, stream bandwidth as well as packet length.

We consider the link capacity ranging from 384 Kb/s to 6 Mb/s because we are considering the enterprise networks. As mentioned in the introduction, some novel access technologies will be available, for example ADSL, WLL and IMT-2000. These access technologies will introduce broad bandwidth and low cost. This range of link capacity can accept a number of voice and video streams but the delay time cannot be accepted occasionally. We treat the impact of link capacity on the delay time in Sect. 4.4.4.

For voice coding algorithm, we choose G.723.1, as described earlier. Thus, the bandwidth of voice stream is 6.4 Kb/s, the frame lengths or the payload lengths of voice stream are 24 Bytes and the interarrival time is 30-ms [ITU95]. In Internet Protocol (IP) version 4 [Ste94, p. 34], there is a 40-Byte IP/User Datagram

Protocol/Real-Time Protocol (IP/UDP/RTP [SCFJ96]) header. Therefore, a 64-Byte (equals a 40-Byte header and a 24-Byte payload) packet is generated in every 30-ms interval. This means the bit-rate of each voice stream at the network layer becomes about 17.1 Kb/s. Furthermore, if a compressed header of 4 bytes is used, the bit-rate of each voice stream is equal to 7.47 Kb/s. The header compression is explained later in detail in this section.

As a video coding, we employ the H.263+ coding algorithm [Rij96]. Unlike in voice coding, there is no standard regarding the frame length, the bit-rate and the interval time. In this dissertation, we mainly treat the quarter common intermediate format (QCIF), in which a frame of 176 times 144 in resolution and 15 frames are generated per second. Frames are transmitted at a constant bit rate, i.e., CBR. The resulting bit rate becomes about 50 Kb/s to 250 Kb/s due to variable packet length. Our target of video quality is for video conferencing and videophones.

Moreover, one packet is made by each picture frame because of our analytic constraint. As a result, packets are generated in every 66.7-ms interval. We should choose the interarrival time as the multiple of 30-ms (or interval time of voice streams) due to an analytic condition. Consequently, we employ the interarrival time of 60-ms instead of 66.7-ms. Thus, the frame rates become 16.7 frames per second (fps). Furthermore, the frame is of 250, 500, 750, 1000 or 1500 Bytes in length. Thus, packet lengths including a 40-Byte header are 290, 540, 790, 1040 or 1540 Bytes, and bit-rates are within a range of 38.7 Kb/s to 206 Kb/s.

From the above parameters, packets of 64 Bytes from a voice stream are generated in every 30-ms, and packets from a video stream in every 60-ms, thus voice traffic is in a short-interval class and video traffic in long-interval class. In summary, we give the related characteristics in Table 4.1. Furthermore, we will treat RTP header compression [CJ99] where a 40-Byte header can be reduced to a 4-Byte one.

For non-real-time traffic, we are focusing on TCP streams. Lengths of the TCP packets are fixed at either 250, 500, 750, 1250 or 1500 Bytes.

To this end, we can specify the distribution for CBR service time and non-real-time service time. Let $B_S(t)$ and $B_L(t)$ denote the service time distribution for short-interval and long-interval CBR, and $U(t)$ denote non-real-time service time

distribution. In our numerical experiments, short-interval and long-interval CBR packet length and non-real-time packet length are fixed at b_S, b_L and u, respectively. Therefore, $B_S^*(t)$, $B_L^*(t)$ and $U^*(t)$, which are LST of $B_S(t)$, $B_L(t)$ and $U(t)$, are given as

$$B_S^*(s) = e^{-sb_S},$$
$$B_L^*(s) = e^{-sb_L},$$
$$U^*(s) = e^{-su}.$$

Next, we calculate $W_S(t)$ and $W_L(t)$, which are the delay time distribution for short-interval CBR and long-interval CBR, by means of Laplace inversion of $W_S^*(s)$ and $W_L^*(s)$. We employ Jagerman's method [Jag82] for the Laplace inversion, which offers an excellent numerical approximation for the Laplace inversion.

Suppose that $F(s)$ indicates a Laplace transform of $f(t)$. $\sigma_n(t)$ denotes a numerical approximation of $f(t)$ derived by means of Jagerman's method, which can be obtained as follows:

$$\sigma_n(t) = e^{-\alpha t}\left\{\left(2+\frac{1}{n}\right) f_{2n}(t) - \left(1+\frac{1}{n}\right) f_n(t)\right\},$$

where $f_n(t)$ is given by the following equation.

$$f_n(t) \cong \frac{n+1}{tqr^n}\sum_{j=1}^{q} e_q(-nj) F\left\{\frac{n+1}{t}\left[1 - re_q(j)\right]\right\},$$

$$e_q(x) = e^{i\frac{2\pi x}{q}},$$

where i is the imaginary unit. r and q should be determined for each $f_n(t)$ in a way to give a good approximation, in which r satisfies $0 < r < 1$, and q is a prime number satisfying $q > n$ for natural number n. Note that r and q for $f_{2n}(t)$ can be different from them for $f_n(t)$. We use 64-bits floating point numbers for this calculation, and then parameterize r and q as follows.

	r	q
$n = 50$	0.86	229
$2n = 100$	0.937	541

We then choose 0.20 for α. In Fig. 4.2, we give the comparison between our analysis and simulation results. The simulation experiments were performed 10^7 times. Since

numerical Laplace inversion is an approximate solution, the analysis and the simulation results cannot agree perfectly. However, we can say that our analysis gives a good approximation as shown in this figure. Note that the figure contains a caption of voice (17.1Kb/s, 64Byte) $\star$ 70, video (139Kb/s, 1040Byte) $\star$ 1. This means that a superposition of 70 voice streams and one video stream arrives at a multiplexer. This '$\star$' notation will be used in the rest of figures in this section.

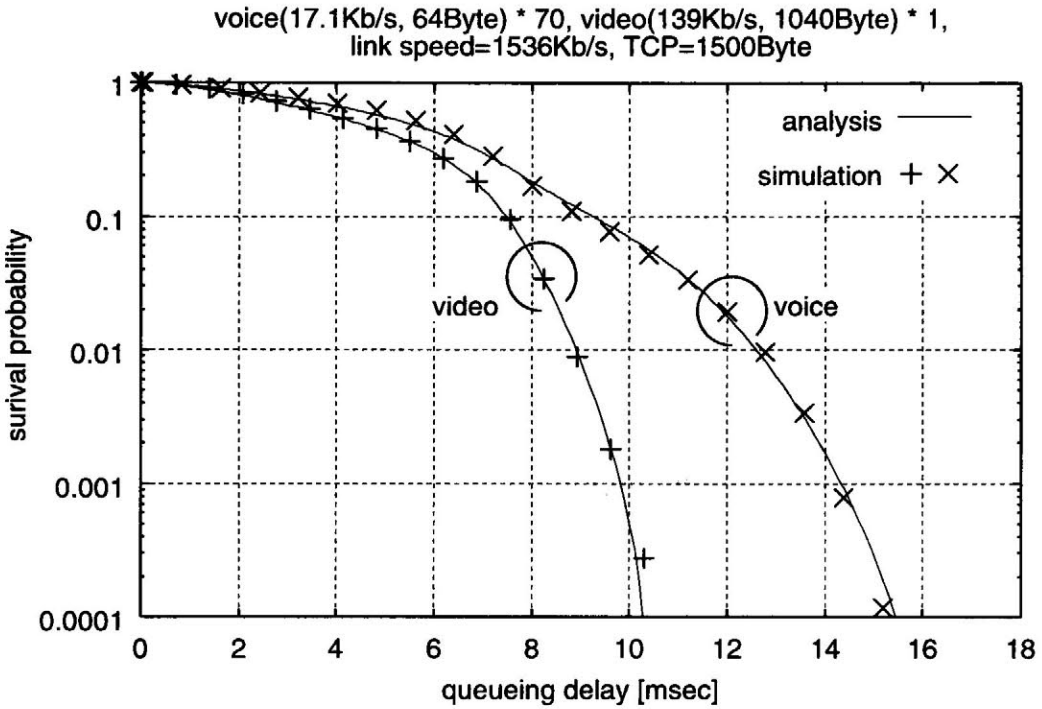
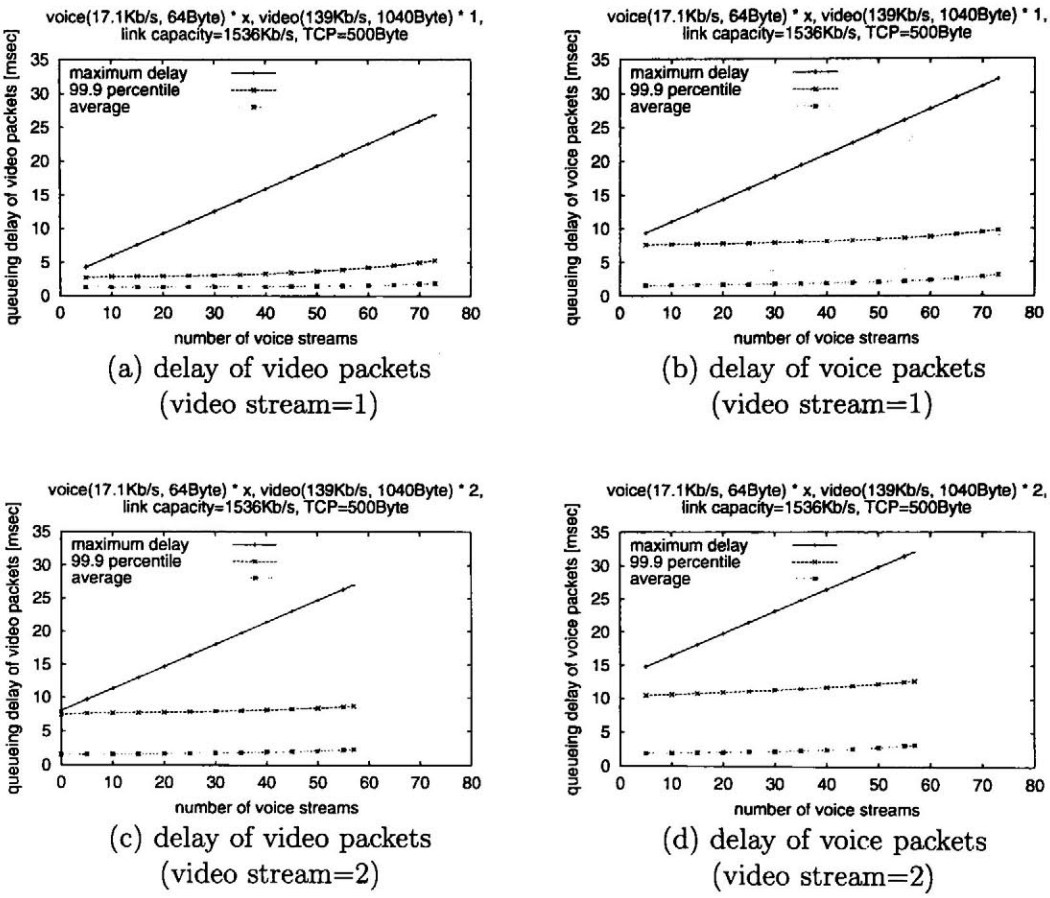


Figure 4.2: Comparison between analysis and simulation results.

4.4.3 Interaction between voice streams and video streams

First, we give some numerical results to understand interactions between voice streams and video streams. In this dissertation, packets from voice streams have the same priority as packets from video streams. Therefore, one type of streams affects the delay time of the other type of streams.



(a) delay of video packets (video stream=1)

(b) delay of voice packets (video stream=1)

(c) delay of video packets (video stream=2)

(d) delay of voice packets (video stream=2)

Figure 4.3: Impact of the number of voice flows on CBR delay.

In Fig. 4.3, we give the impact of the number of voice streams on queueing delay of both kinds of CBR streams. We set the link capacity to 1536 Kb/s and the TCP packet length to 500 Bytes in this figure. Figure 4.3(a) and Fig. 4.3(b) show the queueing delay time of video packets and of voice packets, respectively. Both figures are obtained in case that there is one video stream. Moreover, Fig. 4.3(c) (resp. Fig. 4.3(d)) describes the delay time of video packets (resp. of voice packets) in case of two video streams. In all of these figures, the maximum delay is heavily affected by the number of voice streams. The maximum delay of short-interval CBR (voice), and of long-interval CBR (video) can be described as follows. For short-interval CBR packets, it is

$$(N_S - 1) \times b_S + N_L \times b_L + u,$$

and for long-interval CBR packets, it becomes

$$(N_L - 1) \times b_L + N_S \times b_S + u,$$

where N_S (resp. N_L) denotes the number of short-interval (resp. long-interval) CBR steams, b_S (resp. b_L) denotes short-interval (resp. long-interval) CBR service time, and u indicates the TCP service time as mentioned earlier. Note that the maximum delay is the worst case delay in other words.

On the other hand, the 99.9-percentile delay times, which are obtained from our analysis, are almost insensitive to the number of voice streams. Therefore, if CBR traffic can tolerate some loss, say 0.1 percent loss, due to late arrival, the 99.9-percentile bound can be employed in a call admission control (CAC) and will be more effective in using link capacity efficiently in comparison with the maximum delay, in particular in a case with a large number of voice streams. The average time is added just for reference.

The delay time of voice streams is always larger than the delay time of video streams. For example, delay in Fig. 4.3(b) is larger than delay in Fig. 4.3(a) and delay in Fig. 4.3(d) is larger than delay in Fig. 4.3(c). We explain the reason later in this subsection.

Figure 4.3(a) and Fig. 4.3(c) also include the delay time in case of no voice streams at the leftmost point. A case of no voice streams is equivalent to the homogeneous

CBR traffic case; thus the result was obtained from our previous analysis [ITSO01b]. In case of one video stream, the 99.9-percentile delay of zero voice streams and of 73 voice streams are 2.60 ms and 5.26 ms. In case of two video streams, the 99.9-percentile delay of zero voice streams and of 57 voice streams are 7.46 ms and 8.70 ms. Therefore, we can conclude that the number of voice streams gives a little impact on the delay of video streams.

We then show the impact of the number of video streams on queueing delay of voice streams. Figure 4.4 treats a case of link capacity equal to 1536 Kb/s and Fig. 4.5 treats a case of 640 Kb/s. From both figures, the 99.9-percentile delay is heavily affected by the video streams, compared with the impact of voice traffic on video delay.

