Reducing and Characterizing Packet Loss for High-Speed Computer Networks with Real-Time Services

A Dissertation Presented

by

Henning G. Schulzrinne

Submitted to the Graduate School of the
University of Massachusetts in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

May 1993

Department of Electrical and Computer Engineering

Reducing and Characterizing Packet Loss for High-Speed Computer Networks with Real-Time Services

A dissertation Presented

by

Henning G. Schulzrinne

Approved as to style and content by:

James F. Kurose, Chair of Committee

Donald F. Towsley, Member

Christos G. Cassandras, Member

Wei-Bo Gong, Member

Aura Ganz, Member

Lewis E. Franks, Acting Department Head
Department of Electrical and Computer Engineering

To Carol, Nathan Paavo and my parents

Acknowledgements

It is with great pleasure that I acknowledge the encouragement, insight and valuable advice of Professor James F. Kurose and Professor Donald F. Towsley, without whom this work would not have been possible. I feel very fortunate to have had them as my advisors. The opportunities for travel and meeting fellow researchers provided by them allowed me to broaden my professional horizon.

Appreciation is also due to Professor Christos Cassandras for help in charting the course through the thesis process, his technical guidance and making workstations available for some of the kernel and Nevot work. The editing efforts of Professor Weibo Gong and Aura Ganz are gladly acknowledged.

The work was supported financially in part by the Office of Naval Research under contract N00014-87-K-304 and N00014-90-J-1293, the Defense Advanced Research Projects Agency under contract NAG2-578, an NSF equipment grant, CERDCR 8500332 and a fellowship from the graduate school of the University of Massachusetts at Amherst. DARTnet is funded by the Defense Advanced Research Projects Agency.

I have enjoyed the priviledge of sharing spirited discussions with fellow graduate students, in particular the members of our research group both during our research group meetings and in one-on-one conversations.

Charles Lynn (Bolt, Beranek and Newman in Cambridge) helped with navigating the intricacies of the ST-II protocol implementation; Karen Seo (BBN) helped to establish and foster the connection between our research group and BBN. Steve Deering (Xerox Parc) elucidated IP multicast issues. Steve Casner (Information Sciences Institute) and Allison Mankin (MITRE Corporation) supported this research by making the loan of a video codec possible. I am also grateful to Steve

Casner, Van Jacobsen (LBL) and Eve Schooler (ISI) and for sharing their expertise in audio conferencing. Lixia Zhang (Xerox Parc) quick answers to my questions regarding her FIFO+ experiments were much appreciated, as well as the discussions with David Clark (MIT) on resource control in networks. It has been a pleasure to become part of the DARTnet research community through meetings and through the regular teleconferences.

The friendship of the members of First Congregational Church has enriched my stay in Amherst and have made Amherst a home to return to. Finally, I would like to thank my wife Carol and my parents for their encouragement, understanding and support.

Abstract

Henning G. Schulzrinne

Reducing and Characterizing Packet Loss for High-Speed Computer Networks with Real-Time Services

May 1993

B.S., Technische Hochschule Darmstadt

(Federal Republic of Germany)

M.S., University of Cincinnati

Ph.D., University of Massachusetts

Directed by: Professor James F. Kurose

Higher bandwidths in computer networks have made application with real-time constraints, such as control, command, and interactive voice and video communication feasible. We describe two congestion control mechanisms that utilize properties of real-time applications. First, many real-time applications, such as voice and video, can tolerate some loss due to signal redundancy. We propose and analyze a congestion control algorithm that aims to discard packets if they stand little chance of reaching their destination in time as early on their path as possible. Dropping late and almost-late packets improves the likelihood that other packets will make their deadline.

Secondly, in real-time systems with fixed deadlines, no improvement in performance is gained by arriving before the deadline. Thus, packets that are late and have many hops to travel are given priority over those with time to spare and close to their destination by introducing a hop-laxity priority measure. Simulation results show marked improvements in loss performance. The implementation of the algorithm

within a router kernel for the DARTnet test network is described in detail. Because of its unforgiving real-time requirements, packet audio was used as one evaluation tool; thus, we developed an application for audio conferencing. Measurements with that tool show that traditional source models are seriously flawed.

Real-time services are one example of traffic whose perceived quality of service depends not only on the loss rate but also on the correlation of losses. We investigate the correlation of losses due to buffer overflow and deadline violations in both continuous and discrete-time queueing systems. We show that loss correlation does not depend on value of the deadline for $M/G/1$ systems and is generally only weakly influenced by buffer sizes. Per-stream loss correlation in systems with periodic and Bernoulli on/off sources are evaluated analytically. Numerical examples indicate that loss correlation is of limited influence as long as each stream contributes less than about one-tenth of the overall network load.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Computer networks are in a transition period, moving from relatively slow communication links and data-oriented services to high-speed links supporting a diverse set of services, including those such as video and voice with stringent real-time constraints. These real-time services demand not only high bandwidth, but a predictable quality of service not offered by current best-effort delivery networks. Clearly, scaling bandwidths by a factor of thousand or more is bound to have profound effects on all aspects of networking, but supporting the more diverse mix of services raises issues that go beyond mere bandwidth and bandwidth-delay product scaling.

Real-time services also shift the emphasis from throughput and delay as the preeminent network performance metrics to packet loss caused by congestion. Traditional data services use retransmission schemes to hide network packet loss from the application; packet loss is only reflected in the throughput and delay characteristics. A 10% packet loss, for example, reduces throughput by a barely noticeable 10% if the retransmission algorithm is implemented efficiently, but could well make an audio or video connection unusuable[1]. Similarly, a doubling of the end-to-end delay from 50 to 100 ms would be barely noticed for a file transfer, but could lead to severe talker echo in a voice conference.

For most high-speed connections[2], packet losses are primarily due to buffer overflows in switching nodes and deadline violations rather than channel noise. The

[1] Due to delay constraints, retransmission is usually not possible with real-time services

[2] excluding, for example, packet radio or analog modems

problem of buffer overflows is aggravated by bursty input traffic and the large amount of data "stored" by the transmission medium, which makes feedback controls difficult to implement.

This dissertation proposes and evaluates algorithms to reduce congestion-induced losses in packet-switched networks carrying traffic with real-time constraints. In addition to the packet losses due to channel errors and queue overflows experienced by all network services, real-time services also lose packets that violate their end-to-end delay constraints. Two complementary heuristics for reducing loss caused by deadline violation are presented and their performance is analyzed. The first heuristic selectively discards packets within the network that either have, or are likely to exceed, their delay bounds. Approximate analytical methods for tandem queues are shown to provide good engineering estimates of the loss probabilities. The network configurations, load and deadline regimen under which selective packet dropping provides are identified, showing realistic networks that can reduce such packet loss by a factor of two. The second heuristic, called hop-laxity scheduling, schedules packets for transmission on an outgoing link based on the ratio of the time available to their deadline expiration and the number of hops needed to reach the destination. This heuristic can be used to reduce deadline-violation losses (or, equivalently, provide similar quality of service to packet flows traversing network paths of different length and cross traffic loading). The heuristic is evaluated both by simulation and by measuring the performance of an implementation within an experimental transcontinental network using audio and video traffic streams. A number of tools for traffic generation were also developed as part of this work and are discussed in this thesis.

The dissertation also provides analytical methods that allow the characterization of the structural properties of packet losses in queues subject to buffer size limitations or delay bounds. The loss properties are characterized here by measures of loss correlation. Both queues serving single traffic streams as well as queues multiplexing

a number of correlated sources are analyzed. In general, we find that loss correlation is either independent of, or only very weakly correlated with, the buffer size or delay threshold. The influence of other factors, such as traffic burstiness, is quantified.

1.1 Organization

The dissertation is organized as follows. First, we propose and analyze algorithms that relieve congestion by selectively discarding packets (Chapter 2). As a complementary approach, a number of queue scheduling disciplines that strive to maximize the amount of on-time arrivals are discussed in Chapter 3. Through simulations, we find that hop-laxity scheduling appears particularly promising. Both for selective discarding and deadline-based scheduling, we emphasize end-to-end methods rather than concentrating on a single queue. The implementation of the hop-laxity scheduling algorithm within an experimental network is discussed in Chapter 4. We discuss necessary protocol extensions for implementing hop-laxity scheduling within the IP and ST-II Internet network layer framework. Also summarized are the kernel modifications required to support this and other similar scheduling methods. We also highlight the pertinent features of a voice conference application known as NEVOT, which we developed to provide realistic traffic sources in our experimental setting.

The structural properties of loss are addressed in Chapters 5 and 6, where we analyze the time and correlation properties of packet loss in a large class of queueing systems, both with a single and multiple arrival streams, where losses are caused by either buffer overflow or deadline violations.

Note that, due to the diversity of topics covered, each major chapter contains its own literature review.

1.2 Contributions

The contributions of this thesis can be summarized as follows:

- **development of a light-weight congestion control mechanism for real-time services in networks with high delay-bandwidth product:** The mechanism attempts to reduce congestion by removing packets that are unlikely to meet their end-to-end deadline. Approaches for choosing the parameters of the algorithm are presented.

- **end-to-end performance evaluation of a multi-hop queueing network employing the proposed congestion control method:** Using transform-domain techniques, the end-to-end performance of the selective discarding heuristic is studied. The approximate analysis is found to provide good conservative estimates of actual network performance and is suitable for numeric parameter optimization.

- **development of a link-level deadline-based scheduling policy for real-time services that is aware of end-to-end performance requirements:** The policy is shown to reduce delay-induced losses for many network configurations and equalizes the delay of flows traversing paths with different number of hops and network loading.

- **implementation of the proposed scheduling policy in an operating router:** The scheduling policy was implemented in a standard BSD-based operating system in the SPARCstation routers operating within the DARTnet cross-country network testbed and its performance measured and compared to FIFO queueing. Realistic traffic sources based on actual voice and video conferences, rather than statistical models, were used for the evaluation. Based on the experiences with implementing non-FIFO disciplines with a BSD Unix kernel,

suggestions are offered on implementing a more general-purpose interface to transmission scheduling.

- **tool for audio conferencing:** The work creating NEVOT a tool for audio conferencing, allowed us to explore issues in implementing real-time services in non-real-time operating systems. The tool also serves as a traffic generator for the network, either directly or through its extensive tracing facility.

- **trace-driven simulation and network traffic generation tools:** These program provide tools to the experimenter to recreate a packet interarrival time trace on an actual network, thus facilitating the comparison of simulation with actual network performance and the reconstruction of fault scenarios.

CHAPTER 2

CONGESTION CONTROL FOR REAL-TIME TRAFFIC IN HIGH-SPEED NETWORKS

2.1 Introduction

The higher bandwidths promised by broadband integrated services digital networks (BISDN) have made applications with real-time constraints, such as control, command, and interactive voice and video communications, feasible. Many types of real-time traffic are characterized by "perishable", but redundant messages. In other words, excessive delay renders them useless, but a certain degree of loss can be tolerated without objectionable degradation in the grade of service. Real-time packets are lost for several reasons. The packet may arrive at the receiver after the end-to-end deadline has expired because it suffered excessive waiting times in the intermediate nodes (*late loss*). Also, queues may overflow or intermediate nodes may shed load by dropping packet as an overload control measure (*drop loss*).

The tolerance for packet loss varies with the type of traffic carried and the measures the receiver is prepared to take to reconstruct lost packets. For speech coded with PCM and adaptive DPCM, losses from 2 to 5% are tolerable without interpolation, while odd-even interpolation raises the threshold of objectionable loss to between 5% and 10% [1]. Compressed video is far less tolerant of lost packets. A variable-rate DPCM coder operating at around 30 Mb/s [2] is designed for a loss rate of below 10^{-11}.