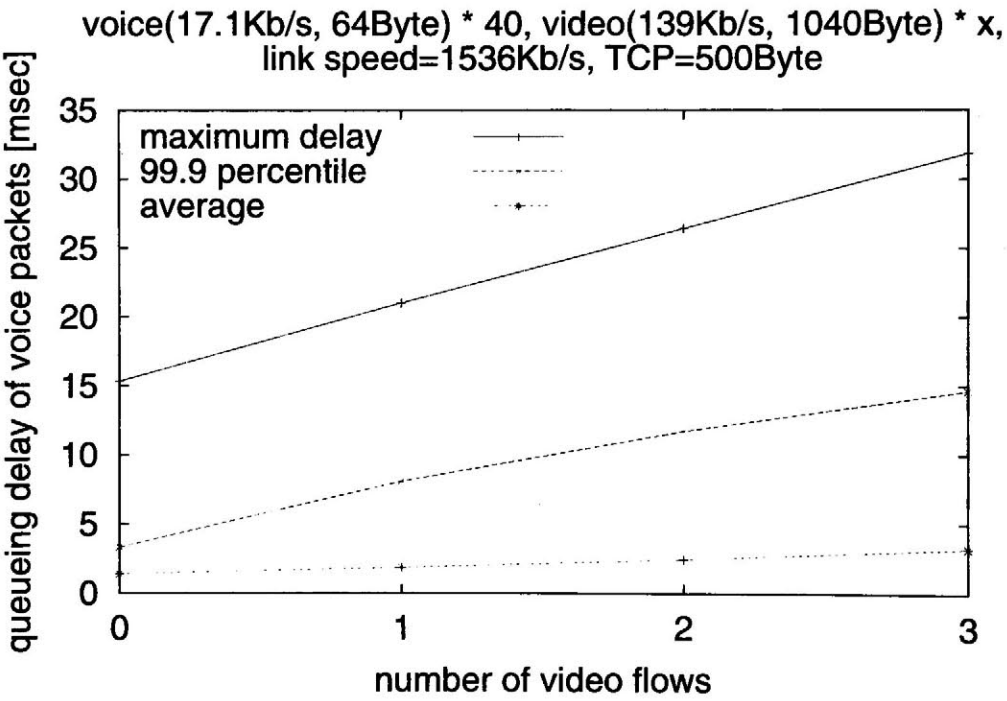


Figure 4.4: Impact of the number of video streams on voice delay (1536 Kb/s).

To understand video impacts on voice delay, we give three figures. Figure 4.6

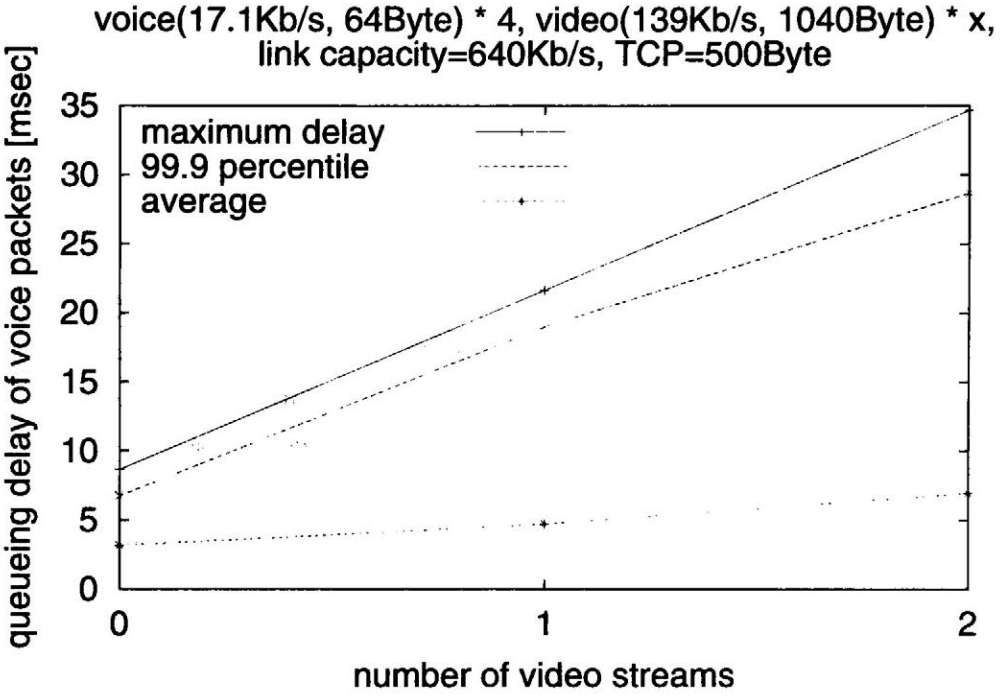


Figure 4.5: Impact of the number of video streams on voice delay (640 Kb/s).

shows the impact of video rates, Fig. 4.7 indicates the impact of the number of video frames per second (fps), and Fig. 4.8 treats the impact of packet length of video streams. In the first figure, the fps is fixed at rates of 16.7, thus the packet length varies with the transmission rates. On the other hand, in the second figure, the packet length is fixed at 1040 Bytes. From these two figures, voice delay is a little affected by the fps, while the increase of the video rates from 38.7 kb/s to 205 kb/s increases the queueing delay time of voice packets by about seven ms. This seems due to long video packets.

To show this observation, we lastly discuss the impact of packet length of video streams in Fig. 4.8. In this figure, the video transmission rates at the application layer are fixed at 133 Kb/s. However, since the fps is varied, the transmission rates at the network layer are varied due to addition of a header of constant length to each frame. For example, packets of 290 Bytes are generated at a rate of 66.7 packets per second, and each of them includes a 40-Byte header. As a result, the transmission rates at the network layer become 155 Kb/s in this example. We can see that length of video packets gives no impact of video delay because there is only one video stream, while voice delay is heavily affected by the length of video packets. Thus, we conclude that large length of video packets increases voice delay.

Generally speaking, voice traffic has a stringent requirement on the transmission delay time compared with video traffic. However, packet length of video traffic gives a large impact of voice delay as mentioned in this subsection. If we transmit both voice traffic and video traffic in the same network, video packets should be so small enough to satisfy a requirement on voice delay.

We note that if Weighted Fair Queueing (WFQ) was used instead of priority scheduling, deterministic delay bound of a stream is not affected by packets belonging to the other streams [Par93, PG93, PG94]. In other words, the delay bound of voice packets is not affected by video packets. However, in the parameter setting of this subsection, the deterministic delay bound of WFQ is mostly longer than that of priority scheduling because the bound of WFQ includes packet interarrival time of dozens of ms, e.g., 30-ms or 60-ms, whereas that of PQ is the sum of the products of packet service time and the number of arrival streams in each class, as shown

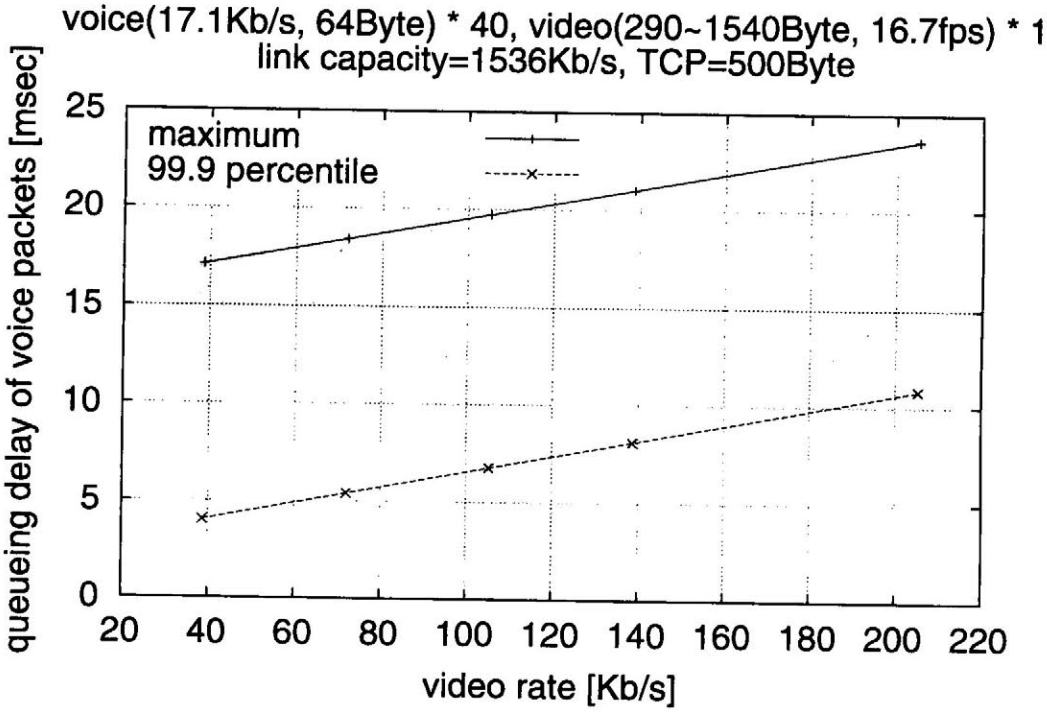


Figure 4.6: Impact of video rates on voice delay.

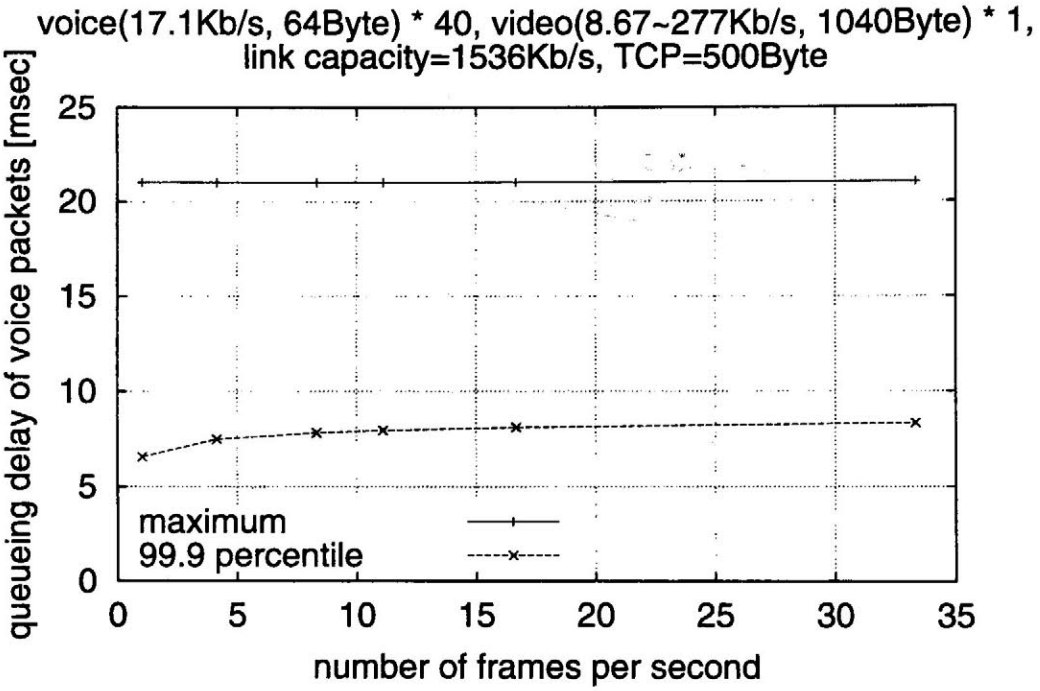


Figure 4.7: Impact of video frame rates on voice delay.

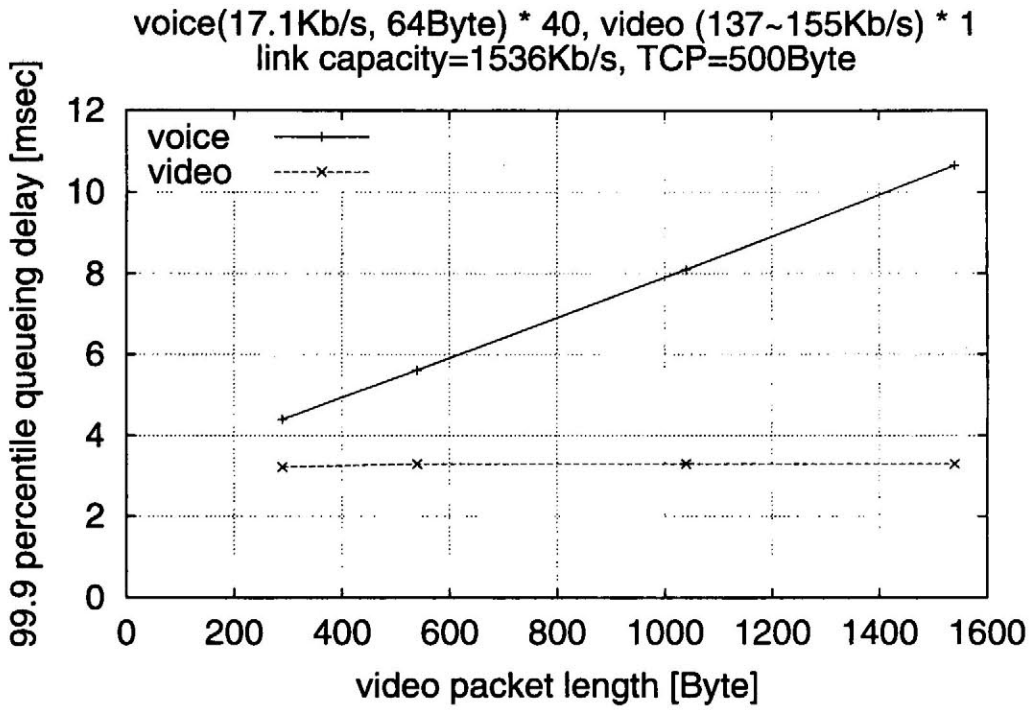


Figure 4.8: Impact of video packet lengths on CBR delay.

earlier; packet interarrival time is always greater than or equal to the products of packet service time and the number of arrival streams. Moreover, WFQ requires higher implementation costs than priority scheduling. For this reason, we employ the priority scheduling in this dissertation.

4.4.4 Impact of link capacity on CBR delay

In this subsection, we treat the impact of link capacity on CBR delay. Figure 4.9 gives the impact link capacity on the CBR delay time. The x-axis indicates link capacity, and the y-axis shows the queueing delay time. In these experiments, one video stream and a number of voice streams share a buffer. This is indicated by "$\star$ X", which means the number of voice streams is chosen in a way that those voice streams occupy all the link capacity for each value of the x-axis. From the figure, the maximum delay of both video traffic and voice traffic is always greater than 30 ms. On the other hand, the 99.9-percentile delay, i.e., the statistical delay is less than 5 ms at link capacity greater than or equal to 6 Mb/s.

ITU-T G.114 describes that communication quality is good as long as the total end-to-end delay is less than 150 ms. Thus, in case of large link capacity such as 6 Mb/s, queueing delay is of little concern from a practical point of view.

In case of thin links, queueing delay becomes large even in the 99.9-percentile delay. For example, in a link of 384 Kb/s, the 99.9-percentile delay of voice traffic (resp. video traffic) becomes 53.3 ms (resp. 34.4 ms). This is because service time of video packets becomes very large (e.g., 21.7 ms at a rate of 384 Kb/s). Moreover, only a few number of voice streams can be accommodated due to the small link capacity. In this context, the statistical delay bound becomes close to the maximum delay, and thus a CAC based upon the statistical delay bound is not effective for extremely small link capacity such as 384 Kb/s.