The goal of this chapter is to investigate controlling congestion for real-time traffic by judiciously discarding packets at the intermediate nodes. Packets that stand little chance of making their end-to-end deadline should be discarded as early

in the virtual circuit as possible. Discarding applied with care has two beneficial effects. First, *instantaneous* congestion at the discarding node is relieved. This type of congestion is caused by the statistical fluctuations in the arrival process. Secondly, downstream nodes are not burdened with traffic that in all likelihood will not meet its deadline at the receiving end, speeding up service for the remaining packets. Reduced downstream load counteracts *longer-term* congestion, expressed as high average load.

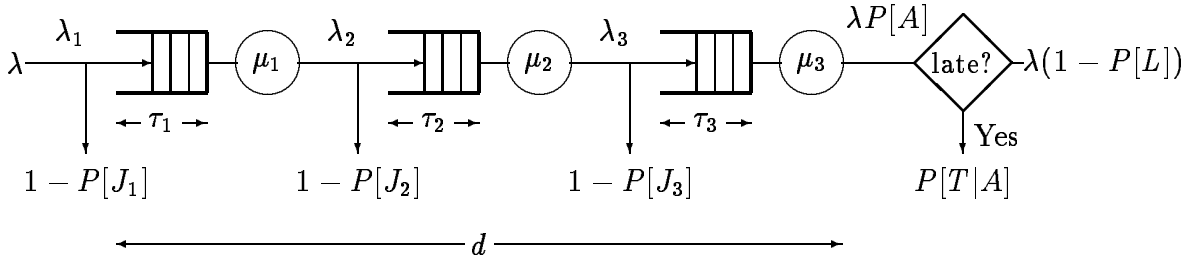

Figure 2.1. Sample virtual circuit

Barring clairvoyance, we have four choices in discarding packets, depending on our degree of caution and amount of available information (see Fig. 2.1 for notation):

1. discard randomly

2. discard a packet if the time spent at the node under consideration exceeds the end-to-end deadline d

3. discard a packet whose time spent in the virtual circuit so far, including time in current node, exceeds the end-to-end deadline d

4. discard a packet if the time to be spent in current node i exceeds a fixed *local* deadline τ_i

The first scheme, random discarding, has been widely studied [3] [4] [5] [6], and generally been found to be lacking as a congestion control mechanism. It, for

example, does not substantially improve the disproportionate probability that long round-trip time traffic is dropped [4]. We will further compare random dropping to our schemes.

The next two schemes above are conservative in the sense that no packet will ever be dropped voluntarily at the intermediate nodes that would not have been late at the destination. The third approach risks the possibility of discarding packets that would have made the end-to-end deadline. The parameter space investigated will cover the first and third possibility, as the second requires "travel history" to be carried with each packet.

We note that in a high-speed wide-area environment, control measures such as flow-control windows, backpressure or choke packets [7] lose effectiveness since the high delay-to-bandwidth ratio may render the feedback information obsolete by the time it reaches the controlling agent. Because of this, local, distributed control mechanisms are preferable to centralized algorithms. Also, not all real-time sources can be readily rate-controlled (e.g., standard PCM voice). Even in high-speed networks, window-based flow control with appropriate feedback may still be suitable for avoiding congestion during file transfers [8–14]. Resource allocation may avoid congestion, but typically with the penalty of reduced carrying capacity if streams do not utilize their allocated bandwidth. For a survey of resource allocation policies, see [15].

After outlining the system model underlying this thesis in section 2.2 and the general equations governing packet loss in section 2.3, we will in turn investigate two queueing models for the proposed control mechanism, first-in–first-out with bounded *system* time, referred to for brevity as FIFO-BS (section 2.4), and first-in–first-out with bounded *waiting* time (FIFO-BW) in section 2.5.[1] For each, the expressions for the conditional distribution of the system or waiting time available in the literature

[1]In this thesis, *system time* is defined to include service, i.e., transmission, time. *Waiting time* refers to the delay between arrival and start of service.

will be reviewed and extended where necessary. Expressions and numerical values for the end-to-end delay distributions are presented and compared to simulation results. Node rejection probability and the tail of the delay distribution establish the end-to-end loss probability under the various congestion control schemes. A simple scheme of using the same value for all local deadlines in the VC performs will be shown to perform almost as well as the more difficult scheme of selecting optimal local deadlines for each node. We discuss implementing these policies in high-speed networks and show that they compare favorably to random discarding as a congestion countermeasure. We conclude this chapter with some alternative approaches and future work in section 2.7.

2.2 System Model and Notation

This work is primarily concerned with packet voice traffic. The approaches are also applicable to other real-time traffic, such as sensor data and control commands, assuming that it shares a similar tolerance for loss and similar traffic characteristics.

We study virtual circuits (VCs) with nodes labeled $i = 1, 2, \ldots M$, typically switching nodes in a wide-area network interconnected by high-bandwidth fiber optic links. Packet losses caused by bit errors and other hardware faults are ignored as the probability of their occurrence falls several orders of magnitude below that of buffer-related losses. Packet arrivals to all nodes are approximated by Poisson processes with arrival rate λ_i. Modeling a superposition of SAD (speech activity detection) voice sources as a Poisson source leads to overly optimistic estimates in the case of a voice multiplexer [16]. However, for the case of voice trunks considered here, the number of voice sources is possibly two orders of magnitude higher than that found at the T1 rate (1.544 Mb/s) multiplexer studied by [17] and others. A 150 Mbit channel can support 4380 64 kB/s PCM sources of speech activity factor 0.42 with a utilization of 0.8. Thus, according to the arguments of [18], we can conclude that for short time spans, the superposition process is adequately modeled by a

Poisson process. The discarding mechanism further limits the interaction period of packets in the queue, improving the approximation [16].

Even assuming that the input traffic to the first node is indeed modeled accurately by a Poisson source, packet dropping at the intermediate nodes will make the input stream to subsequent nodes non-Poisson. The analysis, however, ignores this, based on the premise that the relatively small losses investigated should have limited effects on the delay distribution. Simulation experiments will be used to assess the validity of this assumption, which is also supported by [19, 20].

The service time is assumed to be exponentially distributed with mean $1/\mu$, redrawn at each node (Kleinrock's independence assumption). Without loss of generality, all times are scaled relative to the average service time, i.e., $\mu = 1$. Since we are primarily interested in communication channels, we assume that the packet transmission and waiting time is proportional to the number of bits in the packet or queue, respectively. Also, the service time of each packet is assumed to be known at the instant of arrival to a queue. (The results for FIFO-BW do not change if the packet leaves the queue if it has not started service within the local deadline; the service time of an individual packet does not enter into consideration. For FIFO-BS, we have to assume that the service time is known on joining the queue.)

We investigate a single virtual circuit in isolation, taking into account interfering traffic by reducing the service rate, as in [21]. The length of the virtual circuit obviously depends on the system architecture, but examples in the literature [22] suggest a range of four to eleven, with the values at the lower end appearing more often. Following [23], most of the examples will have five nodes.

In summary, the above assumptions ensure independence of service and arrival sample paths for each node. Nodes are linked only by the reduction in traffic afforded by dropping certain packets. These assumptions are necessary for analytic tractability, but as discussed above, will be validated through simulation.

A number of researchers have investigated the effect of packet dropping on a single packet voice multiplexer (e.g., [24–26], and that of bit dropping on a virtual circuit [19], but the performance of packet dropping in a VC seems to have been considered only in the context of an ARQ-scheme with variable window size [27]. The latter scheme requires acknowledgements, which are not typically used for real-time traffic, and uses feedback, with the concomitant delay problems. Most importantly, packet dropping is performed without regard to the packet's time constraint.

In the following, probability density functions will be denoted by lower case letters, with w and s standing for waiting and system time, respectively. The respective uppercase letters represent the corresponding cumulative distribution functions. An asterisk indicates the one-sided Laplace transform.

2.3 Packet Loss in Virtual Circuit

This section outlines some of the general relations governing packet loss in a virtual circuit, independent of the particular queueing model used for the individual nodes. A packet is considered lost if it is either discarded by any one of the M nodes it traverses (because it missed the local deadline τ_i) or if the total system (FIFO-BS case) or the waiting time (FIFO-BW) exceeds a set, fixed end-to-end deadline d (see Fig. 2.1).

Components of Loss Probability

The arrival event A occurs if a packet reaches its destination, i.e., is not dropped by any of the queue controllers along the VC. In a tandem-$M/M/1$ system, A occurs with certainty. The complementary event will be referred to as D and its probability as the *drop loss*. J_i represents the event that a packet joins queue i (given that it has traversed nodes 1 through $i-1$), while the event of a lost packet, regardless of cause, will be labeled as L. The probability of an event is shown as $P[\cdot]$.

The probability that a packet is not dropped in any of the M (independent) nodes is the destination arrival probability $P[A] = \prod_{i=1}^{M} P[J_i]$ In other words, $P[A]$ does not include the probability that a packet reaching the destination misses its deadline. In computing $P[J_i]$, the reduced traffic due to packets dropped in nodes $0, 1, \ldots, i-1$ needs to be taken into account, where $\lambda_i = \lambda_{i-1} P[J_{i-1}]$.

Thus, given that the end-to-end conditional cumulative distribution of system time for non-discarded packets is $S(d|J) = P[S \leq d|J]$, the total loss probability $P[L]$, encompassing both drop loss and late loss, can be computed by the law of total probabilities as

$$P[L] = (1 - S(d|A))P[A] + (1 - P[A]) = 1 - S(d)P[A].$$

Here, the conditioning on J indicates that we only take packets that were not dropped into consideration. The first part of the equation shows the contribution of dropped and late packets to the total loss. In the case of voice, $1 - P[L]$ is the fraction of packets that are actually played out at the receiver.

Since each link is assumed to be independent of all others, the conditional density of the end-to-end system time, $s(t|A)$, is derived from the convolution of the single-node conditional densities, $s_i(t|A)$. By the convolution property of the Laplace transform we obtain,

$$s(t|J) = s_1(t|J) * \ldots * s_M(t|J) = \mathcal{L}^{-1} \left\{ \prod_{i=1}^{M} s_i^*(s|J) \right\} \tag{2.1}$$

where $\mathcal{L}^{-1}$ represents the inverse Laplace transform.

2.4 FIFO-BS: Bounded System Time

In the first of the two congestion control policies considered, a packet joins queue i $(i = 1, \ldots, M)$ only if the virtual work, that is, the amount of time it would take to empty the queue with no new arrivals, plus the service time of the arrival is less than the local deadline, τ_i. In other words, the time spent in the queueing system,

including service, is bounded from above by τ_i. In this section, the results provided in [28,29] will be simplified to a form more amenable to computation. Some of the tedious algebra has been relegated to a separate report [30].

Laplace Transform of the Pdf of the Virtual Waiting Time

The pdf of the waiting time, $w(t)$, can be expressed as $w(t) = Qh(t)$, where the function $h(t)$ is of no further interest for our purposes. The normalization factor Q is defined as

$$Q = (1-\rho)\left[1 - \rho a + \sum_{n=1}^{\infty} \frac{\rho^n (b^n - a)}{(\rho)_n}\right]^{-1}, \rho \neq 1. \tag{2.2}$$

where $\rho = \lambda/\mu$, $b \equiv e^{-\mu\tau}$, $a \equiv b^{1-\rho} e^{\rho(b-1)}$. $(x)_n = x(x+1)(x+2)\ldots(x+n-1)$ denotes Pochhammer's symbol.

The Laplace transform of the virtual waiting time is derived in [28]. The transform expression can be immediately simplified with the identity , $(i+1) = i!$, yielding in our notation:

$$\begin{aligned} w^*(s) &= w^*(-1) b^{1+s} \left\{1 + (1+s) \sum_{i=0}^{\infty} \rho^i \frac{, (\rho - s - 1)}{, (\rho - s + i)}\right\} \\ &\quad - Q(1+s) \sum_{i=0}^{\infty} (\rho b)^i \frac{, (\rho - s - 1)}{, (\rho - s + i)}. \end{aligned} \tag{2.3}$$

(Throughout this section, we are only concerned with the case $\rho \neq 1$.)

By making use of the identities , $(\rho + i) =$, $(\rho)(\rho)_i$ and , $(\rho - 1) = \frac{\Gamma(\rho)}{\rho - 1}$, we can avoid the evaluation of the gamma function with complex arguments:

$$\begin{aligned} w^*(s) &= Qab^s \left\{1 + \frac{1+s}{\rho - s - 1} \sum_{i=0}^{\infty} \frac{\rho^i}{(\rho - s)_i}\right\} \\ &\quad + Q \frac{1+s}{\rho - s - 1} \sum_{i=0}^{\infty} \frac{(\rho b)^i}{(\rho - s)_i}. \end{aligned}$$

Thus, the transform equation for the density of the system time of a single node is

$$s^*(s) = \frac{1}{P[J]} \frac{\mu}{s + \mu} w^*(s).$$

The technical report [30] also contains closed-form expressions for the cumulative distribution functions of the waiting and system time of a single queue.

2.5 FIFO-BW: Bounded Waiting Time

Arriving customers join queue i if and only if the virtual work at queue i at the arrival instant is below τ_i. In contrast to the FIFO-BS case discussed above, the service time of the arriving customer does not enter into the admission decision, thus eliminating the bias against long packets present in FIFO-BS. Here, the time a packet spends waiting for service to begin is bounded by τ_i. Results for this system are summarized in [31] (called FIFO-TO there). Among the policies for a single queue studied in that paper, FIFO-BW performed best in the sense of maximizing the throughput of packets meeting their deadlines, suggesting that it may be a good candidate for a tandem system as well. As in the previous section, we will extend the results for a single queue and then proceed to obtain closed-form expressions for the densities and distributions of the waiting time of a tandem system. For brevity, only results for $\rho \neq 1$ will be shown here.

A customer's conditional waiting time is distributed with density

$$w(t|J) = \frac{\nu\rho e^{-\nu t}}{\alpha} u(\tau - t)u(t) + \frac{1-\rho}{\alpha}\delta(t),$$

where $u(t)$ is the unit step function and $\alpha = \rho e^{-\nu\tau}$.

Recognizing $u(\tau - t) = 1 - u(t - \tau)$, the waiting time in the Laplace domain is derived:

$$w^*(s|J) = \frac{1}{\alpha}\left[\frac{\nu\rho}{s+\nu}\left\{1 - e^{-(s+\nu)\tau}\right\} + (1-\rho)\right]. \tag{2.4}$$

Extending the results in [31], the nth moment can be written as

$$E[w^n] = \frac{\rho}{\alpha\nu^n}\left[n! - , (n+1, \nu\tau)\right] = \frac{n}{\nu}E[w^{n-1}] - \frac{\rho\tau e^{-\nu\tau}}{\alpha}$$

where , (n, x) is the complementary incomplete Gamma function.

The IMSL routine DINLAP is able to invert Eq. (2.1) (with waiting time replacing system time), using Eq. (2.4), with desired accuracy only for small M and losses above 1%. Therefore, it was attempted to obtain tight bounds on the

tail of the waiting time distribution by using the moment generating function in the Chernoff bound or higher moments in Bienaymé's inequality [32]. Both produced bounds too loose to judge the benefits of applying queue control via local deadlines. An algebraic inversion of the Laplace-transform expression proved more fruitful. This was possible because unlike the transform for the FIFO-BS case (Eq. (2.3)), Eq. (2.4) is rational and thus amenable to closed-form expansion. As a general strategy, the product of sums is expanded into a sum of products and each term in the sum transformed separately.

The multiplicity of poles in the Laplace-domain convolution expression and the potential for analytic and numerical simplification make it expedient to distinguish four cases when inverting the transform. The cases are classified by the dependence of the local deadline τ_i and the traffic parameter $\nu_i = \lambda_i - \mu_i$ on the node index:

1. homogeneous traffic and local deadlines ($\nu_i = \nu_j, \tau_i = \tau_j \; \forall \, i, j$)

2. homogeneous traffic, but arbitrary local deadlines

3. strictly heterogeneous traffic ($\nu_i \neq \nu_j \; \forall \; i \neq j$) and arbitrary local deadlines

4. arbitrary traffic and deadlines, i.e., $\nu_i = \nu_j$ is allowed

The closed-form expressions for calculating losses for these four cases appear in the appendix.

Performance Evaluation: Numerical Examples and Validation

Having derived expressions for the cdf of the end-to-end system and waiting times, we can now compare the performance of the suggested queue control mechanisms. Because of space limitations, we will restrict our discussion to the FIFO-BW case. In this section, the end-to-end deadline d is fixed so that the packet loss equals a given value when no queue control is applied ("uncontrolled loss").

We first consider the case of homogeneous local deadlines, i.e., $\tau_i = \tau_j = \tau$. It is clear that the optimal value of τ has to fall between d/M and d since for values of τ less than or equal to d/M, all packets which are not dropped will make their end-to-end deadline; For $\tau \geq d$, any packet dropped at an intermediate node would be late on arrival at the destination.

Since relatively high load and loss should expose the deficiency of ignoring the non-Poisson nature of input streams, the first example uses the parameter set $M = 5$, $\mu = 1$, $\lambda_1 = 0.8$ and $d = 40.47$, resulting in an uncontrolled loss of 5%. The simulation maintains Kleinrock's independence assumption, but not the Poisson nature of the interior arrival streams. The simulation was terminated once the 95%-confidence interval halfwidths computed by spectral estimation [33] and the regenerative method were both less than 10% of the point estimate. Depending on τ, a simulation run consisted of between one and five million packets. An initial transient period estimated at 3000 packets was discarded.[2]

Fig. 2.2 compares analytical and simulation results for the first example, plotting the total loss, composed of packets dropped in intermediate nodes and packets missing their end-to-end deadline, as a function of the local deadline τ. The horizontal line at 5% loss indicates the asymptotic uncontrolled case with $\tau = \infty$. The graph shows that the analytical results overestimate end-to-end losses, with simulation and analysis agreeing more closely as τ increases towards infinity and the system approaches a tandem-M/M/1 queueing system. Fortunately, the optimal nodal deadlines seem to fall in the same neighborhood for both analysis and simulation. For this set of parameters, the FIFO-BW dropping mechanism reduces total losses from 5% to 3.1% according to the analysis and to 2.4% according to the simulation. From Fig. 2.2 we can also conclude that the drop losses incurred by tight deadlines are not compensated for by the reduction in downstream traffic.

[2]The estimate was based on the diffusion approximation of the G/G/1 queue, see [34] [35].

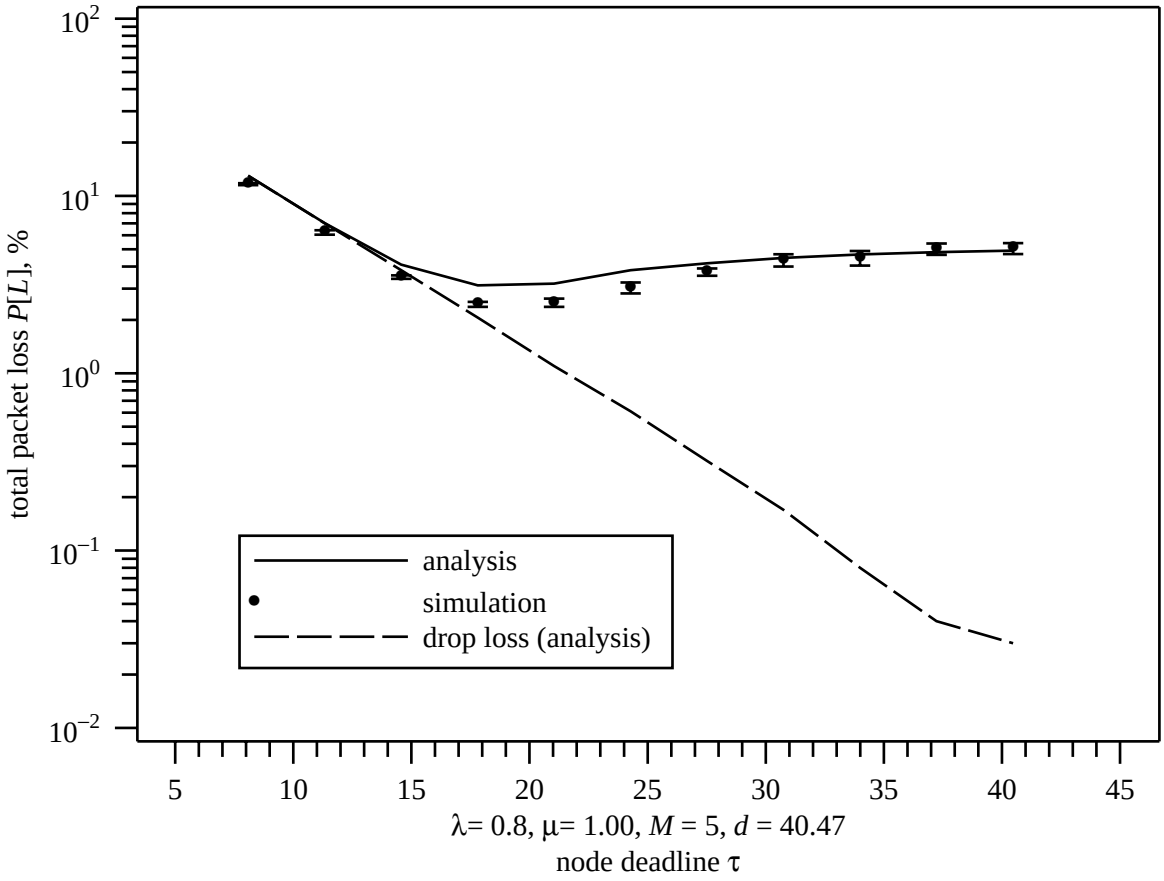

Figure 2.2. Total and drop loss; analysis and simulation

Since the performance varies little in the region from $\tau \approx 17$ to $\tau \approx 21$, a looser deadline in that range is preferable, providing a margin of safety to compensate for errors in the estimation of the system parameters.

The dashed line traces late loss, showing that for very tight deadlines, almost all packets that have not been dropped at intermediate nodes make their deadline.

Optimal Local Deadlines

While the experiments above used the same τ for all nodes, it seems reasonable to assume that the lower traffic impacting downstream nodes could make heterogeneous deadlines advantageous. To test this hypothesis, an M-dimensional simplex method was used to find a set of τ_i's minimizing the overall loss, taking downstream traffic

reductions into account. The optimal homogeneous τ was used as a starting point to minimize the probability of trapping the optimization in a local optimum. From the examples reported in [30], it can be concluded that even for high loads ($\rho = 0.9$) and short VCs such as $M = 2$, heterogeneous deadline yield less than 1% improvement over using a homogeneous τ.

Since the expressions for homogeneous traffic and deadlines are algebraically much simpler and numerically more stable, we examined the performance penalty incurred by ignoring downstream traffic reductions. Clearly, for homogeneous traffic only homogeneous deadlines can be optimal. As the examples in Fig. 2.3 and 2.4 demonstrate, the penalty increases with higher loads, higher uncontrolled losses and longer virtual circuits. (In the figures, the horizontal lines denote the asymptotic losses for the uncontrolled system.)