From another point of view, we show the effect of link capacity on queueing delay. In Fig. 4.10, we investigate the impact of link capacity on the number of acceptable streams with some delay requirements. The x-axis shows link capacity, and the y-axis indicates either the number of voice streams or that of video streams, which is

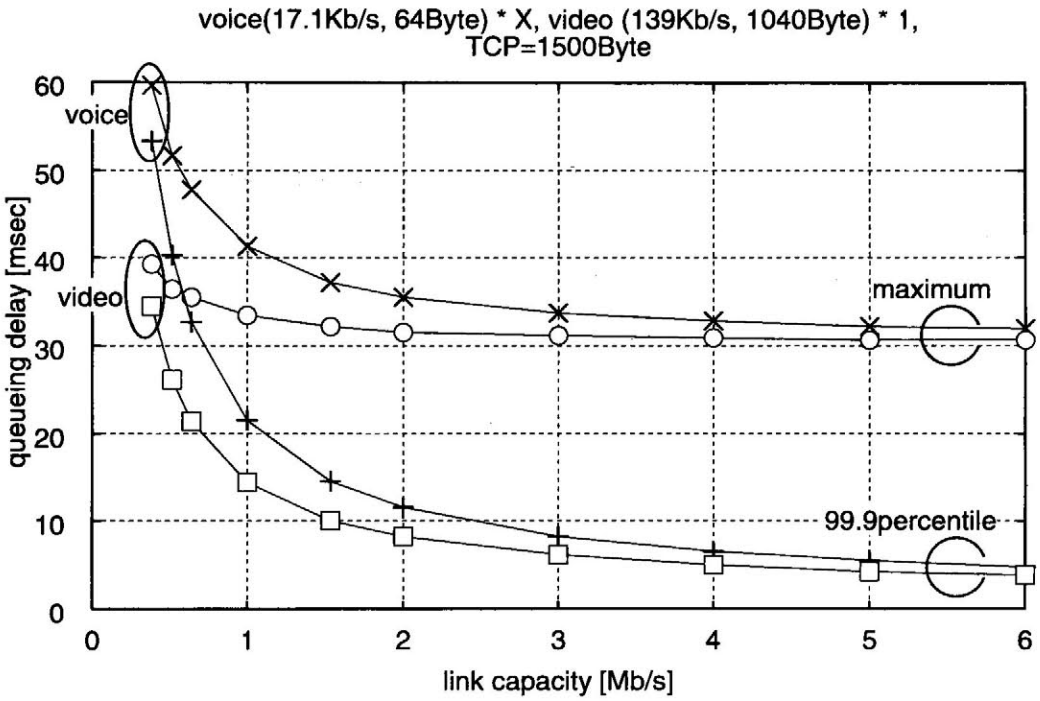


Figure 4.9: Impact of link capacity on CBR delay.

described as "⋆ y" in captions of the figure. Figure 4.10(a) and Fig. 4.10(b) show the number of acceptable video streams where there are 10 voice streams, whereas Fig. 4.10(c) and Fig. 4.10(d) give the number of acceptable voice streams where there are two video streams. The voice delay requirement is 10 ms in Fig. 4.10(a) and Fig. 4.10(c), while it is 20 ms in other two figures.

In case of the 99.9-percentile delay bound, the number of acceptable voice streams is about twice as large as that of the maximum delay bound over a wide range of link capacity. More precisely, the 99.9-percentile bound has an advantage over the maximum delay bound when the link capacity is greater than approximately 1.5 Mb/s in Fig. 4.10(a), 640 Kb/s in Fig. 4.10(b), 2 Mb/s in Fig. 4.10(c) and 1 Mb/s in Fig. 4.10(d). Thus, we can say that the statistical bound is effective for large link capacity and for a stringent delay requirement.

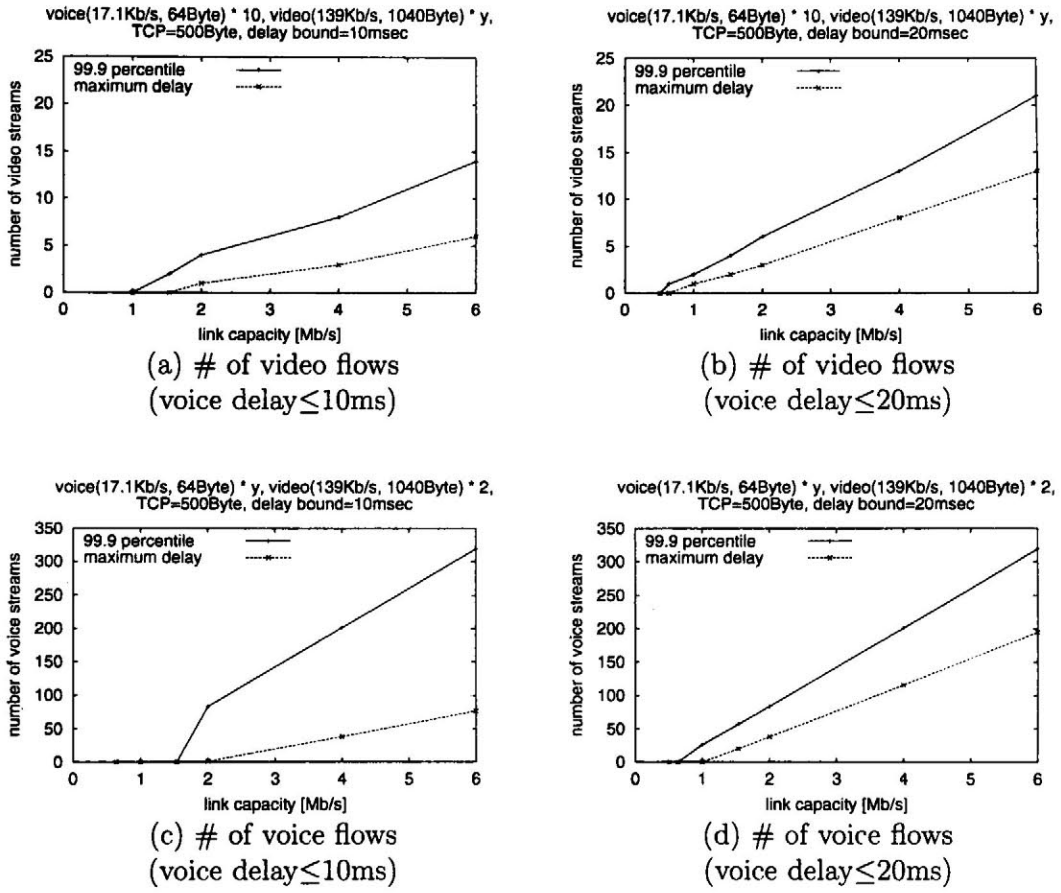
From the above discussions, our statistical delay bound is effective compared with the maximum delay in the enterprise networks of link capacity of from 640 Kb/s to about 6 Mb/s.

For link capacity less than 640 Kb/s, we have to find another solution to satisfy the delay requirement. Our numerical results give a suggestion that, large packet length (e.g., video packets) causes large delay, thus some packet segmentation scheme might be needed. This is a future work.

4.4.5 Effectiveness of header compression and small TCP packet

Finally, we cope with a header compression scheme and small TCP packets. We have already applied these two schemes to homogeneous CBR traffic case in [ITSO01b], and have shown that they are effective in reducing the delay time.

Figure 4.11 gives the 99.9-percentile delay comparison in cases of uncompressed headers and compressed headers. The header can be reduced from a 40-Byte header to a 4-Byte one, as mentioned in [CJ99]. In uncompressed header case, both voice packets and video packets include the ordinary uncompressed header. On the other hand, in compressed header case, both types of CBR packets contain 4-Byte com-



(a) # of video flows (voice delay≤10ms)

(b) # of video flows (voice delay≤20ms)

(c) # of voice flows (voice delay≤10ms)

(d) # of voice flows (voice delay≤20ms)

Figure 4.10: Comaparison between the maximum delay bounds and the statistical bounds.

pressed headers. To compare two cases fairly, the numbers of voice streams of both cases are identical as indicating "⋆ X" in each point of link capacity. From this figure, the compressed header reduces the delay time in all the link-capacity. However, the delay time is almost the same for both two headers in a region of small link capacity. For example, at a rate of 384 Kb/s link capacity, voice delay can be reduced by only 1.60 ms, and video delay by only 0.469 ms. The delay time can be reduced only a little by the compressed headers for small link capacity. This is because the ordinary video packet length is very large (e.g., 1040 Bytes) and the header-compressed packets (e.g., 1004 Bytes) have almost the same length, so that the header compression cannot effectively improve the delay performance.

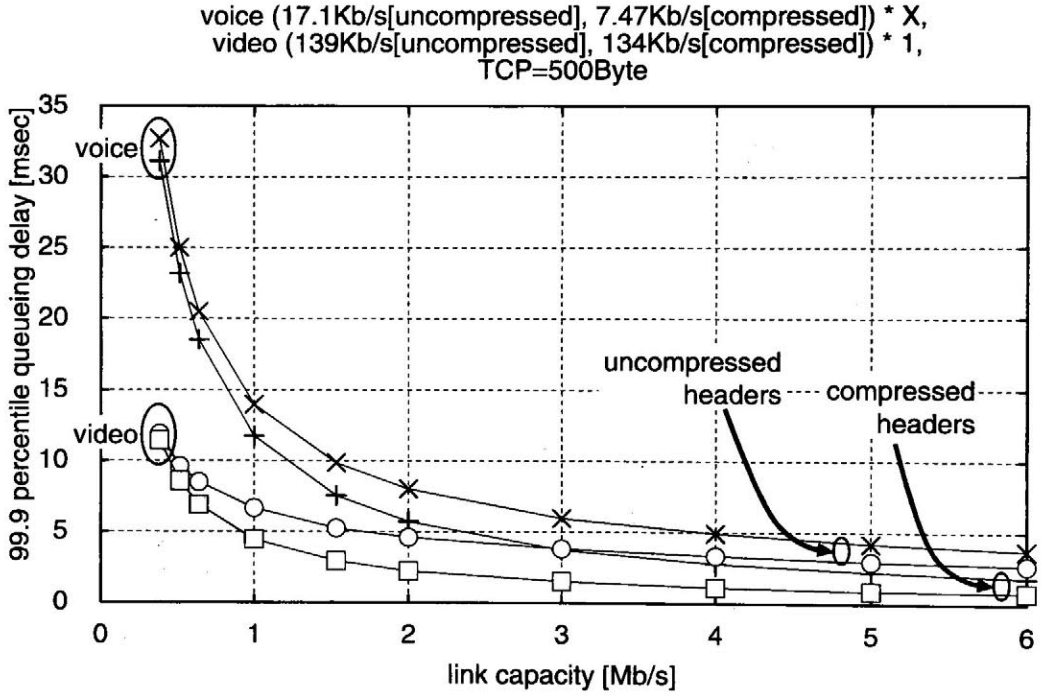


Figure 4.11: Effectiveness of header compression on the statistical delay bound.

Next, we show the impact of TCP packet length on the 99.9-percentile delay in Fig. 4.12. In this figure, there are one video stream and 70 voice streams in a 1536 Kb/s link. Large TCP packets, e.g., 1500 Bytes, cause large delay. Using small

TCP packets, we can reduce the delay performance. For example, 500-Byte TCP packets reduce both voice delay and video delay by about 4.5 ms in comparison with 1500-Byte TCP packets.

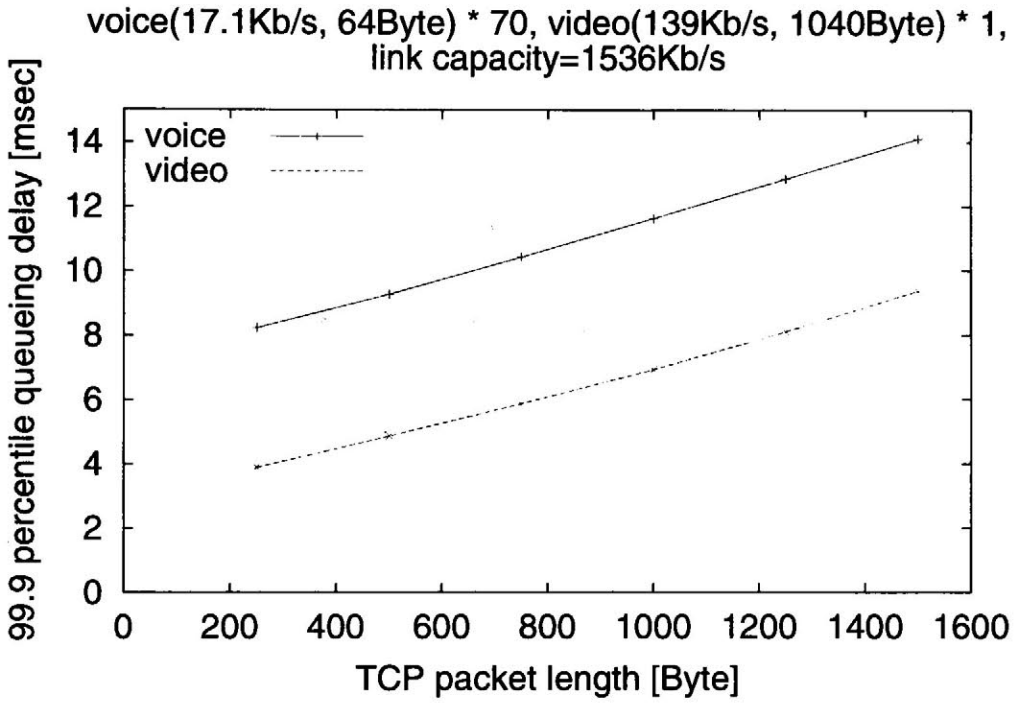


Figure 4.12: Impact of TCP packet length on the statistical delay bound.

4.5 Conclusions and Future Works

Delay time distributions of two types CBR traffic, i.e., voice traffic and video traffic have been examined. They can be heterogeneous in their transmission rates and packet lengths. The delay times are affected by non-real-time traffic. Namely, we have analyzed the $\sum D_i/G/1$ queue with vacations.

We have presented a number of numerical results. 1) voice traffic gives a little impact on the delay time of video traffic, 2) packet length of video traffic has large

influence on the delay time of voice traffic because length of video packets is ten or hundred times as large as voice packets, 3) the statistical delay bound can effectively accept a larger number of streams than the maximum delay in some regions of link capacity, and 4) packet length of non-real-time traffic heavily affects the delay time of both types of real-time packets.

In future works, we should improve the delay performance of CBR packets in extremely small capacity links. This might be achieved by some packet segmentation schemes.

Appendix: General Analysis of Heterogeneous CBR Arrival

In this Appendix, we analyze a more general model than what we need only to obtain $W_n^{(S)}(s)$ and $W_n^{(L)}(s)$. The result is considered as a generalization of that in [Sen90]. The queueing model considered here is characterized as follows.

We consider a single server queue with a superposition of N independent input streams. Arrivals from all arrival streams are characterized by an equilibrium renewal process with an interarrival time of D (deterministic). Arrival streams are classified into K types, class 1 through class K. Service times of class k are independent and identically distributed with a distribution function $B_k(x)$ whose LST is denoted by $B_k^*(s)$. Let b_k $(k = 1, \ldots, K)$ denote the mean service time of class k. Let N_k $(k = 1, \ldots .K)$ denote the number of class k streams. Note that $N = N_1 + \cdots + N_K$. We assume here that there exist real numbers R_k $(k = 1, \ldots .K)$ such that each service time of class k is less than R_k/N_k and $R_1 + \cdots + R_K = D$. The service is a non-preemptive and the service order of arrivals follows the first-come first-serve.

Suppose the queue is empty at time zero. We denote $(n_1, \ldots, n_K)$ by $\boldsymbol{n}$. Let $V_{\boldsymbol{n}}^*(s)$ denote the LST of the virtual waiting time distribution where n_k is equal to the number of class k $(k = 1, \ldots, K)$ arrival streams. Also, let $W_{\boldsymbol{n}}^{(k)}(s)$ $(k = 1, \ldots, K)$ denote the LST of the actual waiting time distribution of class k streams. Using the

equality between the virtual and attained waiting time distributions [Sen89], we have

$$V_{\boldsymbol{n}}^*(s) = 1 - \sum_{k=1}^{K} n_k b_k/D + \sum_{k=1}^{K} (n_k b_k/D) \frac{1 - B_k^*(s)}{s b_k} W_{\boldsymbol{n}}^{(k)}(s),$$

where $0 \leq n_k \leq N_k$ $(k = 1, \ldots, K)$. Further, following the same argument in [Sen90], we can show that the virtual waiting time process is indeed a regenerative process with regeneration cycles of length D. Note that any regeneration cycle has exactly one arrival from each stream and this arrival samples the regeneration cycle uniformly. Thus, an arrival sees the time average work in the system with himself/herself removed [Sen90]. It then follows that

$$W_{\boldsymbol{n}+\boldsymbol{e}_k}^{(k)}(s) = V_{\boldsymbol{n}}^*(s),$$

where $\boldsymbol{e}_k$ denotes the unit vector with the kth element of one. Note here that

$$V_{\boldsymbol{0}}^*(s) = 1.$$

Therefore, $V_{\boldsymbol{n}}(s)$ can be recursively obtained.