Range of Effectiveness

In the course of investigation it became clear that the proposed queue control is advantageous only for a range of parameter values. The contour plot of Fig. 2.5, containing lines of equal ratio of controlled to uncontrolled loss vs. length of VC, M, and traffic intensity ρ for uncontrolled losses of 10^{-5}, 10^{-3}, 10^{-2} and $5{\cdot}10^{-2}$, shows that the ratio decreases (performance gains realized by queue control increase) with increasing loads, shorter virtual circuits, higher uncontrolled losses and higher loads. For example, at an uncontrolled loss of 5%, overall losses can drop to as little as 40% of the uncontrolled losses for a three-node circuit operating at a load of $\rho = 0.9$. On the other hand, at uncontrolled losses of less than 10^{-3}, even loads of $\rho = 0.9$ and the minimal circuit length of 2 does not push the controlled loss much below 75% of the uncontrolled loss. The parameter region of worthwhile gains agrees well with the operating region of overload control for packet voice systems. However, for the uncontrolled losses found in video applications, little gain can be expected. It should be noted that the results shown in the graphs are conservative estimates

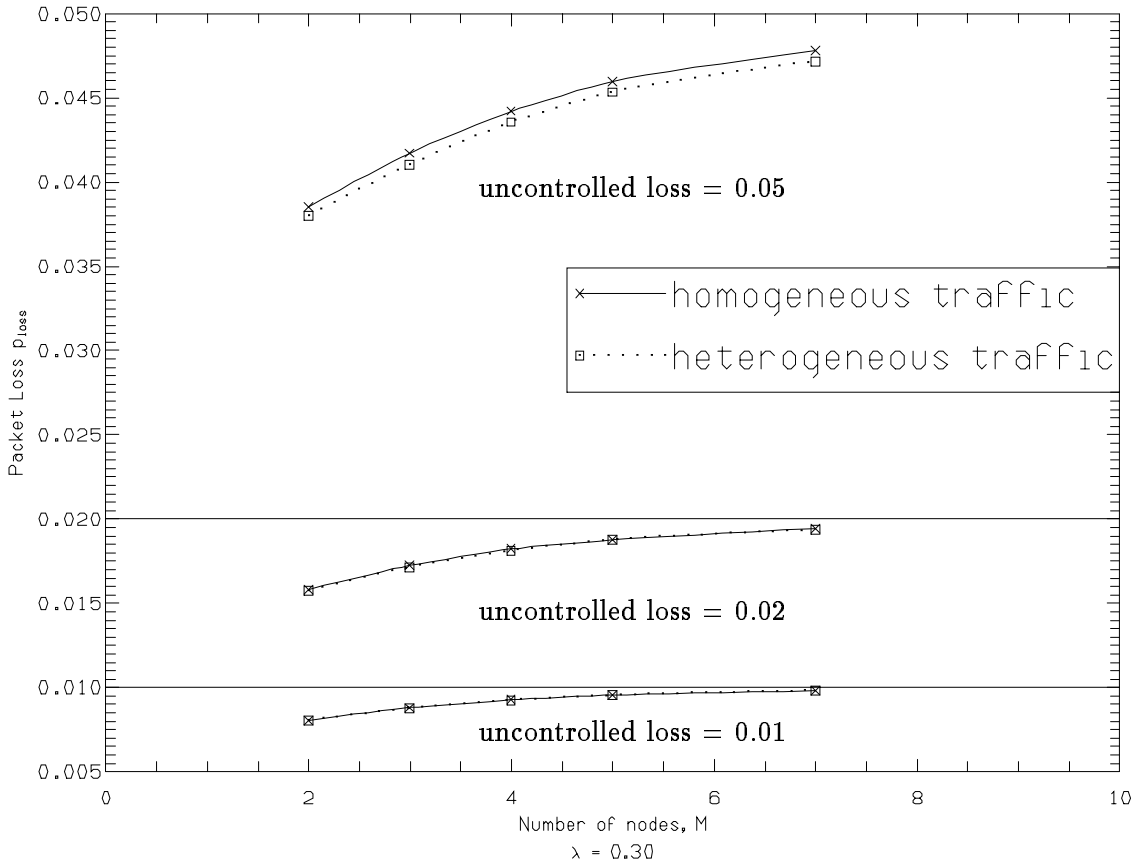

Figure 2.3. Total packet loss vs. number of nodes, for optimal homogeneous deadlines based on homogeneous or decreasing traffic, $\lambda = 0.30$

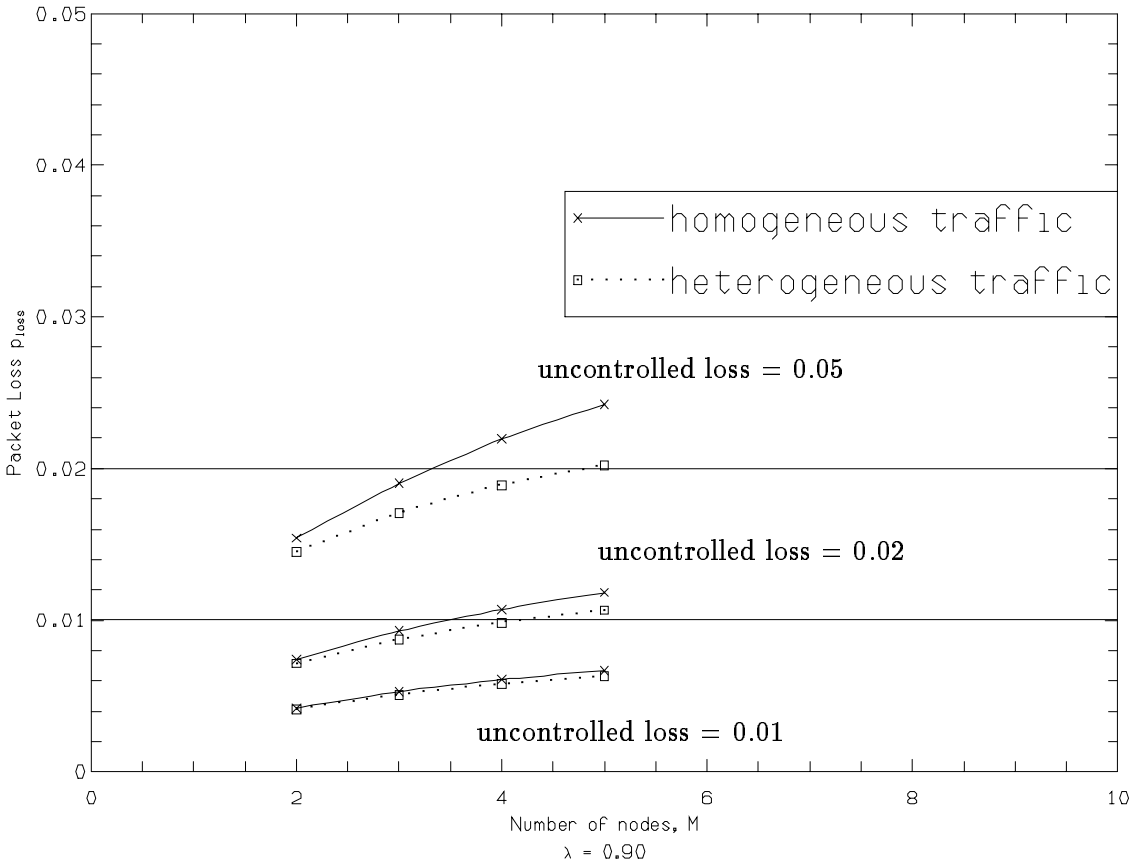

Figure 2.4. Total packet loss vs. number of nodes, for optimal homogeneous deadlines based on homogeneous or decreasing traffic, $\lambda = 0.90$

since they do not take into account the global traffic reduction afforded by having all virtual circuits apply the control policy, especially significant under loads of 0.8 and above. Also, the simulation results discussed earlier show that actual performance may be slightly better than the analysis predicts.

Investigations showed that the optimal local deadlines are relatively insensitive to the number of nodes in the network and, to a lesser extent, the VC load. Also, no obvious relationship seems to exists between drop loss and late loss at the optimal operating point.

2.6 Congestion Control and Robust Local Deadlines

In practical networks, it may not be feasible to measure traffic accurately and adjust the local deadlines accordingly, even with a table-lookup mechanism instead of on-line optimization. However, a static control scenario such as the following would be possible. At call setup, the end-to-end deadline (and, therefore, the playout buffer delay) is set so that the desired loss under non-overload conditions can be met. The local deadlines are then established by table look-up based on the length of the virtual circuit and a tradeoff between overload protection and additional loss incurred under lighter loads. For each packet, the queue controller determines the applicable τ, measured in transmission units, from a look-up table and compares it to the current number of transmission units queued, including those of the packet in transmission. (Here we assume a fixed-rate channel.) These operations are readily implementable in hardware at packet rate.

An example illustrates the approach. Suppose that a 5-node connection can tolerate a loss of 1% without noticeable grade-of-service degradation. A call is allowed into the network only if the average load does not exceed $\rho = 0.8$ at the time of call setup. Correspondingly, the playout buffer is set up for an end-to-end deadline $d = 52.7$. The effect of several choices for τ is shown in Fig. 2.6. We now describe

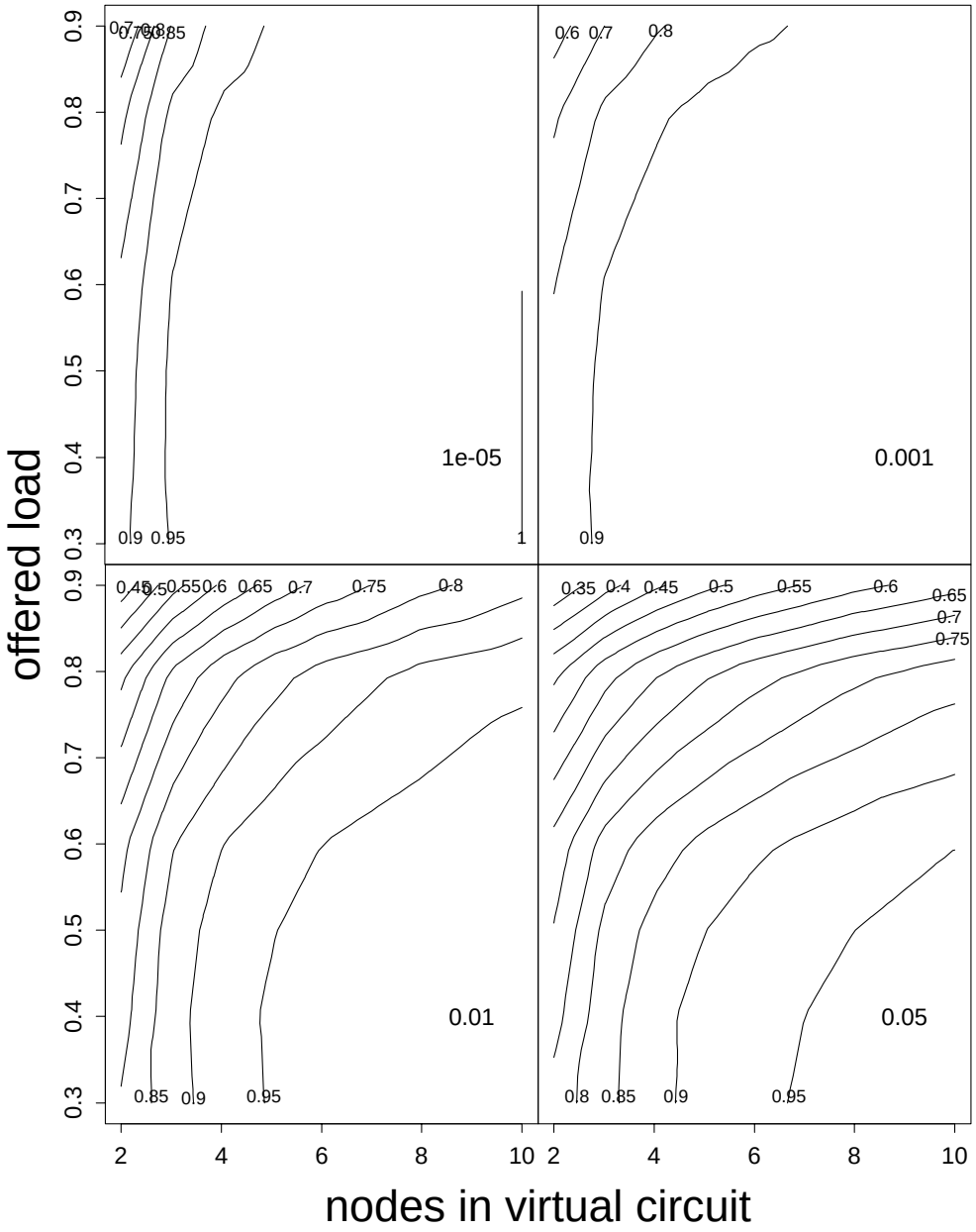

Figure 2.5. Best achievable ratio of controlled (FIFO-BW) to uncontrolled (M/M/1) loss for homogeneous traffic and deadlines; uncontrolled losses of 10^{-5}, 0.001, 0.01 and 0.05

the effect of several different choice of τ. In our first experiment, we pick a value of τ optimized for moderate overload, namely $\rho = 0.9$, yielding $\tau = 18.3$. Assume that during the call the network load rises to that value of 0.9. If no queue control were applied, the losses would reach an intolerable 32%. With the queue control, $\tau = 18.3$, losses are reduced to 7%, which an interpolation method might be able to cope with. However, this deadline is unduly restrictive for lower loads. For example, at $\rho = 0.8$, losses are twice that of applying no control. Tightening the deadline to $\tau = 15$ does not improve overload performance, but leads to further deterioration of the grade of service at $\rho = 0.8$, pushing losses to 3.5%. A more conservative choice, $\tau = 23.3$, ensures that losses at design load and below never exceed 1%, while still limiting overload losses to 9.2%. If load should rise momentarily to $\rho = 0.95$, the overload mechanism will cut losses from an uncontrolled 80% to 20%. As mentioned at the end of section 2.5, actual network performance will most likely be better as all VCs can be assumed to apply similar control mechanisms.

Figure 2.7 compares the goodput, defined as $\lambda(1-P[L])$, of the uncontrolled and controlled system. Beyond a certain load, $\rho = 0.83$ in the example, the goodput for the uncontrolled system actually decreases with increasing load. Optimal random discarding as proposed in [5] can hold the goodput at the peak value even under overload, as indicated by the horizontal line in the figure. It throttles the input traffic to the point of optimal goodput by randomly discarding packets at the source. Optimal random discarding requires, however, on-line traffic or gradient estimation $(\partial P[D]/\partial\lambda)$ to determine the fraction of packets to be discarded. Note also that if we allow the deadline to be adjusted on-line according to the current traffic, FIFO-BW offers a higher goodput than optimal random discarding.

2.7 Summary and Future Work

In this chapter, we have presented an analytic model for evaluating the loss performance of two queue control schemes, based on rejecting packets at intermediate

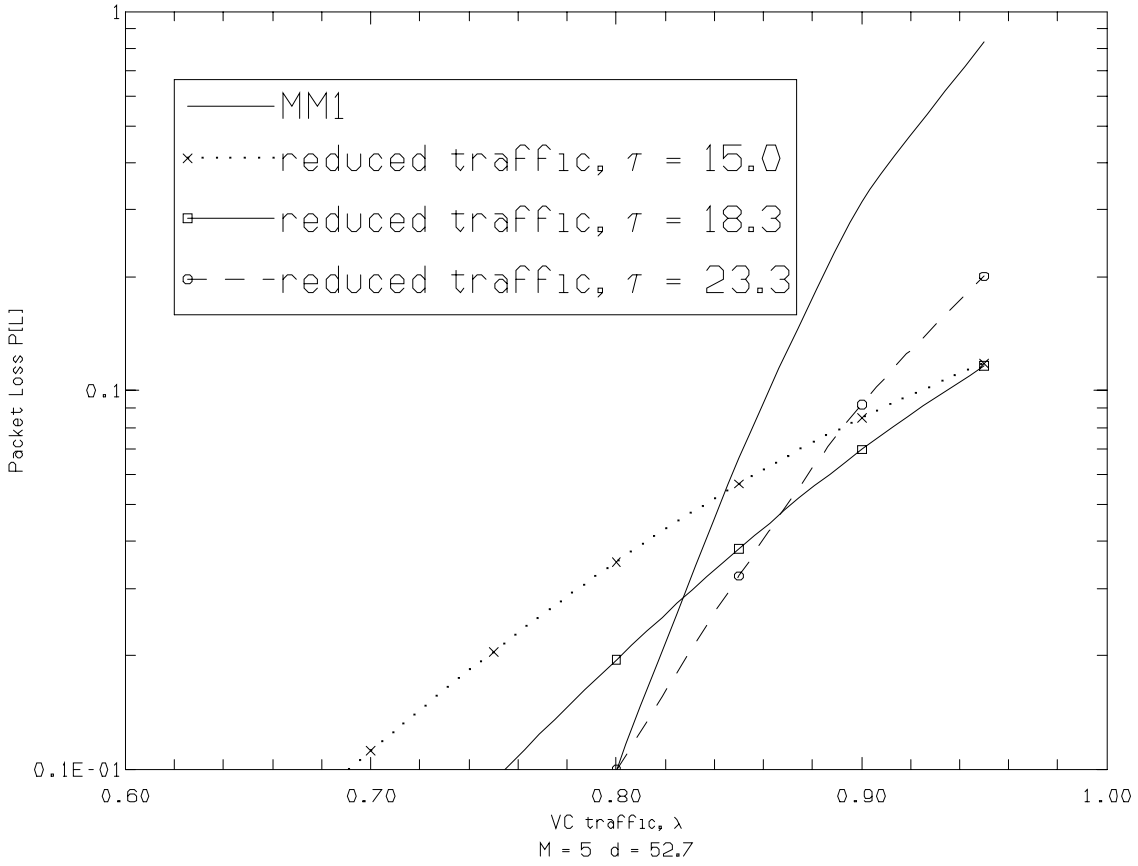

Figure 2.6. Comparison of overload performance of FIFO-BW to that of uncontrolled system; overload region

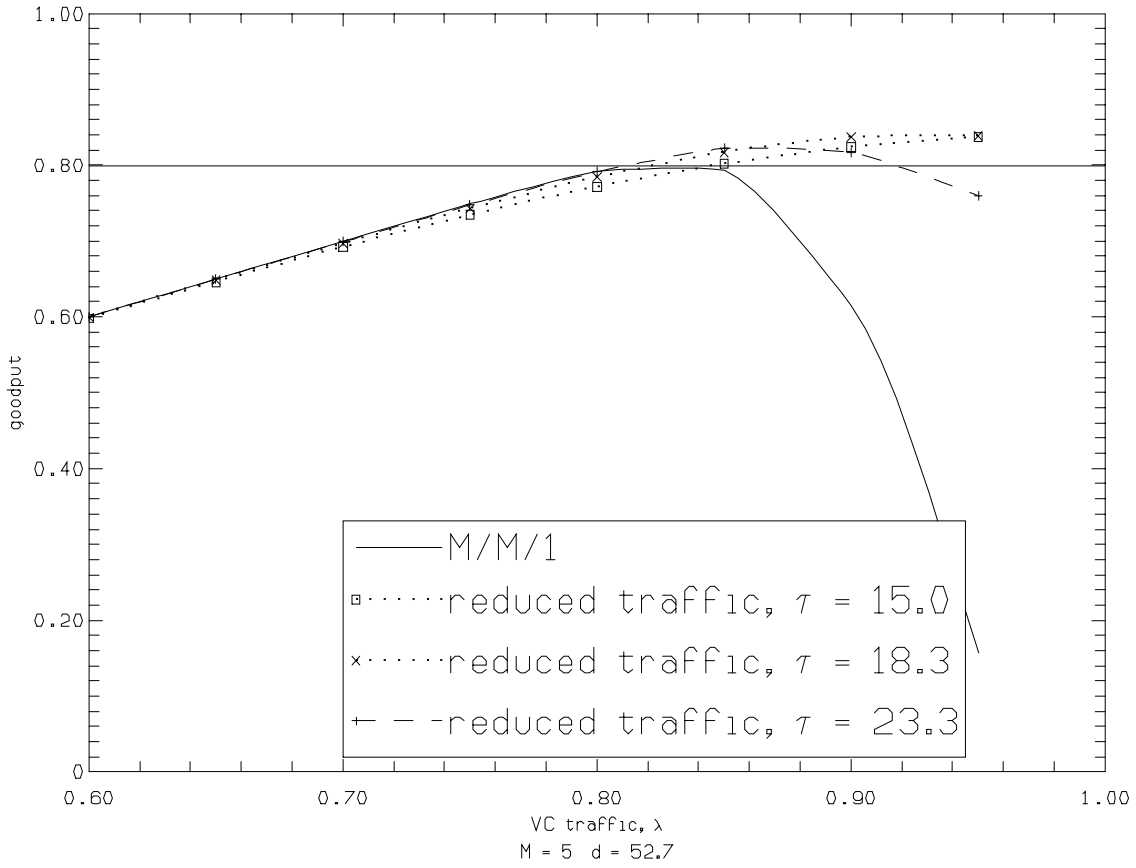

Figure 2.7. Comparison of goodput: tandem-M/M/1 vs. FIFO-BW with various local deadlines, $M = 5$, $\mu = 1$

nodes in a virtual circuit. It was shown that the fraction of lost packets could be reduced substantially for suitable traffic types and virtual circuit size. Even if the exact load is not known, a static overload control mechanism can ameliorate the effects of temporary overload situations. In related work [30], it is shown that analysis technique and performance gains carry over to the case of fixed packet sizes.

It is anticipated that history-based discarding, which admits a packet only if the time spent waiting upstream plus the virtual wait falls below a given threshold, offers improved performance due to better information, albeit with higher processing overhead.

In the next chapter, we will see how selective discarding can be complemented effectively by laxity-based scheduling.

Appendix

In this appendix, closed-form expressions for the waiting-time density and distribution function for the FIFO-BW system will be presented. Homogeneous and heterogeneous traffic leads to Mth order and first-order poles in the Laplace transform expression, respectively, and thus requires separate treatment. A special case for homogeneous traffic simply reduces the number of terms in the expansion. Results for the first three cases enumerated in section 2.5 will be presented; the fourth case of arbitrary τ_i and ν_i is conceptually similar, but incurs another order-of-magnitude increase in complexity not compensated for by increased insight.

For the first case of homogeneous traffic and deadlines, the closed-form expression for the density can be derived from the Laplace transform as

$$\mathcal{L}^{-1}\left[\frac{1}{\alpha}\frac{\nu\rho}{s+\nu}(1-e^{-(s+\nu)\tau})+\frac{1-\rho}{\alpha}\right]^M$$
$$=\mathcal{L}^{-1}\left\{\frac{1}{\alpha^M}\sum_{k+l+m=M}\binom{M}{k,l,m}e^{-l\nu\tau}(\nu\rho)^{k+l}\right.$$

$$\cdot(1-\rho)^m(-1)^l\frac{e^{-l\tau s}}{(s+\nu)^{k+l}}\Bigg\}$$

For $k+l>0$, the fraction containing terms in s can be readily inverted as

$$\frac{(t-l\tau)^{k+l-1}}{(k+l-1)!}e^{-\nu(t-l\tau)}u(t-l\tau)$$

For $k+l=0$, that is, $k=l=0$, the summand reduces to $(1-\rho)^m$. The cdf can be computed by integrating terms over the interval from $l\tau$ to $d-l\tau$.