Chapter 5

Scalability Analysis of Backbone Network

5.1 Introduction

Recently, the traffic conveyed over the Internet has been growing explosively, thereby requiring backbone networks of huge transmission capacity. The asynchronous transfer mode (ATM) and wavelength division multiplexing (WDM) technologies enable such networks to be built and provide the real-time communication over the Internet with great possibilities.

The Internet community has intensively discussed ways to achieve the real-time communication over the Internet. First, *integrated services* (IntServ) architecture and RSVP signaling scheme have been standardized. The RSVP is however based upon the per-flow management, and this causes the scaling problem [FH98, p. 156]. The queueing schemes have to manage a million flows independently, and thus the admission control schemes should respond to hundreds of millions requests per second, which in turn generates a great number of signaling messages going through the network. This further results in difficulties of implementing of high speed routers.

As a solution to that, IETF has discussed new architecture, called *differentiated services* (DiffServ) architecture [BBC+98]. The DiffServ architecture has introduced the edge concept, in which traffic marking and shaping are done at edges of the

network and intermediate routers constituting high speed backbone networks process each packet based upon some mark written in its header; the related processing is called *per hop behavior* (PHB). This architecture allows flow aggregation, i.e., a number of traffic flows can be handled as one unit; we call this unit a *trunk* in this dissertation. Ingress edge routers can enforce the flow aggregation, and egress edge routers operate the flow segregation.

A flow aggregation technology is also applied to IntServ for improving it in terms of its scalability; RSVP aggregation [GBH97], aggregation of Internet Integrated Services (IIS) state [BV98], and aggregation of RSVP for IPv4 and IPv6 reservations [BILFD99] have been proposed.

A high speed link, e.g., 10 Gb/s, can accommodate one million real-time flows (or streams) of 10 kb/s. The per-flow management is a costly way in this context, while the flow aggregation can reduce the number of units managed independently to acceptable one. Moreover, the delay experienced by the real-time packets there will be negligible due to its high speed packet forwarding capability [ITSO01b], and thus the bandwidth assignment will be of major concern.

Here, the signaling messages can be aggregated; i.e., the admission control schemes decide whether to accept flow setup requests or not on a per-trunk basis. Therefore, this can further reduce the flow management cost associated with the admission control and the signaling exchange.

Nevertheless, since some amount of the bandwidth required by a trunk is allocated to the trunk once the first flow of the trunk is accepted, it will lead to inefficient use of the link. As a result, it will suffer the high call blocking probability in a call admission control (CAC) compared with the per-flow management as long as the available bandwidth is fixed.

In what follows, we study the performance of the flow aggregation by exactly analyzing the call blocking probability of real-time flow. Through our numerical results, we have shown the behavior of flow aggregation, in which flow aggregation requires approximately 1.09 times as much as capacity of the link compared with the per-flow management to achieve the same blocking probability, whereas it reduces the number of flow management units from 100,000 flows to 100 trunks. In addition,

our analysis is so general that it can treat various kinds of heterogeneous cases, and helps to determine the capacity allocation to different kinds of trunks.

The rest of the chapter is organized as follows. We define a network model to analyze the performance in section 2. In sections 3 and 4, we provide analysis and its numerical results, respectively. Lastly, we conclude the chapter in section 5. In the appendix, we give a stable algorithm that allows us to calculate the blocking probability even in a huge system.

5.2 Network model

A network commonly consists of a number of domains that serve as transit subnetworks each other. As shown in figs. 5.1 and 5.2, edge routers and intermediate routers constitute a domain; transit flows generated at other domains first arrive at an edge router denoted by ingress routers, go through several intermediate routers and finally leave there through egress routers.

It will be very likely that each intermediate router has to handle a large number of real-time traffic flows of relatively small bandwidth in the next generation Internet. Therefore, aggregating some flows at ingress routers will enable scalable real-time communications in that context. In this dissertation, we consider a case where some capacity of link coming from an ingress router is assigned to transit real-time communications, and flows over the link go to egress routers. Flows destined for an egress router are managed as a trunk, i.e., by aggregating flows as shown in figs. 5.3 and 5.4.

Link capacity of an ingress router is denoted by C. The number of flows each trunk can accommodate depends upon their traffic characteristics. Namely, our analysis allows heterogeneous trunks, i.e., several types of trunks different in terms of flow arriving rate and its mean holding time. A trunk of type k accommodates a maximum of m_k flows. An ingress router establishes a trunk instead of each flow. Thus, when there is no trunk established between ingress router A and egress router B and a setup requirement of flow between them newly arrives, ingress router A checks whether there is still bandwidth available for a new trunk establishment on the link. The ingress

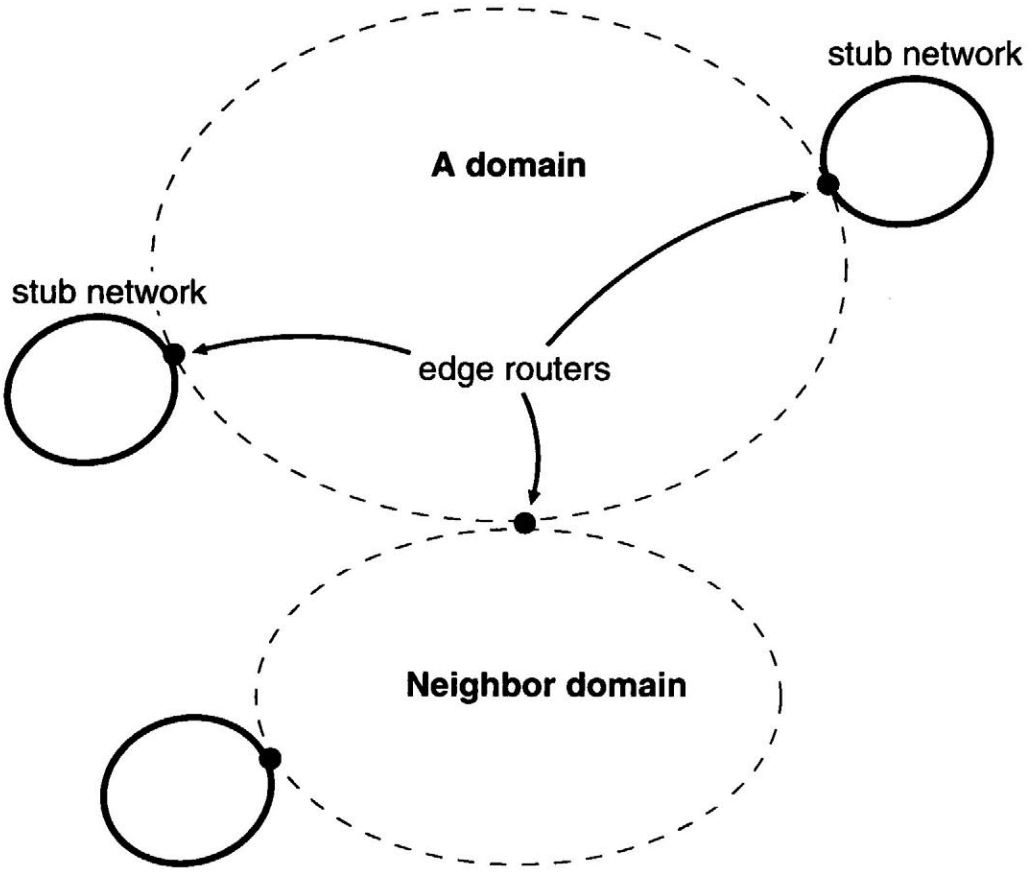


Figure 5.1: Flow aggregation domain

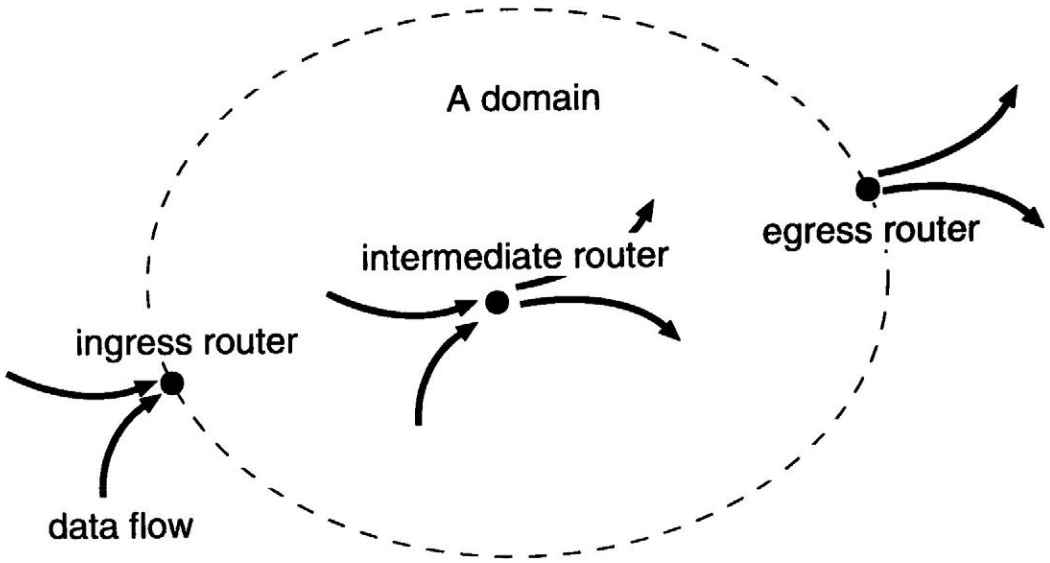


Figure 5.2: 3 types of routers

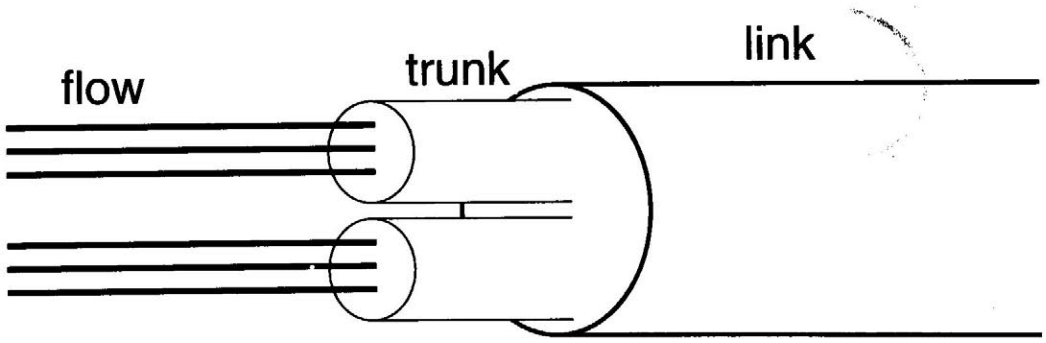


Figure 5.3: Trunk

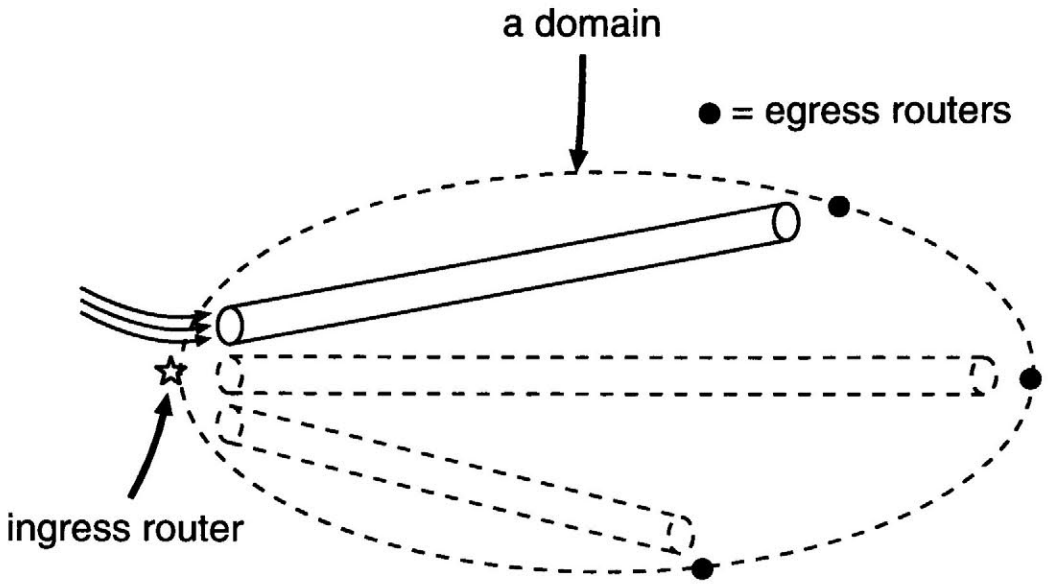


Figure 5.4: System overview

router will reserve some amount of the bandwidth required by m_k flows for a trunk of type k on the link if possible. Otherwise, the trunk will be blocked. Furthermore, if the corresponding trunk is already established and it can further accommodate more flow, the flow will be accepted. Otherwise, the flow will be blocked. The trunk will be released when it has no flow.

In order to evaluate this sort of flow aggregation scheme, we analyze the blocking probability in a blocking model composed of two stages as shown in figs. 5.5 and 5.6. We assume that at most one trunk can be established between ingress router A and each egress router. At the first stage, blocking is caused by contention among flows destined for the same egress router (see fig. 5.5). At the second stage, blocking is then caused by contention among trunks sharing the link (see fig. 5.6).

We will examine how flows can be effectively aggregated at the ingress router for a variety of traffic characteristics and reservation strategies.

The first stage:
blocking is caused by contentions among flows in a trunk

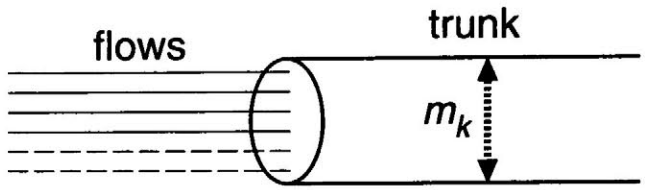


Figure 5.5: Overview of the first-stage

The second stage:
blocking is caused by contentions among trunks

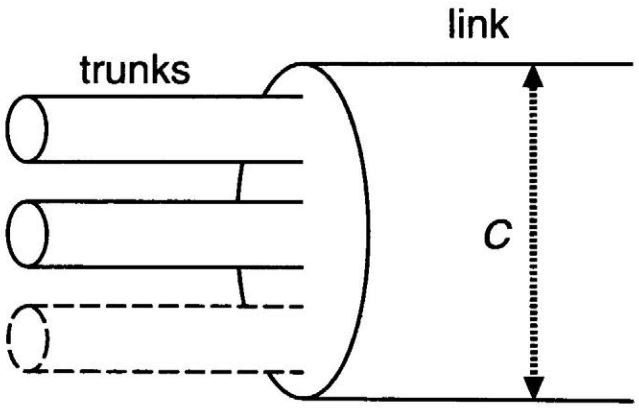


Figure 5.6: Overview of the second-stage

5.3 Analysis

5.3.1 Mathematical model

We first describe the mathematical model of the first stage. We assume that there are P types of queues at the first stage. We denote the number of queues of type k ($k = 1, \ldots .P$) by N_k. Thus, the total number of queues at the first stage is given by $N_1 + \cdots + N_P$. Arrivals of customers at a queue of type k are characterized by an independent Poisson process with rate λ_k, and service times are independent and identically distributed according to a general distribution with finite mean h_k. In a queue of type k ($k = 1, \ldots, P$), there are m_k servers and no waiting room. Thus, arrivals which that all servers busy are lost and never return. Hereafter, we call each queue of type k at the first stage a trunk of type k.