The second case allows for local deadlines that depend on the node index. The Laplace transform can be rewritten as

$$\begin{aligned}\prod_{i=1}^{M}\frac{1}{\alpha^i}&\left[\frac{\nu\rho}{s+\nu}(1-e^{-(s+\nu)\tau})+(1-\rho)\right]\\&=\left[\prod_{i=1}^{M}\frac{1}{\alpha_i}\right]\sum_{k=0}^{M}\sum_{\vec{l}:|\vec{l}|=M-k}\left[\frac{\nu\rho}{s+\nu}+(1-\rho)\right]^k(-1)^{M-k}\\&\cdot\left[\frac{\nu\rho}{s+\nu}\right]^{M-k}\exp\left(-(s+\nu)\sum_{i=1}^{M}l_i\tau_i\right)\end{aligned}$$

where the binary vector $\vec{l}$ has components l_i, restricted to the values zero or one. $|\vec{l}|$ denotes the number of ones in $\vec{l}$.

Each term in the inner sum can be readily inverted for a given k as

$$(-1)^{M-k}e^{-\nu\tau'}\sum_{j=0}^{k}\binom{k}{j}(1-\rho)^{k-j}(\nu\rho)^{j+M-k}A_{k,j}$$

where

$$A_{k,j}=\frac{(t-\tau')^{M+j-k-1}}{(M+j-k-1)!}e^{-\nu(t-\tau')}u(t-\tau')$$

for $j+M-k>0$ and $A_{k,j}=\delta(t-\tau')$ for $j=k-M$. Here, $\tau'=\sum_i l_i\tau_i$. The number of terms depends on the value of τ and t and is less than $2^M(1+M/2)$.

The third case allows for traffic that depends on the node index; the condition of strict heterogeneity in λ_i insures single poles. The PDF is expressed in the time-domain as

$$w_M(t)=\sum_{k=1}^{2^M-1}\left[\prod_{i=1}^{M}\frac{1}{\alpha_i}\right]\left[\prod_{j=1}^{M}r_j^{1-b(k)_j}(\rho_j\nu_j)^{b(k)_j}\right]\cdot$$

$$\sum_{j \in b(k)} \frac{\sum_{m=0}^{2^{|b(k)|}-1} e^{-\beta_m - \nu_j(t-\tau'_m)} u(t-\tau'_m)}{\prod_{\substack{i \in b(k) \\ i \neq j}} \nu_i - \nu_j} + \prod_{n=1}^{M} r_n \delta(t).$$

$b(k)_i \in \{0,1\}$ stands for the ith digit of k written as a base-2 (binary) number. For brevity, $i \in b(k)$ denotes all indices i such that $b(k)_i$ is 1. The computation of the quantities β_m and τ'_m is based on the expansion of the product of the $|l_k|$ terms of the form $1 - \exp(-(s+\nu_i)\tau_i)$. Define the mapping function $h(k)$, whose ith component $h(k)_i$ is the index of the ith one bit in $b(k)$. Then,

$$\tau'_m = \sum_{i=1, i \in b(i)}^{|b(k)|} \tau_{h(k)_i}$$

and

$$\beta_m = \sum_{i=1, i \in b(i)}^{|b(k)|} \nu_{h(k)_i} \tau_{h(k)_i}.$$

The distribution is readily computed by replacing the terms in the summation over m by

$$\frac{1}{\nu_j}\left[1 - e^{-\nu_j(t-\tau'_m)}\right] e^{-\beta_m} u(t-\tau'_m)$$

This case requires the evaluation of at most $2M \cdot 3^{M-1}$ terms, with the exact value depending on t. The large number of terms for high M (above five), differing in absolute value by several orders of magnitude, forces use of quadruple precision arithmetic and magnitude-sorted summation to control round-off errors.

CHAPTER 3

SCHEDULING UNDER REAL-TIME CONSTRAINTS: A SIMULATION STUDY

3.1 Introduction

We first motivate the performance metrics that may be used to evaluate scheduling disciplines by providing some background on a class of real-time services with adaptive constraints. We then provide a classification system that categorizes scheduling disciplines of interest to real-time services and highlight some additional guidelines that may be used to choose between them. With this background, section 3.2 summarizes a number of scheduling policies that suggest themselves in a real-time environment.

Section 3.3 picks up from the previous chapter, where we had investigated FIFO queues with discarding in some detail. Here, we turn our attention to a number of other queue control and scheduling mechanisms for the slotted system covered in that chapter, comparing them to the simple discarding mechanism described there. Simulation experiments using a different network with a more typical topology and bursty traffic will provide additional insights into the behavior of a wide range of scheduling policies (Section 3.4. We will describe the theoretical and simulation aspects in this chapter, while deferring implementation aspects to Chapter 4. At that point, we can then also draw some conclusions as to the suitability of the policies proposed. The chapter concludes in Section 3.5 by providing detailed statistical data on the audio source used. We point out through the example that the commonly assumed statistical descriptions of audio sources may not be accurate, either in isolation or in aggregation.

3.1.1 Performance Metrics

Two basic measurements suggest themselves in evaluating the performance of scheduling disciplines. First, we can specify a fixed deadline and determine the fraction of packets that miss the deadline. This approach models most closely the demands of "rigid" applications, i.e., those that cannot adapt their deadline to changing network conditions and where the packet has to arrive absolutely, positively on time. A control system would probably require fixed deadlines for both measurement and control signals in order to guarantee stability and allow accurate design of the system transfer function in advance. Applications in manufacturing systems also fall into this category.

Beyond this "classical" performance objective, there is a large class of real-time services that have slightly different characteristics. Before discussing the second metric appropriate to their needs, some background information appears called for.

In past chapters, we have categorized real-time applications according to their loss tolerance, emphasizing that many applications can work quite well with a rather substantial amount of packet loss. Although not required for the algorithms, the implication was that the deadline itself was fixed by properties of the application. In discussing scheduling policies for real-time system, however, it is no longer appropriate to consider deadlines static for all applications. Many emerging applications feature deadlines that stay constant over the short term, but can be adjusted over longer time periods. Examples of such applications include the well-known interactive audio, video and other continuous media applications as well as distributed simulation. All these applications have in common that they attempt to replicate the *relative* timing of elements as seen by the sender at the receiver fairly strictly[1]; however, the *absolute* time difference of the sender and receiver sequences is far less critical. All such reconstruction mechanisms are based

[1]for audio, to within a sample, i.e., several tens to a hundred microseconds

on providing the illusion of a channel with a fixed delay, fixed at least over a short time span. The receiver must be provided with an indication by the sender at what time the information was originally transmitted; the receiver then plays out the information, that is, passes it *in order* to the next higher layer, at the origination time plus some fixed delay. Choosing the fixed playout delay for a channel with *variable* and unknown delays is a compromise between two competing objectives: First, it is usually desirable to limit the playout delay, because the application is interactive (as for voice and video) and/or because storing data that has arrived but needs to be delayed is expensive. However, if the delay is too short, data may arrive too late and thus miss its playout date. Thus, the playout delay needs to be chosen conservatively so as to limit information loss to acceptable values.

Given these conflicting objectives, it appears natural to try to choose the smallest possible playout delay that satisfies the loss criterion. Thus, we need two mechanisms: an estimator that predicts future delay distributions so that the playout delay can be chosen appropriately and a mechanism that allows the time lines of sender and receiver to shift slightly without violating the tight short-term synchronization constraints. The first aspect is still an open problem; one simple approach will be discussed in connection with NEVOT, the network voice terminal, in Section 4.4.

Fortunately, the second mechanism is naturally accommodated for some media: For these media, there are distinct blocks of signals that have very tight internal constraints on reconstructing timing, but the spacing between these blocks itself is more flexible. Voice and music have talkspurts and pause-separated phrases, while video has frames[2]. An adaptive application then adjusts the spacing between these blocks to increase or decrease the playout delay. As long as the adaptation is slight and does not affect synchronization with other media, the change will remain unnoticed. (See also p. 82 for a discussion on playout as applied to voice

[2]life has weekends

transmission over variable-delay channels.) Finally, note that adaptation is required even for fixed-delay channels as long as sender and receiver have clocks which are not in phase lock. The same adjustment mechanism will compensate for clock drift, without special efforts.

See [36] for a more detailed discussion of the difference between rigid and adaptive applications.

After this digression, we return to the point of justifying the second performance metric, namely delay percentiles. A delay percentile provides the optimal playout delay that an service could choose, given a perfect estimator. In other words, a service desiring not to loose more than 0.1% of packets due to missed playout would want to set its playout delay to the 99.9% percentile of the end-to-end delay.

It should be noted that the current metric does not take the actual delay adaptation mechanism into account. A true end-to-end evaluation would also simulate the estimator and use some statistical measure of both loss and playout delay as the overall figure of merit. This metric would then capture the dynamic characteristics of the queueing policy. For adaptive applications, a queueing policy where packet delays would be strongly correlated would probably perform better than one where delays are completely random, even though they both yield the same delay percentile. Unfortunately, the overall performance would strongly depend on the time constants of the estimator. The author is planning work in this area.

3.1.2 A Taxonomy of Real-Time Scheduling Policies

Before describing the details of the policies and their performance, it may be helpful to provide a framework into which to place the scheduling policies. We propose to distinguish four classes of service that can be provided by a network. Note that these are not exhaustive of the desirable types of guarantee, merely of the types of service for which known policies exist that implement them.

1. **bounded delay jitter:** the network admits only a certain number of flows and can then guarantee that all packets of a flow experience queueing delays between given upper and lower bounds. Barring the degenerate case of all lower bounds being zero, the node has to implement a non-workconserving queueing discipline, where packets are delayed even if the channel is idle. Hierarchical round-robin [37], jitter-earliest-due-date [38, 39] and stop-and-go [40–45] are examples of jitter-bounding policies.

2. **guaranteed throughput:** The network does not guarantee delay bounds, but assures that each flow can obtain its guaranteed throughput regardless of the behavior of other flows. In addition, such policies have to limit the time period over which a flow can claim accumulated bandwidth credit, so as to avoid unduly delaying other streams. Weighted fair queueing (WFQ) [36, 46–48], virtual clock [49] and similar algorithms fall into this category. These scheduling policies are generally work-conserving. As long as the traffic obeys certain restrictions (in particular, that it is shaped by a leaky bucket), the delay can actually be upper-bounded. The bound is tight in the sense that some packets actually experience the delay bound for certain traffic patterns.

3. **best-effort, "need-based":** While making no guarantees about its performance, the network tries to provide the performance best suited for the particular type of traffic carried. No upper bounds have been put forward for this class of scheduling disciplines.

4. **best-effort, "need-blind":** The network makes no commitments about its service characteristics, although some networks, by their physical properties, can guarantee that packets will not be reordered, for example. Also, the queue scheduler does not take the type of service or service requirements of the packet into account. The current Internet clearly falls into this category, as

IP datagrams can take an arbitrary amount of time to reach the destination, if they reach it at all. Resequencing is also surprisingly prevalent[3]. Under certain limitations on the source traffic and its peak rates, even for FIFO scheduling, deterministic and probabilistic delay bounds can be computed [50].

Why would a real-time application request anything but the first class of service, i.e., guaranteed jitter, except for reasons of cost? The first "social" reason is fairly obvious, as a network guaranteeing jitter bounds has to set aside bandwidth based on the application's peak needs to serve the jitter-controlled traffic. Since many real-time applications are bursty on long time scales (for example, silence periods for audio and no-motion periods for conference video), only a small fraction of the network bandwidth can be used, on average, for real-time services. It should be noted, however, that the network may fill the unused bandwidth reserved for first class service with low-priority fourth-class service, as long as the maximum packet size of these fourth class packets is factored into the service guarantee. If the guaranteed traffic class has a peak-to-mean ratio of, say, ten, a value representative of uncontrolled video sources, close to 90% of the overall traffic would have to be data (non-guaranteed) traffic. Smoothing of video traffic [51] may reduce the peak-to-mean ratio to a more tenable two to three.

There is another advantage to need-based service policies, namely that delay can be shifted from one flow to another. These can take two forms: flows traversing short paths with mostly uncongested nodes can assume some of the delay of long-haul flows without noticeably degrading their own perceived quality of service. Thus, need-based scheduling disciplines can help ensure uniform and distance-independent quality of service[4]. Secondly, since adaptive applications cannot make use of low delays available only for an occasional packet, low-delay packets with time to spare

[3]The author has experienced instances where the proportion of reordered packets reached 3%.

[4]This is something we have come to expect, for example, in the telephone network.

might as well be delayed to give better service to other flows more in need. For services with rigid deadlines, the latter rationale becomes even stronger, as packets arriving before the deadline do not improve the quality of service seen by the application.

The adaptability of applications can also be taken as another example of the end-to-end argument [52]. Since all packet-switched networks with statistical multiplexing introduce some amount of delay jitter, an application has to be prepared to compensate for that. Given the low cost of memory, bounding the jitter from below appears to be of secondary importance [53]. However, limiting the delay variability, particularly short term fluctuations, improves the performance of adaptive applications by yielding more accurate playout delay estimates.

Surprisingly, there are also "greedy" reasons why a real-time application may opt for second or third class service. First, it may be able to obtain service when first class bandwidth is not available due to network congestion or network provider policies. But secondly, the guaranteed service provided by the first class service may be worse than that provided by second or third class service. For first class service, the channel may be idle even though a packet is ready to be transmitted. But, on average, third class service may be better than second class service for adaptive applications. We will see, using the example shown in [36], that the 99.9% queueing delay percentiles for a particular second class service, namely WFQ, are significantly higher than for some of the third class services investigated here.

Clearly, all four classes of scheduling policies will have their place in an integrated-services network, with decreasing cost from jitter-controlled to need-blind, best-effort service. However, it should be emphasized that better service, higher network utilization and call acceptance rates may result by not automatically assigning real-time services to the jitter-controlled class, as seems to be the tendency in some quarters. (See also [53] for a more elaborate argument why isochronous services do not require an isochronous network.)

The remainder of this chapter will only be concerned with need-based (third class) scheduling disciplines. The scheduling policies described here are designed for the common case and do not address the issue of policing and fairness between streams. Separate security mechanisms must be in place to keep individual users from flooding the network. Policing can be handled with different degrees of force: It may be desirable to allow an individual (paying) customer to exceed committed resource reservations as long as the network is underutilized, possibly with increased charges for excess data[5]. Then, excess packets may be marked for preferred deletion either by the customer, based on its knowledge of the relative importance of the data, or the network [54]. Finally, a fuse-like mechanism may terminate the connection if severe overload persists, as it may be due to faulty customer premise equipment. Given these different policing methods, it appears that there might be an advantage to keeping scheduling and policing separate, as advocated here. Within the context presented, a marking agent could, for example, transform excess packets from deadline-scheduled to best-effort.

3.1.3 Objectives for Scheduling Policies

Beyond the basic performance expressed as deadline violation probability or percentile, other considerations should enter into evaluating the merits of the candidate policies:

local information only: For general applicability, a scheduling policy should depend only on local queue information, node state and information carried by the packets themselves. Obtaining data from downstream queues will usually not be feasible as the propagation delays make the information inaccurate. Also, the every output queue would have to maintain information for every output queue of the next hop. Worse, it would effectively have to duplicate the routing

[5] This is similar to how some electric utilities charge large customers.

performed by the next node to decide whom to tell about which downstream queue occupancy.

bandwidth overhead: Almost all need-based policies have to carry additional information needed by the service policy, thus increasing network utilization particularly for the short packets that are often found with real-time applications (e.g., packet audio). For some policies, we can trade flow or connection state with per-packet header overhead.

computational cost: The scheduling policies intervene either at enqueueing time or dequeueing time. LIFO and FIFO do not need to inspect other packets within the queue, for a total computational complexity of $O(1)$.. Other algorithms such as earliest-deadline-first, fair queueing or virtual clock maintain a sorted list of packets, into which each newly arriving packet is entered. The relative ordering of packets in the queue is time-invariant. With proper data structures, such as a binary tree, the insertion complexity can be limited to $O(\log N)$, where N is the queue length. Finally, in policies where the urgency of packets changes as a non-linear function of time, the scheduling order can only be determined at dequeue time. At that time, the urgency of each packet in the queue has to be recomputed, with the packet having the highest urgency selected for transmission. Policies of this kind incur computation of $O(N)$ per packet.

In order to avoid idling the communications channel, we cannot generally make the dequeueing decision when the previous packet has completed transmission. However, making the decision just after a packet has entered the server can lead to packet transmissions which are suboptimal as the decision at that point may differ from the one taken at the dequeue instant. In general, the performance penalty should be slight, but may make performance guarantees difficult.

reordering: While deadline-based and FIFO policies maintain packet sequences within the same stream, other policies such as LIFO may reorder packets. The consequences of reordering depend on whether the application can process packets out of order or needs to reorder packets internally. In the latter case, any delay advantages of the early, but out of order packets are lost to the application. Also, care has to be taken that delay measurements that form the base for delay adaptation mechanisms only take into account in-sequence packets. Furthermore, some audio/video transport protocols, for example NVP [55] and PVP [56], have difficulties with reordered packets. It is conceivable that a video application could display packets representing a pixel block within a frame out-of-order, while audio applications clearly cannot do that.

robustness: "First, do no harm". The performance of the scheduling policy should not degrade below FIFO even if the parameters (where applicable) are not set exactly right. If the policy estimates local queue behavior, the effect of noisy estimates needs to be determined, as changing loads and bursty traffic will likely make high-precision estimation difficult.

Since the bandwidth and computational costs depend very much on the low level details of the implementation, we ignore them in the simulation experiments in this chapter, but these concerns will assume center stage in Chapter 4. We will discuss how the scheduling policies fare in general under these considerations in the next section.

Due to the relative complexity of the scheduling mechanisms and since we are interested in end-to-end performance, only simulation results are available. Independently, Kobza and Liu [57] simulated related laxity-based scheduling disciplines for a three-node continuous-time system with deterministic service. In their study, the laxity-based scheduler only operated in the final node of the virtual circuit. Here, we attempt to cover a wide range of parameter values for a more realistic network model.

Also, the combined effect of scheduling and packet discarding will be investigated. A subset of the simulations results also serve to validate the discrete-time results obtained in the previous chapter.

3.2 Scheduling and Discarding Policies

In this section, we extend our focus from simple first-in, first-out service to a number of service disciplines, the last two of which are based on laxity[6]:

first-in, first-out (FIFO): Packets are served in order of arrival.

last-in, first-out (LIFO): The most recent local arrival is served first. This discipline was investigated since [58] indicated that for concave deadline distributions, LIFO yields the highest fraction of on-time packets among non-preemptive, non-deadline dependent scheduling disciplines. Thus, performance better than FIFO might be expected even for deterministic deadlines.

transit priority (TP): Packets that enter the network defer to those passing through (so-called transit traffic). Implemented as two queues, the complexity for enqueueing and dequeueing is $O(1)$. (For related considerations in a non-real-time context, see [59] and [60].)

long-haul first (LHF): Packets are simply scheduled according to the number of remaining hops.

FIFO+: This policy is proposed by Clark, Shenker and Zhang in [36]. Each node estimates the average delay for all packets of a given class passing through it[7]. Each packet carries a priority field that is initialized to zero as the packet

[6]Laxity is the time remaining until the packet's deadline expires, i.e., until its time in the network exceeds the deadline

[7]for example, by using a first-order estimator

enters the network. As a packet departs from a node, the difference between the queueing delay actually experienced at the node and the average queueing delay for all packets of the class is added to the priority field. Thus, a positive value in the priority field indicates that the packet has spent more than the class average in the queues traversed so far. Conversely, a negative value indicates that the packet has received better-than-average service. At any queue, the accumulated delay difference is used as the priority measure in enqueueing the arriving packet. If the system time (including service time) is used instead of the waiting time, reordering of variable-length packets within a stream is possible.

hop laxity (HL): Packets are served smallest-per-hop-laxity-first. More precisely, the laxity measure determining the order of service at a packet's ith hop is given by the ratio of laxity to hops remaining,

$$d_i = \frac{d + M - a}{M - i + 1}$$

where a is the age of the packet, that is, the time between its creation and the current time slot at the node. While waiting, the d_i value of a packet decreases with a rate that depends on the number of nodes left to travel. At an output link, the packet with the lowest value of d_i is always transmitted next. Ties between packets with the same laxity are broken in favor of the earliest arrival. A packet is eligible for consideration one slot after its arrival, reflecting the processing delay. This local time-based priority captures the system time remaining until extinction, scaled by the number of nodes that the packet still has to traverse. Thus, packets tend to be "compensated" by a higher priority for above-average delays suffered at earlier nodes. This policy is identical to the "Budgeted Residual-Life Dependent Discipline" (BURD) in [57].

minimum laxity (ML): The earliest-extinction-first "seniority" discipline is similar to HL, but based solely on laxity, without regard to the number of hops left to travel, i.e., $d_i = d + M - a$. Again, the packet with the lowest value of d_i is transmitted. If all deadlines are the same, the policy of selecting the packet with the largest accumulated queueing delay is equivalent to ML.

Variations on local earliest-deadline-first or minimum-laxity (ML) scheduling are quite common [49] [61] [62] [63] [15], as ML is known to be an optimal policy in many circumstances. Note, however, that our policy is meant to be end-to-end, rather than hop-by-hop.

Some of the qualitative properties of these policies are tabulated in Tab. 3.1. Note that all proposed policies are local in the sense that they depend only on information contained within the packets themselves plus the local queue state. They do not depend on the state of queues further down the virtual circuit, although it could be imagined that the extension of the policies based on laxity and accumulated queueing time could benefit from estimates of future delay. Cognizant of the difficulties due to propagation delays and the considerable overhead particularly for datagram networks, these extensions are not pursued further here, but may prove useful as a sort of bound for achievable performance.

The policies above are scheduling policies. Two *discarding* policies were studied. In *local-wait-based discarding*, a packet is dropped if its waiting time at a given node exceeds the threshold τ_i. For FIFO service, this case corresponds to the analysis presented in the first part of this chapter, with τ_i replacing the system size limit K. In *age-based discarding*, on the other hand, a packet is discarded at the ith hop of its path of length M if its age, that is, the time between entering the system and departing from the ith node in its path, exceeds τ_i. Among the many possible ways to set the τ_i's within the network, we explored τ_i's of the general form

$$\tau_i = \beta \left(\frac{i}{M}\right)^{\alpha} d,$$

Table 3.1. Properties of scheduling disciplines (parenthesized header cost can be avoided by maintaining router state)

	reorders		complexity		header	router	estimation
policy	const. length	var. length	enqueue	dequeue	cost	state	
FIFO	no	no	$O(1)$	$O(1)$	none	none	no
LIFO	yes	yes	$O(1)$	$O(1)$	none	none	no
FIFO+	no	no	$O(\log N)$	$O(1)$	yes	yes	yes
HL	no	yes	$O(1)$	$O(N)$	yes	optional	no
ML	no	no	$O(\log N)$	$O(1)$	yes	none	no
TP	no	no	$O(1)$	$O(1)$	none	none	no
LHF	no	no	$O(\log N)$	$O(1)$	(yes)	optional	no
WFQ	no	no	$O(\log N)$	$O(1)$	none	yes	no

with control parameters α and β. As before, d denotes the end-to-end system time deadline. The expression for τ_i is motivated by trying to distribute the permissible travel time d over M nodes. For $\alpha = 1$, the local deadline is proportional to an equal allotment of travel time among all nodes. For $\alpha < 1$, deadlines are looser for early nodes, reflecting the higher uncertainty about the end-to-end waiting time (i.e., we might expect that a packet early in its travel has better chances of "making up" time along the remainder of the VC).