We assume that there are C servers at the second stage. Note that C represents the system capacity as denoted earlier. When an arrival occurs at an empty trunk of type k, it tries to reserve m_k servers at the second stage. If the trunk succeeds in reserving m_k servers simultaneously, it keeps those servers until it becomes empty, and then release them. If the reservation is failed, the arrival is simply lost and never return. Thus, the loss of arrivals can occur in two ways; all servers at the first stage are busy on arrival, and a queue at the first stage is empty but the reservation at the second stage is failed on arrival.

5.3.2 Performance analysis

Let $\rho_k = \lambda_k h_k$ denote the traffic intensity of a trunk of type k at the first stage. For a while, we assume the number C of servers at the second stage is infinite. In this case, each trunk at the first stage behaves as an independent M/G/m/m queue. Let $P_k(i)$ denote the steady state probability of i ($i = 1, \ldots, m_k$) customers at a trunk of type k. Thus, using the result of the M/G/m/m queue [Ros95], we have

$$P_k(i) = P_k(0)\frac{\rho_k^i}{i!}, \qquad i = 0, 1, \ldots, m_k,$$

where

$$P_k(0) = \left[\sum_{i=0}^{m_k} \frac{\rho_k^i}{i!}\right]^{-1}.$$

For the later use, we derive the conditional mean number M_k of customers of a trunk of type k given that it reserves m_k servers at the second stage. By definition, the conditional mean number M_k is given by

$$M_k = \frac{\sum_{i=1}^{m_k} iP_k(i)}{1 - P_k(0)}.$$

Since the numerator on the right-hand side of the above equation represents the unconditional mean number of customers of a trunk of type k, M_k can be rewritten to be

$$M_k = \frac{\rho_k(1 - B(m_k, \rho_k))}{1 - P_k(0)},$$

where $B(m, \rho)$ denotes the blocking probability of an M/G/m/m queue with traffic intensity ρ. It is well known that $B(m, \rho)$ can be computed by the recursion [Coo90]:

$$B(m, \rho) = \frac{\rho B(m-1, \rho)}{m + \rho B(m-1, \rho)}, \qquad m = 1, 2, \ldots$$

with $B(0, \rho) = 1$.

Next, we derive the mean length H_k of busy periods of a trunk of type k. Since the sequence of idle and busy periods is regarded as an alternating renewal process, we have

$$1/\lambda_k : H_k = P_k(0) : 1 - P_k(0).$$

We thus obtain

$$H_k = \frac{1 - P_k(0)}{\lambda_k P_k(0)}.$$

We now consider the second-stage with a finite number C of servers. Note that the second stage is regarded as a generalized Engset model [Hui90], where service times are characterized by busy periods of trunks. We define $Q(\boldsymbol{n})$ as the steady state probability of n_k $(k = 1, \ldots, P)$ trunks of type k at the second stage, where

$\boldsymbol{n} = (n_1, \ldots, n_P)$. It then follows from Insensitivity Theorem on p.222 in [Hui90] that

$$Q(\boldsymbol{n}) = \begin{cases} \dfrac{\prod_{k=1}^{P} \binom{N_k}{n_k} \sigma_k^{n_k}}{\sum_{\boldsymbol{n} \in \mathcal{T}(\boldsymbol{N}, C)} \prod_{k=1}^{P} \binom{N_k}{n_k} \sigma_k^{n_k}}, & \boldsymbol{n} \in \mathcal{T}(\boldsymbol{N}, C), \\ 0, & \text{otherwise}, \end{cases}$$

where $\boldsymbol{N} = (N_1, \ldots, N_P)$,

$$\sigma_k = \lambda_k H_k, \qquad k = 1, \ldots, P,$$

and

$$\mathcal{T}(\boldsymbol{N}, C) = \left\{ \boldsymbol{n} \;\middle|\; \sum_{k=1}^{P} n_k m_k \leq C,\ 0 \leq n_k \leq N_k \ \ (k = 1, \ldots, P) \right\}.$$

To obtain the blocking probability $Q_{loss}(k)$ of trunks of type k, we define $G(\boldsymbol{N}, C)$ and $U_k(\boldsymbol{N}, C)$ as

$$\begin{aligned} G(\boldsymbol{N}, C) &= \sum_{\boldsymbol{n} \in \mathcal{T}(\boldsymbol{N}, C)} \prod_{k=1}^{P} \binom{N_k}{n_k} \sigma_k^{n_k}, \\ U_k(\boldsymbol{N}, C) &= \sum_{\boldsymbol{n} \in \mathcal{T}(\boldsymbol{N}, C)} n_k Q(\boldsymbol{n}), \end{aligned} \tag{5.1}$$

respectively. Note that $U_k(\boldsymbol{N}, C)$ denotes the mean number of trunks of type k at the second stage. It then follows from (3.15) in [Ros95] that

$$Q_{loss}(k) = 1 - \frac{N_k \dfrac{G(\boldsymbol{N}, C - m_k)}{G(\boldsymbol{N}, C)} - U_k(\boldsymbol{N}, C - m_k)}{N_k - U_k(\boldsymbol{N}, C)}.$$

As stated, customer loss occurs in two ways. One is that a customer arrives when his/her trunk is full, and the other is that a customer arrives when the trunk is empty but he or she fails the reservation at the second stage. We denote the blocking probability of customers arriving at trunks of type k in the former and the latter cases by $P_{loss}^{(1)}(k)$ and $P_{loss}^{(2)}(k)$, respectively. Apparently, the overall blocking probability $P_{loss}(k)$ of customers arriving at a trunk of type k is given by

$$P_{loss}(k) = P_{loss}^{(1)}(k) + P_{loss}^{(2)}(k).$$

We first consider $P_{loss}^{(1)}(k)$. We note that the steady state probability $r(k)$ that a particular trunk of type k is reserving m_k servers at the second stage is given by

$$r(k) = \frac{U_k(\boldsymbol{N}, C)}{N_k}.$$

Therefore, we have

$$P_{loss}^{(1)}(k) = r(k)\frac{B(m_k, \rho_k)}{1 - P_k(0)}.$$

Next, we consider the overall blocking probability $P_{loss}(k)$. Since the blocking probability is a fraction of lost customers to customers arriving in a unit time, we have

$$P_{loss}(k) = 1 - \frac{M_k U_k(\boldsymbol{N}, C)}{N_k \rho_k}.$$

Thus, we have $P_{loss}^{(2)}(k) = P_{loss}(k) - P_{loss}^{(1)}(k)$.

The loss probabilities $P_{loss}^{(1)}(k)$ and $P_{loss}^{(2)}(k)$ are given in terms of $U_k(\boldsymbol{N}, C)$ that should be numerically evaluated. We note that a recursion to compute the mean number $U_k(\boldsymbol{N}, C)$ of trunks of type k is known [Ros95]. However, the recursion in [Ros95] produces the normalizing constant G that will be very huge for systems of large size and runs into numerical difficulty. In the appendix, we provide an alternative algorithm to avoid this numerical difficulty, which is similar to the mean value analysis for Erlang loss models with Poisson arrivals [Chl94].

5.4 Numerical results

In this section, we evaluate the efficiency of flow aggregation using our analysis. First, we examine the homogeneous trunk case, and then the heterogeneous trunk one. Finally, we show a design example of flow aggregating networks.

5.4.1 Homogeneous trunk case

To see the fundamental characteristics of our system, we treat a homogeneous case and suppose the following situation. Each trunk can accommodate a maximum of 100 flows ($m = 100$), and each flow arriving at the ingress router is equally destined

for each egress router; there are 100 egress routers in total, i.e., $N = 100$. Among them, up to 80 trunks are accepted simultaneously due to the limited link capacity, which means that the link capacity C is set to 8,000 flows.

In fig. 5.7, we illustrate the impact of the traffic intensity on the mean number of arriving, accepted and allocated flows. The x-axis indicates traffic intensity ρ for each of trunks. From this figure, we can observe that about 7,886 flows are already allocated for trunks when $\rho = 2$, whereas only 182 flows are accepted on an average over the link. Hence, we can say that most of the link capacity is wasted there. Furthermore, the blocking probability reaches approximately 0.1, which is not acceptable. The reason why this sort of event happens can be explained as follows. Arriving flows are going to each egress router evenly. Thus, as the traffic intensity increases, most pairs of the ingress router and an egress one will need their own trunks, but at most only 80 trunks among 100 egress routers can be accepted. Consequently, flow aggregation causes both high blocking probability and low link utilization.

We next treat the effect of link capacity upon the blocking probability in fig. 5.8. In this figure, the traffic intensity ρ, which is the mean number of arriving flows, is set to 100, and there are 200 egress routers ($N = 200$) in the domain. To keep the blocking probability of the first stage less than 10^{-4}, we set m to 137, i.e., each trunk can accommodate at most 137 flows. When the link is of capacity, C, equal to or greater than 27,400 flows, it can accommodate all of the 200 trunks each including 137 flows. In this case, the blocking never occurs at the second stage of our analytical model so that the blocking probability is kept less than 10^{-4}. Otherwise, i.e., if C is less than 27,400, the blocking can occur at the second stage as well as the first stage. For example, when $C = 27,300$, at most 199 trunks can be accepted among 200 ones. In this case, the blocking probability at the second stage is dominant so that the total blocking probability is approximately equal to 1/200 as shown in fig. 5.8. Similarly, when $C = 27,200$, the blocking probability reaches 2/200 because the link can accept at most 198 trunks of 200 ones.

Here, we shall compare the flow aggregation aware system and the unaware system. In the flow aggregation aware system, the edge routers set up the trunks and each flow is accommodated in one of the trunks. On the other hand, in the unaware

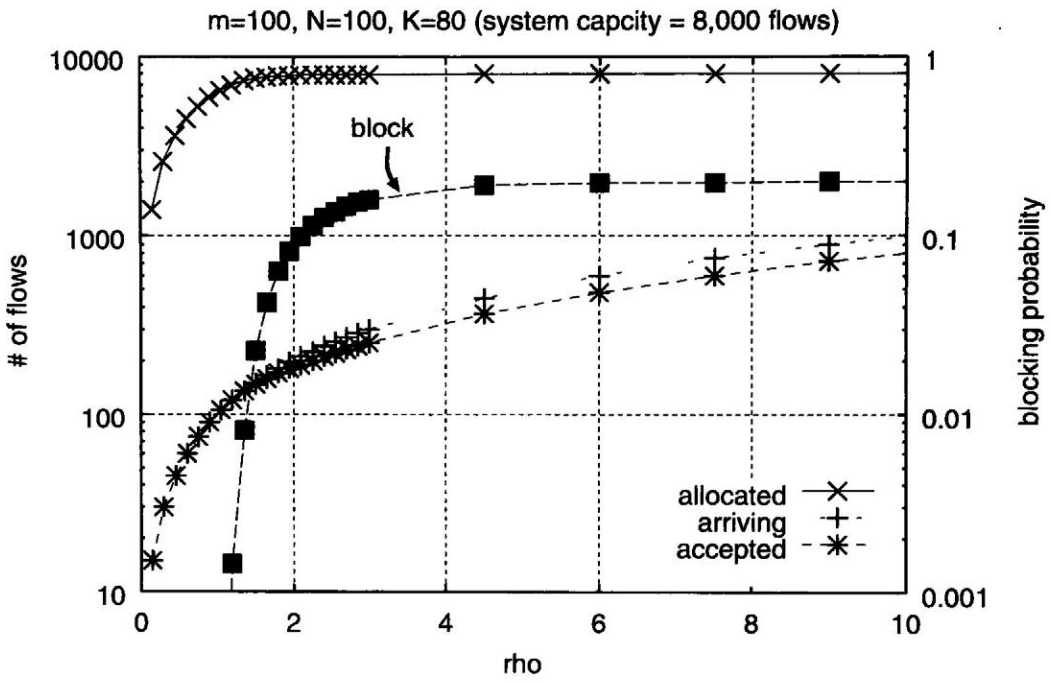


Figure 5.7: The impact of the traffic intensity ρ

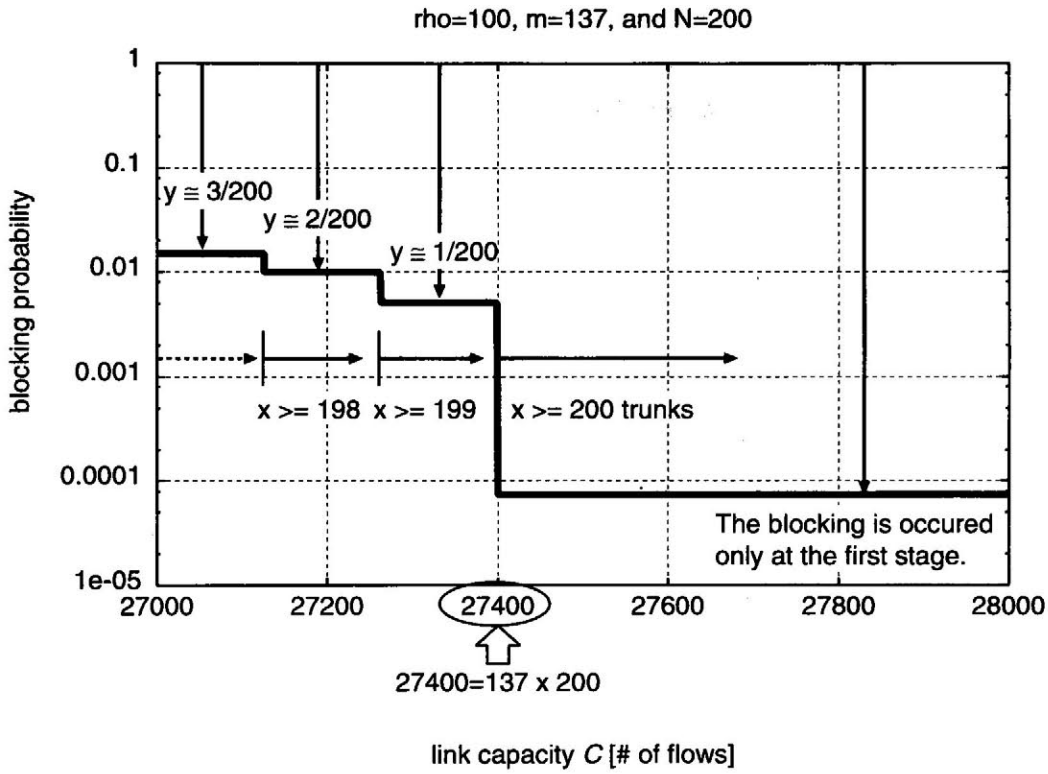


Figure 5.8: The impact of the link capacity C

system, the edge routers do not set up the trunks so that the intermediate routers have to manage every flow, i.e., the per-flow management. In fig. 5.9, we investigate the blocking probability of both systems. The parameters for the aware system are the same as in fig. 5.8. In the unaware system, the blocking probability can be easily obtained by Erlang's B formula as stated in the previous section. As shown in this figure, the flow aggregation aware system suffers higher blocking probability than the unaware system. When the mean number of arriving flows is 20,000, the aware system needs a link of capacity of 27,400 flows to attain the blocking probability less than 10^{-4}, while a link of 20,400 flows is sufficient in the unaware system.

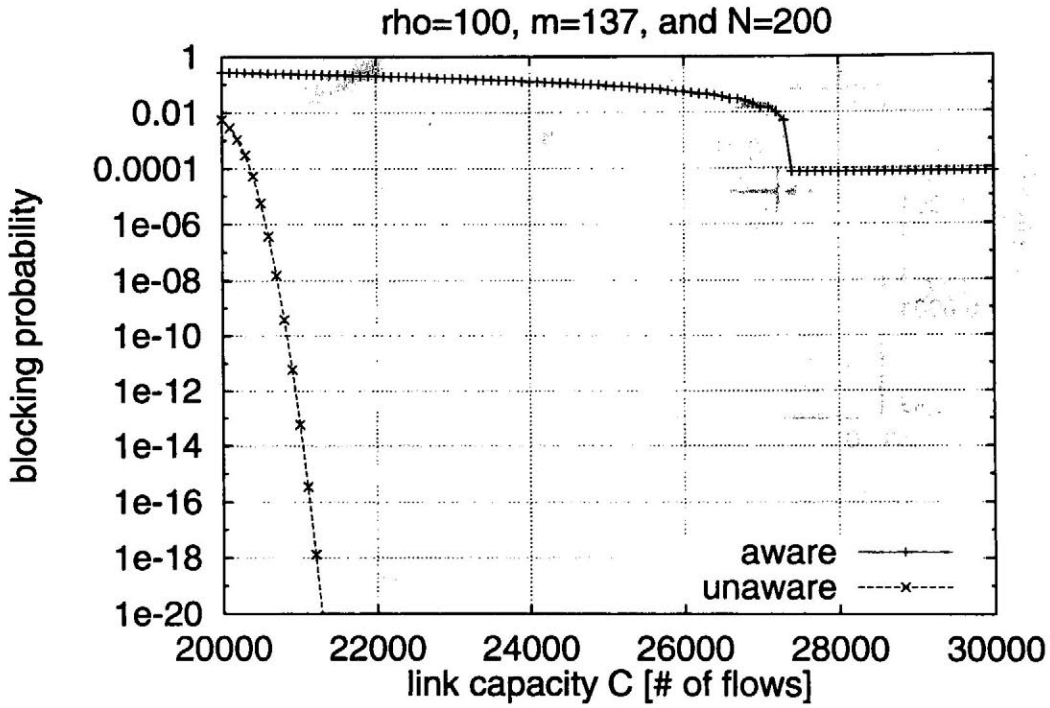


Figure 5.9: The comparison of aware system and unaware system

In order to improve the blocking probability, we have to provide more capacity for real-time traffic class than that required on an average. More specifically, every pair of the ingress router and an egress router should have a trunk because the blocking

probability at the second stage is dominant as large as we stated before. At the first stage, capacity of each trunk can be increased from the mean arriving traffic to decrease the blocking probability to the acceptable one.

Table 5.1: Flow aggregation aware system versus unaware system: effectiveness of provisioned capacity

Mean amount of traffic for all trunks	Mean # of flows can be accepted	Ratio of the provisioned capacity of aware system to that of unaware system
10Mb/s	1,000	$\frac{24.0\text{Mb/s}}{11.0\text{Mb/s}} = 2.18$
100Mb/s	10,000	$\frac{137\text{Mb/s}}{103\text{Mb/s}} = 1.33$
1,000Mb/s	100,000	$\frac{1.10\text{Gb/s}}{1.01\text{Gb/s}} = 1.09$

In table. 5.1, we give the provisioned capacity required to keep the blocking probability less than 10^{-4}. We focus on the Internet telephony and regard flow as CBR traffic at a rate of 10 kb/s. The number N of egress routers is fixed to 100. For example in this table, if the traffic intensity $\rho = 1000$, there are $100,000$ flows, which should be managed separately in the unaware system, while the aware system manages at most 100 trunks at the intermediate routers. As shown in the table, the bandwidth of 11.0 Mb/s, which is called the provisioned capacity here, is needed to carry traffic of 10 Mb/s to keep the blocking probability less than 10^{-4} in the unaware system, although the unaware system assigns each flow as much capacity as required. On the other hand, the bandwidth of 24 Mb/s is needed in the aware system. The ratio of the provisioned capacity for the flow aggregation aware system to that for the unaware system is given in the table. The ratio is indeed small for trunks of strong intensity ρ. In other word, trunks of stronger intensity, or trunks of larger capacity make the ratio smaller. From the discussion, we can say that flow aggregation can be implemented while keeping the blocking probability within the acceptable level and without wasting the bandwidth so much in the future high speed networks.

5.4.2 Heterogeneous trunk case

The heterogeneous trunk case more often happens in actual networks. We have to decide how much capacity should be allocated to each trunk according to the amount of traffic carried over it. In addition, some trunks share a link of limited capacity, and thus trunks of different capacity can further affect each other in the heterogeneous case. Therefore, we examine a capacity allocation issue arising in a heterogeneous case in which a link is shared by two types of trunks, which are different in their mean interarrival times of flow setup requests as follows.

Given parameters

- Number of traffic types: $P = 2$. (Low and High)
- Bandwidth required by each flow = 10kb/s.
- Link capacity: C = 15,000 (150Mb/s)
- Number of low trunks: $L = 100$. (denoted by N_1 in the previous section)
- Traffic intensity for each low trunk: $\rho_L = 2$ flows.
  - Mean holding time: $h_L = 1$ [min].
  - Mean interarrival time: $(1/\lambda_L) = 30$ [sec].
- Number of high trunks: $H = 10$. (denoted by N_2 in the previous section)
- Traffic intensity for each high trunk: $\rho_H = 1250$ flows.
  - Mean holding time: $h_H = 1$ [min].
  - Mean interarrival time: $(1/\lambda_H) = 48$ [ms].

Design parameters

- Capacity of each of low-class trunks: m_L flows. (denoted by m_1 in the previous section)
- Capacity of each of high-class trunks: m_H flows. (denoted by m_2 in the previous section)

In this subsection, we suppose two types of traffic; say low traffic and high traffic. The low (resp. high) traffic flows are accommodated by one of low (resp. high) trunks. The high trunks have larger capacity to meet their higher demand than the low trunks. Figs. 5.10 and 5.11 show the impact of m_L and m_H on the blocking probability in which m_L and m_H are the maximum number of flows accommodated by each high class and low class trunk, respectively. In these figures, the total amount of traffic is fixed: 100×2 (for low trunks) $+ 10 \times 1,250$ (for high trunks) $= 12,700$. Thus, the mean utilization becomes $12,700/15,000 = 84.7\%$ if there is no blocking. From fig. 5.10, $(m_L, m_H) = (10, 1400)$ is the optimal value to minimize the blocking probability for low trunks, in which both high and low trunk blocking probability becomes smaller than 10^{-4}.

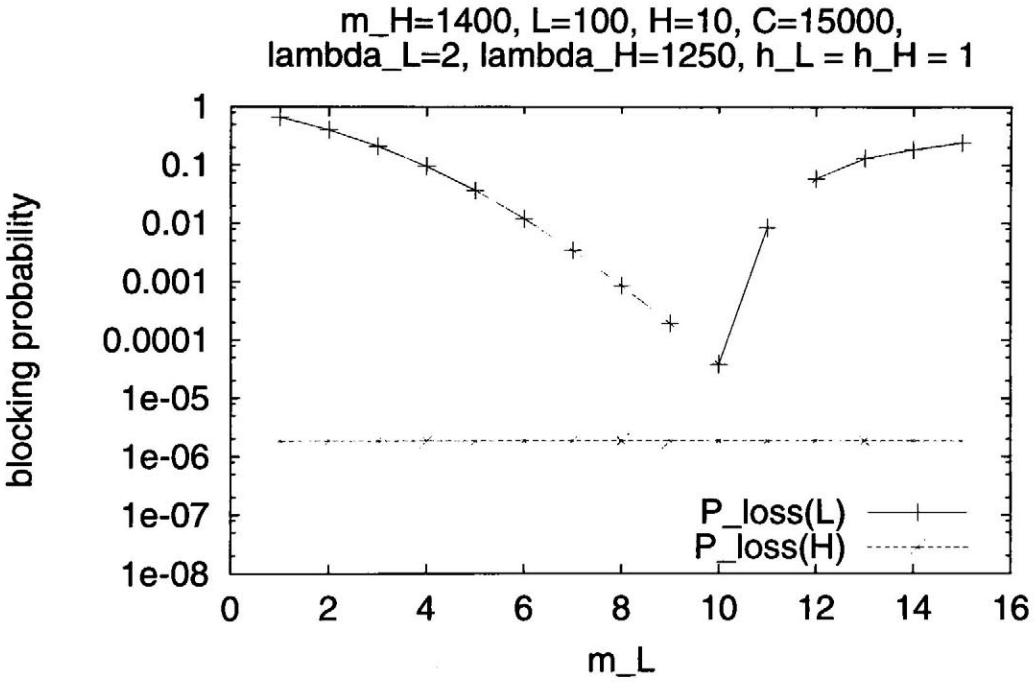


Figure 5.10: The impact of m_L on blocking probability

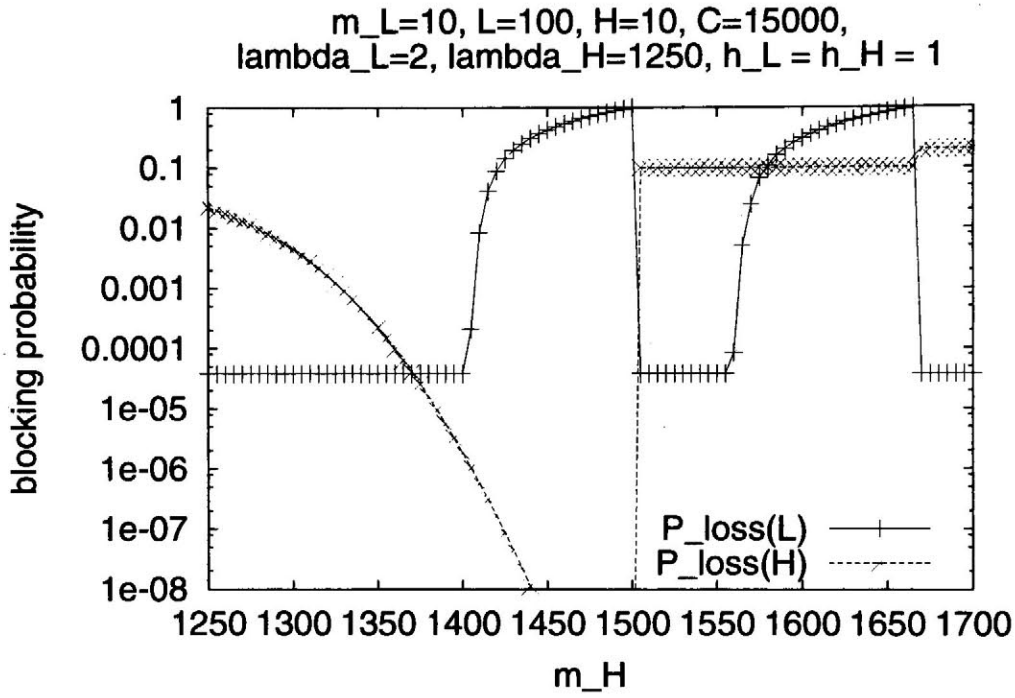


Figure 5.11: The impact of m_H on blocking probability

Fig. 5.11, shows the total probability of blocking occurring at both the first and second stages, illustrated in figs. 5.5 and 5.6, as a function of m_H. The performance characteristic looks so complicated that we have further to investigate it in detail. Thus, in fig. 5.12, we discriminate between the blocking probability of high trunks at the first stage and that at the second stage, instead of showing the total one. From this figure, the blocking probability at the first stage decreases monotonously with the trunk capacity m_H. This is a straightforward feature.

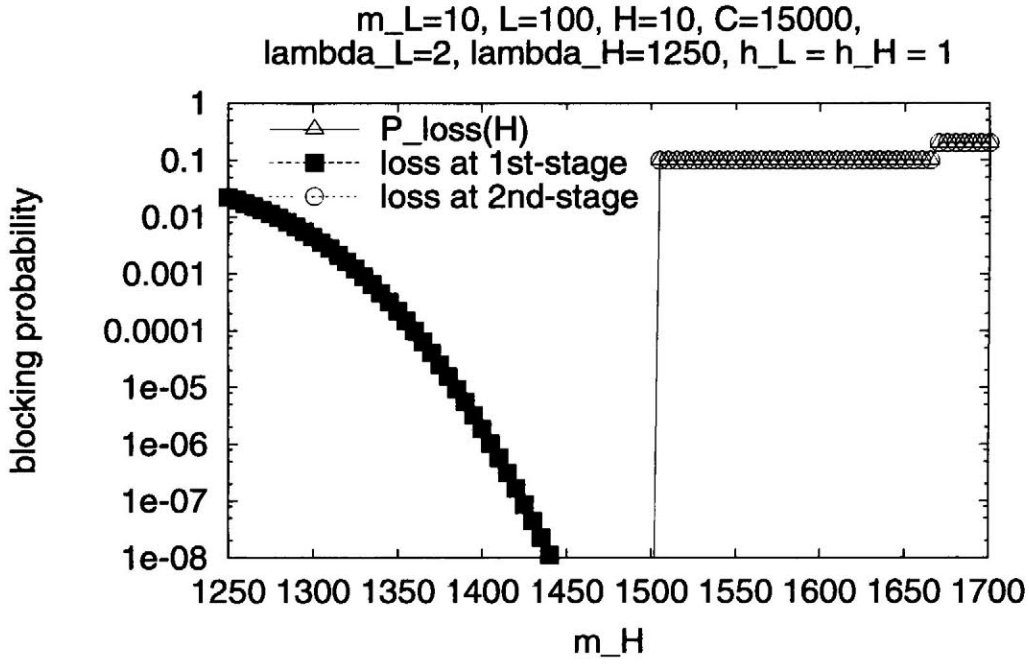


Figure 5.12: The impact of m_H on $P_{loss}(H)$ at the first stage and the second stage

Further to examine the blocking probability characteristics at the second stage, we then give the impact of m_H on the mean number of accepted trunks in fig. 5.13. We denote the mean number of accepted trunks at the second stage by U_L and U_H for low-class and for high-class, each of which can be computed by $U_L(\boldsymbol{N}, C)$ and $U_H(\boldsymbol{N}, C)$ as in eq. (5.1). First, we will focus on the behavior of high trunks. In

this figure, U_H becomes 10 when $m_H \leq 1,500$. Namely, noting that $H = 10$, we see that all high-class trunks can be accepted; i.e., the blocking probability at the second stage is 0. When m_H exceeds 1,500, at most nine trunks can be accepted among 10 trunks and the blocking probability increases abruptly as shown in fig. 5.12. Next, we pay attention to the behavior of low trunks. Since we set the link capacity C to 15,000, as m_H increases, the total capacity for high trunks approaches C and U_L decreases. Thus, if m_H becomes 1,500, U_L is 0, i.e., all flows on low trunks are rejected. When m_H exceeds 1,500, at most nine trunks can be accepted for high class as stated earlier and the rest of capacity can be allocated for low trunks thereby improving the blocking probability of low trunks. As m_H further increases, the bandwidth available for the low-class decreases and then its blocking probability also decreases. This is the reason why the blocking probability at the second stage has a complicated feature as shown in fig. 5.11.

5.4.3 Capacity allocation scheme in flow aggregation

In the previous subsection, we have investigated characteristics of the blocking probability in the heterogeneous case. Using our analysis, we can cope with a design issue arising in allocating capacity to each trunk. The issue is to determine how much capacity should be allocated to trunks of different classes to meet some specified blocking probability, say 10^{-4}.

In the case treated in subsection 5.4.2, it has been shown from figs. 5.10 and 5.11 that a pair of $m_L = 10$ and $m_H = 1400$ satisfies the above blocking probability. As a result, the total capacity required by a link is 15,000 flows, whereas the mean utilization is 12,700 as stated earlier in 5.4.2.

Therefore, our analysis helps us to design a way of flow aggregation and know how much capacity is actually needed to meet some blocking probability requirement.

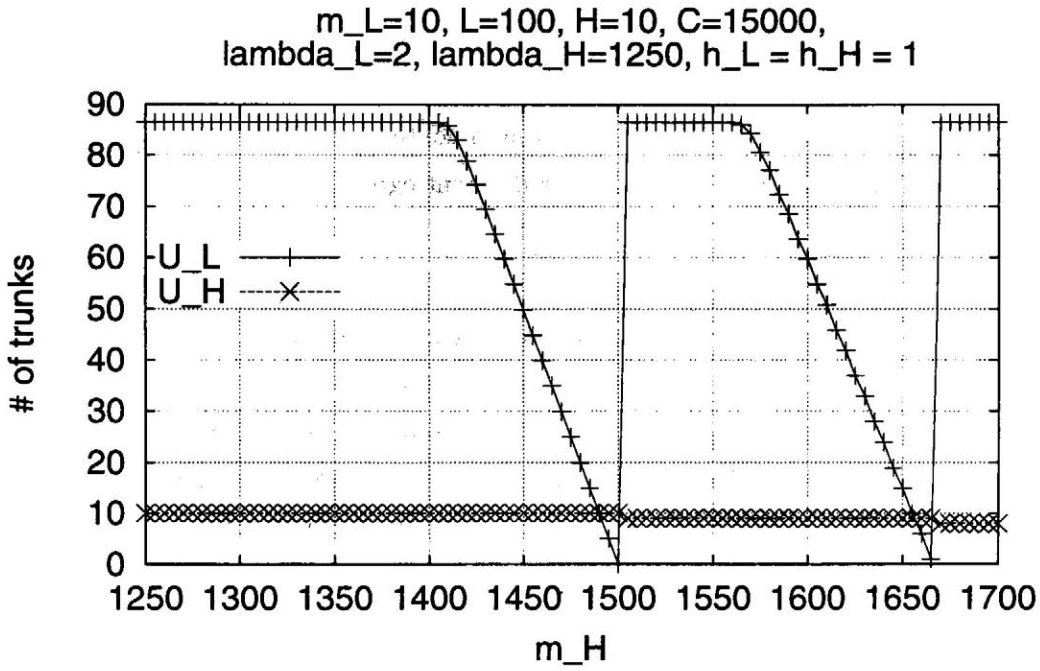


Figure 5.13: The impact of m_H on the mean number of accepted trunks on the second stage: U_L and U_H

5.5 Conclusions

The flow aggregation scheme can greatly reduce the flow management cost in accommodating low speed voice calls in gigabit networks, whereas it can result in inefficient use of the network resources due to redundant capacity allocation. Accordingly, we have exactly analyzed the performance of flow aggregation of constant bit rate type traffic, in terms of the call blocking probability at the ingress routers.

First, we have described the network model developed here to analyze the blocking performance. That can be regarded as the two-stages blocking model. At the first stage, flows destined for the same egress router contend for a trunk of limited capacity. At the second stage, trunks destined for different egress routers then contend for the link capacity.

We have then analyzed the model in which the first stage has been modeled by M/G/m/m type queueing system and the second stage can be analyzed by the generalized Engset model. Moreover, in the appendix, we have proposed a stable numerical algorithm which allows us to easily obtain numerical results even in a link of huge capacity accommodating as many as 100,000 flows; it does not require a normalized constant of G that will be very huge for systems of large size and thus will lead to difficulties in obtaining numerical results.

Finally, we have presented some numerical results using the analysis. Through our results, we have seen that flow aggregation is effective in a large system, while it requires larger capacity than the flow aggregation unaware system to achieve some specified blocking probability.

Appendix: Mean-Value Analysis of 2-Stage Blocking Model

In this appendix, we show a numerically stable recursion to compute the mean number $U_k(\boldsymbol{N}, C)$ of trunks of type k by means of mean value analysis [Chl94]. We

first rewrite (5.1) to be

$$\begin{aligned}
U_k(\boldsymbol{N},C) &= \frac{1}{G(\boldsymbol{N},C)} \sum_{\boldsymbol{n}\in\mathcal{T}(\boldsymbol{N},C)} n_k \prod_{l=1}^{P} \binom{N_l}{n_l} \sigma_l^{n_l} \\
&= \frac{1}{G(\boldsymbol{N},C)} \sum_{n=1}^{N_k} \sum_{\substack{\boldsymbol{n}\in\mathcal{T}(\boldsymbol{N},C) \\ n_k=n}} n \binom{N_k}{n} \sigma_k^n \prod_{\substack{l=1 \\ l\neq k}}^{P} \binom{N_l}{n_l} \sigma_l^{n_l} \\
&= \frac{1}{G(\boldsymbol{N},C)} \sum_{n=0}^{N_k-1} \sum_{\substack{\boldsymbol{n}\in\mathcal{T}(\boldsymbol{N},C) \\ n_k=n+1}} N_k \sigma_k \binom{N_k-1}{n} \sigma_k^n \prod_{\substack{l=1 \\ l\neq k}}^{P} \binom{N_l}{n_l} \sigma_l^{n_l} \\
&= \frac{N_k\sigma_k}{G(\boldsymbol{N},C)} \sum_{\boldsymbol{n}\in\mathcal{T}(\boldsymbol{N}-\boldsymbol{e}_k,C-m_k)} \prod_{l=1}^{P} \binom{N_l}{n_l} \sigma_l^{n_l},
\end{aligned}$$

where $\boldsymbol{e}_k$ $(k = 1, \ldots, P)$ denotes the unit vector having one at the kth position. Thus, we obtain

$$U_k(\boldsymbol{N},C) = N_k\sigma_k \frac{G(\boldsymbol{N}-\boldsymbol{e}_k, C-m_k)}{G(\boldsymbol{N},C)}. \tag{5.2}$$

We define $E_k(\boldsymbol{N},C)$ $(k = 1, \ldots, P)$ as the mean number of idle trunks of type k. We then have

$$\begin{aligned}
E_k(\boldsymbol{N},C) &= \frac{1}{G(\boldsymbol{N},C)} \sum_{\boldsymbol{n}\in\mathcal{T}(\boldsymbol{N},C)} (N_k - n_k) \prod_{l=1}^{P} \binom{N_l}{n_l} \sigma_l^{n_l} \\
&= \frac{1}{G(\boldsymbol{N},C)} \sum_{n=0}^{N_k-1} \sum_{\substack{\boldsymbol{n}\in\mathcal{T}(\boldsymbol{N},C) \\ n_k=n}} (N_k - n) \binom{N_k}{n} \sigma_k^n \prod_{\substack{l=1 \\ l\neq k}}^{P} \binom{N_l}{n_l} \sigma_l^{n_l} \\
&= \frac{1}{G(\boldsymbol{N},C)} \sum_{n=0}^{N_k-1} \sum_{\substack{\boldsymbol{n}\in\mathcal{T}(\boldsymbol{N},C) \\ n_k=n}} N_k \binom{N_k-1}{n} \sigma_k^n \prod_{\substack{l=1 \\ l\neq k}}^{P} \binom{N_l}{n_l} \sigma_l^{n_l}.
\end{aligned}$$

Thus, we have

$$E_k(\boldsymbol{N},C) = N_k \frac{G(\boldsymbol{N}-\boldsymbol{e}_k, C)}{G(\boldsymbol{N},C)}. \tag{5.3}$$

We define $E(\boldsymbol{N},C)$ as the mean number of idle servers at the second stage. To obtain an equation which $E(\boldsymbol{N},C)$ satisfies, we define $\mathcal{S}(c)$ as

$$\mathcal{S}(c) = \left\{ \boldsymbol{n} \;\middle|\; \sum_{k=1}^{P} m_k n_k = c, \quad 0 \le n_k \le N_k \quad (k = 1, \ldots, P) \right\}, \tag{5.4}$$

and let $q(c)$ be

$$q(c) = \sum_{\boldsymbol{n}\in\mathcal{S}(c)} Q(\boldsymbol{n}). \tag{5.5}$$

We then have

$$
\begin{aligned}
E(\boldsymbol{N},C) &= \sum_{m=0}^{C-1}(C-m)q(m) \\
&= C\frac{G(\boldsymbol{N},C-1)}{G(\boldsymbol{N},C)} - \sum_{m=1}^{C-1} mq(m) \\
&= C\frac{G(\boldsymbol{N},C-1)}{G(\boldsymbol{N},C)} - \frac{G(\boldsymbol{N},C-1)}{G(\boldsymbol{N},C)} \sum_{m=1}^{C-1} \sum_{\boldsymbol{n}\in\mathcal{S}(m)} \frac{m\prod_{l=1}^{P}\binom{N_l}{n_l}\sigma_l^{n_l}}{G(\boldsymbol{N},C-1)} \\
&= C\frac{G(\boldsymbol{N},C-1)}{G(\boldsymbol{N},C)} - \frac{G(\boldsymbol{N},C-1)}{G(\boldsymbol{N},C)} \sum_{m=1}^{C-1} \sum_{\boldsymbol{n}\in\mathcal{S}(m)} \frac{\left(\sum_{l=1}^{P} m_l n_l\right)\prod_{l=1}^{P}\binom{N_l}{n_l}\sigma_l^{n_l}}{G(\boldsymbol{N},C-1)} \\
&= \frac{G(\boldsymbol{N},C-1)}{G(\boldsymbol{N},C)}\left(C - \sum_{k=1}^{P} m_k U_k(\boldsymbol{N},C-1)\right).
\end{aligned}
$$

It then follows from this equation and

$$
E(\boldsymbol{N},C) + \sum_{k=1}^{P} m_k U_k(\boldsymbol{N},C) = C \tag{5.6}
$$

that

$$
E(\boldsymbol{N},C) = \frac{G(\boldsymbol{N},C-1)}{G(\boldsymbol{N},C)}(E(\boldsymbol{N},C-1)+1). \tag{5.7}
$$

Note that (5.2), (5.3) and (5.7) constitute a recursion to compute $U_k(\boldsymbol{N},C)$. Let $\alpha_k(\boldsymbol{N},C)$ $(k=1,\ldots,P)$ and $\beta(\boldsymbol{N},C)$ be

$$
\alpha_k(\boldsymbol{N},C) = \frac{G(\boldsymbol{N}-\boldsymbol{e}_k,C)}{G(\boldsymbol{N},C)}, \tag{5.8}
$$

$$
\beta(\boldsymbol{N},C) = \frac{G(\boldsymbol{N},C-1)}{G(\boldsymbol{N},C)}, \tag{5.9}
$$

respectively. With those, we rewrite (5.2) in two different ways:

$$
\begin{aligned}
U_k(\boldsymbol{N},C) &= N_k\sigma_k\frac{G(\boldsymbol{N}-\boldsymbol{e}_k,C)}{G(\boldsymbol{N},C)}\cdot\frac{G(\boldsymbol{N}-\boldsymbol{e}_k,C-m_k)}{G(\boldsymbol{N}-\boldsymbol{e}_k,C)} \\
&= N_k\sigma_k\alpha_k(\boldsymbol{N},C)\prod_{m=0}^{m_k-1}\beta(\boldsymbol{N}-\boldsymbol{e}_k,C-m),
\end{aligned} \tag{5.10}
$$

and

$$
U_k(\boldsymbol{N},C) = N_k\sigma_k\frac{G(\boldsymbol{N}-\boldsymbol{e}_k,C-m_k)}{G(\boldsymbol{N},C-m_k)}\frac{G(\boldsymbol{N},C-m_k)}{G(\boldsymbol{N},C)}.
$$

$$= N_k \sigma_k \alpha_k(\boldsymbol{N}, C - m_k) \prod_{m=0}^{m_k-1} \beta(\boldsymbol{N}, C - m). \tag{5.11}$$

With (5.3), (5.8) and (5.10), we have

$$\begin{aligned} N_k &= U_k(\boldsymbol{N}, C) + E_k(\boldsymbol{N}, C) \\ &= N_k \sigma_k \alpha_k(\boldsymbol{N}, C) \prod_{m=0}^{m_k-1} \beta(\boldsymbol{N} - \boldsymbol{e}_k, C - m) + N_k \alpha_k(\boldsymbol{N}, C), \end{aligned}$$

from which, it follows that

$$\alpha_k(\boldsymbol{N}, C) = \frac{1}{1 + \sigma_k \prod_{m=0}^{m_k-1} \beta(\boldsymbol{N} - \boldsymbol{e}_k, C - m)}. \tag{5.12}$$

On the other hand, it follows from (5.6), (5.7) and (5.11) that

$$\begin{aligned} C &= E(\boldsymbol{N}, C) + \sum_{k=1}^{P} m_k U_k(\boldsymbol{N}, C) \\ &= \beta(\boldsymbol{N}, C)(E(\boldsymbol{N}, C-1) + 1) + \sum_{k=1}^{P} m_k N_k \sigma_k \alpha_k(\boldsymbol{N}, C - m_k) \prod_{m=0}^{m_k-1} \beta(\boldsymbol{N}, C - m). \end{aligned}$$

$$\begin{aligned} \beta(\boldsymbol{N}, C) &= \frac{C}{E(\boldsymbol{N}, C-1) + 1 + \sum_{k=1}^{P} m_k N_k \sigma_k \alpha_k(\boldsymbol{N}, C - m_k) \prod_{m=1}^{m_k-1} \beta(\boldsymbol{N}, C - m)} \\ &= \frac{C}{\left(\begin{array}{l} C - \sum_{k=1}^{P} m_k N_k \sigma_k \alpha_k(\boldsymbol{N}, C - 1 - m_k) \prod_{m=0}^{m_k-1} \beta(\boldsymbol{N}, C - 1 - m) \\ \qquad + \sum_{k=1}^{P} m_k N_k \sigma_k \alpha_k(\boldsymbol{N}, C - m_k) \prod_{m=1}^{m_k-1} \beta(\boldsymbol{N}, C - m) \end{array} \right)}, \end{aligned}$$

from which, it follows that

$$\beta(\boldsymbol{N}, C) = \frac{C}{\left(\begin{array}{l} C + \sum_{k=1}^{P} m_k N_k \sigma_k [\alpha_k(\boldsymbol{N}, C - m_k) - \alpha_k(\boldsymbol{N}, C - 1 - m_k) \beta(\boldsymbol{N}, C - m_k)] \\ \qquad \cdot \prod_{m=1}^{m_k-1} \beta(\boldsymbol{N}, C - m) \end{array} \right)}. \tag{5.13}$$

To this end, we can show a numerically stable recursion to compute $U_k(\boldsymbol{N}, C)$. Let $\mathcal{N}_n$ $(n = 1, \ldots, |\boldsymbol{N}|)$ be

$$\mathcal{N}_n = \left\{ \boldsymbol{n} = (n_1, \ldots, n_P) \;\middle|\; \sum_{k=1}^{P} n_k = n, n_k \geq 0 \ (k = 1, \ldots, P) \right\}.$$

Using (5.12) and (5.13), we can compute $U_k(\boldsymbol{N}, C)$ in the following way.

1. Set for $k = 1, \ldots, P$,

$$\alpha_k(\boldsymbol{n}, c) = \begin{cases} 0 & (c < 0 \text{ or } n_k = 0) \\ 1 & (c = 0 \text{ and } n_k > 0) \end{cases}$$

   and set

$$\beta(\boldsymbol{0}, c) = \begin{cases} 0 & (c \leq 0) \\ 1 & (c \geq 1). \end{cases}$$

2. For $n = 1, \ldots, |\boldsymbol{N}|$,

   (a) for all $\boldsymbol{n} \in \mathcal{N}_n$, if $n_k > 0$, then compute $\alpha_k(\boldsymbol{n}, c)$ $(c = 1, \ldots, C)$ by (5.12),

   (b) for all $\boldsymbol{n} \in \mathcal{N}_n$, compute $\beta(\boldsymbol{n}, c)$ $(c = 1, \ldots, C)$ by (5.13).

3. Compute $U_k(\boldsymbol{N}, C)$ $(k = 1, \ldots, P)$ by (5.11).

Chapter 6

Concluding Remarks

In this dissertation, I have constructed theories for designing Internet Telephony Network. Recall Internet Telephony Network would be the next killer application to the world wide web. However, there are few theories to design Internet Telephony Network. Therefore, I have tried to organize design principles of Internet Telephony Network for end-to-end communications.

In Chapter 2, I have seen the whole of the network to give the big picture of end-to-end communications. My design architecture mainly has consisted of three types of networks, i.e., the domain networks, the access networks, and the stub networks. Among them, I have pointed out that issues arising in the former two networks for end-to-end QoS provisioning are important. The access networks cause large transmission delay because of small transmission capacity. On the other hand, the domain networks have relatively large transmission capacity, so that transmission delay time is insignificant. But, a scalability problem in terms of the number of flows is crucial because such a network often accommodates a huge number of flows.

In Chapter 3 and 4, I have treated the access networks, in which the most crucial issue is large delay time of a packet transmission due to relatively small capacity of links. It is known that large delay time heavily degrades the QoS. Therefore, limiting the maximum number of voice flows, which can keep delay time under a value, becomes an important issue. To evaluate such delay time in access network, I construct two models of access networks; homogeneous flows and heterogeneous

flows. The homogeneous flows mean that their packet transmission rates and their packet lengths are the constants. In my model, there are a number of voice flows and also data flows. Data flows are obstacles of voice flows. This is a preliminary but an important case, because widely spread dial-up access lines have very small capacity, which can accept only flows of small bandwidth. In the homogeneous case (Chapter 3), I obtain an exact and closed form solution, so that we do not need any approximations or numerical Laplace inversions. Schemes to reduce delay are proposed and their performance is evaluated.

Furthermore, in Chapter 4, I have investigated the heterogeneous flow case, which treats three types of traffic; voice, video, and data traffic. The model allows a multiplexing of different transmission rates and different packet lengths, e.g., accommodating both voice and video traffic. Similar to the homogeneous case, data flows are the obstacles of voice and video traffic. My numerical results show the large packet length of video packet has a large influence of voice delay.

Next, in Chapter 5, I have attempted to provide a design architecture of backbone networks. The backbone networks have a large transmission capacity, so that delay experienced by real-time packets, like voice and video, will be negligible because of its high speed packet forwarding capability. The bottleneck of backbone networks is state requirements. Real-time flows request to allocate their dedicated bandwidth in an end-to-end region. Consequently, the network nodes should maintain a number of states of active flows, and are desired to exchange the signaling messages for allocating some dedicated bandwidth to the real-time flows. For example, a high-speed link of 10 Gb/s can accommodate one million voice flows of 10 kb/s. Consequently, the state requirements will be of major concern in future high-speed networks. Furthermore, call blocking probability, which describes a network utilization from user points of view, has been also investigated. Through the numerical results, I have examined trade-offs between call blocking probability and the state requirements.

Finally, I would like to emphasize that my research covers both access networks and backbone networks. Both access and domain networks are an important and large portion of sub-networks on the Internet. The former networks have a bottleneck of link capacity, thus a delay performance is practically important. On the other hand,

the later networks have enough capacity of links, thus the number of flows becomes huge, as a consequent, the state management is of practical importance. To realize Internet Telephony as an advanced Internet service, I believe that this viewpoints separation is very important due to existence of a grate variety of sub-networks.

One more important thing is that I provide two suggestions about future research of Internet Telephony. First, in very small capacity of links, just layer-3's (or the network layer's) packet scheduling scheme cannot sufficient reduce the delay time (in Chapter 3 and 4). This might show a need of some layer-2 (or the datalink layer) packet segmentation schemes. The second suggestion is about flow managements in backbone networks (in Chapter 5). In this dissertation, I have considered that just one trunk case for each of edge-to-edge paths for optimizing the state requirements. However, this needs too much capacity than actually required one. To decrease the amount of capacity requirements, when the original trunk becomes full, a method of creating alternate trunk would be useful because such method can further improve network utilization. This is a future work.

Bibliography

[BBC+98] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and Weiss, W.: "An architecture for differentiated services", *IETF RFC2475*, Dec. 1998.

[BCD+00] Boyle, J., Cohen, R., Durham, D., Herzog, S., Rajan, R. and Sastry, A.: "The COPS (common open policy service) protocol", *IETF RFC2748*, Jan. 2000.

[BILFD99] Baker, F., Iturralde, C., Le Faucheur, F. and Davie, B.: "Aggregation of RSVP for IPv4 and IPv6 reservations", *IETF draft-baker-rsvp-aggregation-01.txt*, June 1999.

[BV98] Berson, S. and Vincent, S.: "Aggregation of internet integrated services state", *Proc. IFIP IWQoS'98*, May 1998.

[BZB+97] Braden, R., Zhang, L., Berson, S., Herzog, S. and Jamin, S.: "Resource reservation protocol (RSVP) — version 1 functional specification", *IETF RFC2205*, Sept. 1997.

[CEGK98] Côté, G., Erol, B., Gallant, M. and Kossentini, F.: "H.263+: Video coding at low bit rates", *IEEE Trans. Circuits Syst. Video Technol.*, Vol. 8, No. 7, pp. 849–866, Nov. 1998.

[Chl94] Chlebus, E.: "Mean-value analysis for examining call admission control thresholds in multi-service networks", in *ITC 14*, pp. 1207–1216, Elsevier, Amsterdam, Netherlands, 1994.

[CJ99] Casner, S. and Jacobson, V.: "Compressing IP/UDP/RTP headers for low-speed serial links", *IETF RFC2508*, Feb. 1999.

[CMT98] Claffy, K., Miller, G. and Thompson, K.: "The nature of the beast: recent traffic measurements from an internet backbone", in *Proc. ISOC INET'98*, Geneva, Switzerland, July 1998.

[Coo90] Cooper, R. B.: *Introducation to Queueing Theory*, CEEPress, Washington, D.C., 3rd edition, 1990.

[Cox97] Cox, R.: "Three new speech coders from the ITU cover a range of appli cations", *IEEE Commun. Mag.*, pp. 40–47, Sept. 1997.

[Dos90] Doshi, B.: "Generalizations of the stochastic decomposition results for single server queues with vacations", *Commun. Statist.-Stochastic Models*, Vol. 6, No. 2, pp. 307–333, 1990.

[Dou00] Douskails, B.: *IP Telephony: The Integration of Robust VoIP Services*, Prentice Hall, Upper Saddle River, NJ, USA, 2000.

[DRS91] Dron, L., Ramamurthy, G. and Sengupta, B.: "Delay analysis of continuous bit rate traffic over an ATM network", *IEEE J. Select. Areas Commun.*, Vol. 9, No. 3, pp. 402–407, Apr. 1991.

[FH98] Ferguson, P. and Huston, G.: *Quality of Service: Delivering QoS on the Internet and in Corporate Networks*, Wiley, Jan. 1998.

[FJ95] Floyd, S. and Jacobson, V.: "Link-sharing and resource management models for packet networks", *IEEE/ACM Trans. Networking*, Vol. 3, No. 4, pp. 365–386, Aug. 1995.

[GBH97] Guerin, R., Blake, S. and Herzog, S.: "Aggregating RSVP-based QoS requests", *IETF draft-guerin-aggreg-rsvp-00.txt*, Nov. 1997.

[Goo99] Goodman, B.: "Internet telephony and modem delay", *IEEE Network Mag.*, pp. 8–16, May/Jun. 1999.

[HBH93] Humblet, P., Bhargava, A. and Hluchyj, M.: "Ballot theorems applied to the transient analysis of nD/D/1 queues", *IEEE/ACM Trans. Networking*, Vol. 1, No. 1, pp. 81–95, Feb. 1993.

[Hel98] Held, G.: *Voice Over Data Networks*, McGraw-Hill, New York, NY, USA, Apr. 1998.

[HGP00] Hersent, O., Gurle, D. and Petit, J. P.: *IP Telephony: Packet-based multimedia communications systems*, Addison-Wesley, 2000.

[Hui90] Hui, J. Y.: *Switching and Traffic Theory for Integrated Broadband Networks*, Kluwer, Boston, 1990.

[IKTO99] Iida, K., Kawahara, K., Takine, T. and Oie, Y.: "Performance analysis of flow aggregation of constant bit rate type traffic at ingress router", in *Proc. IEEE Globecom'99*, pp. 92–99, Rio de janeiro, Brasil, Dec. 1999.

[IKTO00] Iida, K., Kawahara, K., Takine, T. and Oie, Y.: "Performance evaluation of the architecture for end-to-end quality-of-service provisioning", *IEEE Commun. Mag.*, Vol. 38, No. 4, pp. 76–81, Apr. 2000.

[ITSO01a] Iida, K., Takine, T., Sunahara, H. and Oie, Y.: "Delay analysis for CBR traffic in multimedia enterprise network", *IEICE Trans. Commun.*, Vol. E84-B, No. 4, pp. 1041–1052, Apr. 2001, Earlier version has been presented in *Proc. IEEE Globecom'98*, pp. 1256–1263, Sydney, Australia, Nov. 1998.

[ITSO01b] Iida, K., Takine, T., Sunahara, H. and Oie, Y.: "Delay analysis for CBR traffic under static-priority scheduling", *IEEE/ACM Trans. Networking*, Vol. 9, No. 2, pp. 177–185, Apr. 2001, Preliminary version has been presented at SPIE Conference of Performance and Control of Networks, in *Proc. SPIE*, Vol. 3231, pp. 311–322, Dallas, TX, USA, Nov. 1997.

[ITU95] ITU-T Rec. G.723.1, : "Dual rate speech codec for multimedia telecommunications transmitting at 6.4 and 5.3 kbit/s", 1995.

[ITU98] ITU-T Rec. H.263 version 2, : "Video coding for low bit rate communication", Jan. 1998.

[Jag82] Jagerman, D.: "An inversion technique for the Laplace transform", *Bell Syst. Tech. J.*, Vol. 61, No. 8, pp. 1995–2002, Oct. 1982.

[K+98] Kostas, T., et al.: "Real-time voice over packet-switched networks", *IEEE Network Mag.*, pp. 18–27, Jan./Feb. 1998.

[Kes97] Keshav, S.: *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network*, Addison-Wesley, Reading, MA, USA, 1997.

[KR01] Kurose, J. F. and Ross, K. W.: *Computer Networking: A Top-Down Approach Featuring the Internet*, Addison-Wesley, 2001.

[Par93] Partridge, C.: *Gigabit Networking*, Addison-Weslay, Reading, MA, USA, Oct. 1993.

[PG93] Parekh, A. K. and Gallager, R. G.: "A generalized processor sharing approach to flow control in integrated services networks: The single-node case", *IEEE/ACM Trans. Networking*, Vol. 1, No. 3, pp. 344–357, June 1993.

[PG94] Parekh, A. K. and Gallager, R. G.: "A generalized processor sharing approach to flow control in integrated services networks: The multiple-node case", *IEEE/ACM Trans. Networking*, Vol. 2, No. 2, pp. 137–150, Apr. 1994.

[PS98] Privalov, A. and Sohraby, K.: "Per-stream jitter analysis in CBR ATM multiplexors", *IEEE/ACM Trans. Networking*, Vol. 6, No. 2, pp. 141–149, Apr. 1998.

[Rij96] Rijkse, K.: "H.263: Video coding for low-bit-rate communication", *IEEE Commun.*, pp. 42–45, Dec. 1996.

[Ros95] Ross, K. W.: *Multiservice Loss Models for Broadband Telecommunication Networks*, Springer, Berlin, 1995.

[RS91] Ramamurthy, G. and Sengupta, B.: "Delay analysis of a packet voice multiplexer by the $\sum D_i/D/1$ queue", *IEEE Trans. Commun.*, Vol. 39, No. 7, pp. 1107–1114, July 1991.

[RV91] Roberts, J. and Virtamo, J.: "The superposition of periodic cell arrival streams in an ATM multiplexer", *IEEE Trans. Commun.*, Vol. 39, No. 2, pp. 298–303, Feb. 1991.

[SCFJ96] Schulzrinne, H., Casner, S., Frederick, R. and Jacobson, V.: "RTP: A transport protocol for real-time applications", *IETF RFC1889*, Jan. 1996.

[Sch99] Schaphorst, R.: *Videoconferencing and Videotelephony: Technology and Standards*, Artech House, Norwood, MA, USA, second edition, 1999.

[Sen89] Sengupta, B.: "An invariance relationship for the G/G/1 queue", *Advances in Applied Probability*, Vol. 21, pp. 956–957, 1989.

[Sen90] Sengupta, B.: "A queue with superposition of arrival streams with an application to packet voice technology", in King, P., et al. eds., *Performance'90*, pp. 53–59, North-Holland, Elsevier Science, Amsterdam, Netherlands, 1990.

[Ste94] Stevens, W. R.: *TCP/IP Illustrated, Volume 1: The Prottocols*, Addison-Wesley, Reading, MA, USA, 1994.

[Tak96] Takine, T.: "A continuous version of matrix-analytic methods with the skip-free to the left property", *Stochastic Models*, Vol. 12, pp. 673–681, Nov. 1996.

[Wan01] Wang, Z.: *Internet QoS: Architectures and Mechanisms for Quality of Service*, Moragn Kaufmann, San Francisco, CA, USA, 2001.

[WC99] Wenger, S. and Côté, G.: "Using RFC2429 and H.263+ at low to medium bit-rates for low-latency applications", in *Proc. Packet Video'99*, New York, NY, USA, Apr. 1999.

[Wen97] Wenger, S.: "Video redundancy coding in H.263+", in *Proc. Int. Workshop on Audio-Visual Services over Packet Networks*, Aberdeen, Scotland, UK, Sept. 1997.

[WKOK98] Wenger, S., Knorr, G., Ott, J. and Kossentini, F.: "Error resilience support in H.263+", *IEEE Trans. Circuits Syst. Video Technol.*, Vol. 8, No. 7, Nov. 1998.

[WLSC98] Willebeek-LeMair, M., Shae, Z.-Y. and Chang, Y.-C.: "Robust H.263 video coding for transmission over the internet", in *Proc. IEEE Infocom'98*, pp. 225–232, Mar. 1998.