Note that the service and discarding policy are orthogonal. For FIFO, the discarding decision can be made on entering the system as the waiting time within the node is known at that point; for the other disciplines, a packet that is eventually discarded will occupy buffer space until it is discarded.

A packet is first scheduled according to the service discipline, regardless of whether it will be discarded later on. When the packet does enter service, the discarding policy is applied. If the packet is indeed deemed expendable, the next packet enters service within this same time slot and the process is repeated. Actual implementations might choose to discard packets while waiting to make better use of finite buffer resources.

3.3 Scheduling Policy Performance in a Symmetric Network

In this section, the simulation results for the various combinations of discarding and scheduling policy will be presented and discussed. For results where explicit confidence intervals are shown, 10^6 packets contributed to the loss statistic. For the other experiments, the simulation was terminated when all confidence interval halfwidths were within 20% of the point estimate. The confidence level was set at 90%. The confidence intervals were computed using the spectral method [33]. The first 2000 observations were discarded as transient data in all experiments.

Unless noted otherwise, external arrival occur in Poisson-distributed batches so that the total system load (including arrivals from other nodes) at each node without discarding equals $\lambda = 0.8$.

For the discrete-time simulations, we chose arbitrarily to let arrivals from other nodes (internal arrivals) enter the queue before arrivals from outside the network (external arrivals). Among internal arrivals, those coming from lower-numbered queues acquire the outgoing link first. This ordering should be equivalent to random service since the sequence of node numbers a packet traverses is random. Pilot experiments we performed indicate that the ordering between internal and external arrivals has virtually no effect.

In the analysis of the previous chapter, interfering traffic was accounted for by reducing the service rate. In a discrete-time simulation, interfering traffic must be generated explicitly as a straightforward implementation of a tandem queue with deterministic servers would lead to zero waiting times at all but the first queue (the so-called "pipelining" effect). We introduce interfering traffic in such a way that it contributes to our performance statistics. Specifically, we choose for each packet an arbitrary, but loop-free random path of length M through the network of N nodes for each packet. Since a node can receive packets from any other node, this model

is representative of a highly-interconnected network with homogeneous links and uniform traffic.

Table 3.2. Losses (in percent) for discarding based on local wait, $M = 5$, $\lambda = 0.8$, $d = 20$; net-equilibrium traffic model

τ	FIFO			LIFO	HL	ML
	$N = 5$	$N = 50$	$N = 90$	$N = 50$	$N = 50$	$N = 50$
4	4.66±0.09	9.35±0.11	9.66±0.15	32.2±0.14	33.000±0.267	16.000±0.094
5	2.55±0.08	6.24±0.13	6.48±0.14	28.1±0.15	27.500±0.249	12.700±0.106
6	1.55±0.07	4.24±0.13	4.47±0.13	24.9±0.15	22.900±0.249	10.100±0.105
7	**1.28**±0.09	3.11±0.11	3.28±0.13	22.3±0.17	18.080±0.231	8.110±0.114
8	1.39±0.12	2.58±0.13	2.78±0.15	20.2±0.15	15.200±0.202	6.460±0.094
9	1.56±0.13	**2.47**±0.16	**2.68**±0.18	18.4±0.16	12.000±0.158	5.110±0.110
10	1.73±0.16	2.58±0.15	2.78±0.20	16.8±0.16	9.140±0.110	4.040±0.082
11	1.84±0.17	2.80±0.18	3.02±0.23	15.6±0.14	6.570±0.103	3.140±0.100
12	1.90±0.18	3.02±0.18	3.23±0.27	14.6±0.15	4.410±0.789	2.430±0.079
13	1.94±0.18	3.31±0.23	3.45±0.29	13.7±0.14	2.650±0.064	1.840±0.080
14	1.96±0.18	3.40±0.25	3.61±0.36	13.0±0.12	1.410±0.042	1.380±0.067
15	1.96±0.18	3.51±0.23	3.77±0.36	12.4±0.15	0.605±0.029	1.020±0.061
16	1.97±0.18	3.66±0.23	3.89±0.40	12.0±0.13	0.211±0.026	0.740±0.056
17	1.97±0.18	3.66±0.25	3.98±0.42	11.6±0.12	0.094±0.020	0.532±0.049
18	1.97±0.18	3.76±0.28	4.01±0.39	11.4±0.14	0.059±0.017	0.387±0.047
19	1.97±0.18	3.80±0.29	4.09±0.40	11.2±0.14	**0.047**±0.019	0.305±0.042
20	1.97±0.18	3.78±0.28	4.05±0.43	**11.1**±0.17	0.050±0.023	**0.296**±0.050
21	1.97±0.18	3.79±0.28	4.05±0.45	11.3±0.15	0.069±0.030	0.459±0.087
50	1.97±0.18	3.81±0.28	4.14±0.47	12.7±0.14	0.129±0.061	0.905±0.160

In a first experiment, we showed that the analysis performed in the previous section predicts the simulation results well. For deadlines that are looser than optimal, all analytical results fall within the confidence intervals, while for tighter deadlines the analysis tends to overestimate the errors, but by no more than about 20% of the simulation estimate. The agreement between analysis and simulation improves as N increases. In all cases, both arrive at the same estimate for the optimal local deadline.

Let us now turn to the effect of scheduling and discarding on end-to-end loss, using the results for our running example, tabulated in Table 3.2 and 3.3 for

local-wait-based and age-based discarding, respectively. In the tables, the best performance is shown in bold face. Table 3.3 also contains data for the extreme cases of no discarding (last row) and discarding of expired packets only (first row). We see, for example, that age-based discarding is generally not appropriate for the scheduling disciplines investigated, but that there is significant benefit (loss reductions by a factor of two to four) to be gained by discarding expired packets. Ony LIFO gains relatively little by this approach.

Table 3.3. Losses (in percent) for age-based discarding, $M = 5$, $N = 50$, $\lambda = 0.8$, $d = 20$; † discard expired packets; * no discarding

α	β	FIFO	LIFO	HL	ML
0.00	1.00†	1.99±0.13	**10.5**±0.12	**0.0317**±0.012	**0.21**±0.030
0.20	1.00	**1.92**±0.12	11.0±0.13	0.1140±0.023	1.36±0.074
0.33	1.00	1.98±0.10	11.5±0.15	0.2300±0.035	2.99±0.085
0.40	1.00	2.01±0.11	11.6±0.12	0.2880±0.040	3.83±0.091
0.50	1.00	2.30±0.11	12.1±0.14	0.4380±0.047	5.96±0.101
0.60	1.00	2.66±0.09	12.5±0.13	0.5600±0.049	7.38±0.098
0.66	1.00	3.23±0.10	12.7±0.12	0.6970±0.056	9.15±0.111
0.75	1.00	4.31±0.09	13.2±0.13	0.8480±0.060	11.20±0.097
1.00	1.00	6.26±0.10	14.0±0.13	1.1800±0.066	13.90±0.106
1.00	1.50	3.05±0.09	11.6±0.12	0.6210±0.051	9.05±0.108
1.00	2.00	2.41±0.12	11.4±0.14	0.3780±0.037	5.90±0.097
1.00	2.50	2.77±0.21	11.5±0.13	0.2400±0.032	3.75±0.083
1.00	3.00	3.14±0.19	11.8±0.14	0.1580±0.028	2.28±0.075
1.00	5.00	3.82±0.28	12.5±0.14	0.0744±0.036	0.33±0.062
1.00	50.00*	3.81±0.28	13.1±0.19	0.1290±0.061	0.91±0.160

The lowest packet loss fractions achieved for the various combinations of discarding and scheduling policies for the running example are summarized in Table 3.4. It is clear from this table that scheduling has a far greater effect on the loss than the discarding policy. HL lowers the losses by about two orders of magnitude, from 3.8% for the uncontrolled system with FIFO scheduling to 0.03% for HL scheduling with the optimally "tuned" discarding parameter. For the set of parameters studied, LIFO is by far the worst scheduling policy, with losses in excess of 10%.

On the other hand, it has been shown [58,64] that LIFO maximizes the goodput of a single $G/G/c$ queue among all work-conserving, non-preemptive scheduling disciplines that do not take the service time or the laxity into account, as long as the deadlines have a concave cumulative distribution function (equivalently, a non-decreasing pdf). Thus, HL and ML would not fall into this category as they obviously schedule based on laxity. Also, queues that discard packets would disqualify as they are not work-conserving. Even for non-discarding FIFO and LIFO, the result does not apply in our case since the system investigated has deterministic deadlines. Kallmes *et al.* show [58] that for looser, constant deadlines FIFO may indeed result in lower losses than LIFO.

The mean wait experienced at the ith hop of a packet (omitted here for reasons of space) clearly shows the effect of the scheduling disciplines. This mean wait is naturally identical for each hop in the case of FIFO and LIFO, it increases with i (as the packet approaches its destination) for HL, while it decreases for ML.

The relative benefits of discarding packets, the influence of the discarding policy and the parameter τ_i depend on the scheduling discipline. *Judicious discarding can lower the losses by a factor of between 1.3 for LIFO, to 2 for FIFO and up to factors of 4 and 3 for HL and ML, respectively.* In all cases, moreover, age-based discarding improves upon discarding based on local waiting times. This should be expected since age-based discarding uses more information, reflecting the whole travel history, than local-wait based discarding. The difference between the two discarding policies is least pronounced for LIFO, with a factor of 1.05 improvement, while introducing age-based discarding with the other scheduling disciplines lowers losses by a factor of between 1.3 and 1.5 times compared to local-wait discarding. Since HL and ML scheduling require laxity information for scheduling, age-based discarding seems to recommend itself for these two policies, while for FIFO the choice is more of a tradeoff between performance and simplicity of implementation.

Table 3.4. Packet losses (in percent) for different service and discarding policies; $M = 5$, $N = 50$, $\lambda = 0.8$, $d = 20$

	Service policy			
Dropping	FIFO	LIFO	HL	ML
none	3.81	13.1	0.129	0.905
expired	1.99	10.5	0.032	0.210
age	1.92	10.5	0.032	0.210
local wait	2.47	11.1	0.047	0.296

For age-based discarding, discarding only expired packets seems to be the best policy (at least for the cases studied here), even though marginal improvements are sometimes possible for slightly tighter deadlines. (It is not clear why tighter deadlines work better for $N = 5$.) Discarding expired packets is a no-risk policy as it never discards packets that may make their deadline, requires no adjustment of parameters and does not depend on model assumptions with regard to traffic statistics.

For local-wait based discarding, the optimal deadline for FIFO differs markedly from that for the other scheduling disciplines. For FIFO, a deadline significantly below d (here, around $0.5d$) recommends itself, while for the others, discarding packets that have spend their entire allotment of d waiting at a single node performs best or very close to best.

After this rather detailed investigation of the five-node VC, the question of the performance of the policies studied under a wider range of parameters needs to be addressed. We limit ourselves to discarding expired packets for the reasons noted above as well as to keep the parameter space manageable. We maintain the node load at $\lambda = 0.8$ and the network size at $N = 50$, but vary the end-to-end deadline d to obtain the loss performance under different amounts of packet loss. Also, both the case of geometrically and that of Poisson-distributed external arrivals were

simulated. The results are summarized in Table 3.5. Zero loss values indicate that no loss occurred during a simulation that processed 10^7 packets.

Table 3.5. Packet losses (in percent) with discarding of expired packets; Poisson and geometric arrivals, $N = 50$, $\lambda = 0.8$

M	d	FIFO		LIFO		HL		ML	
		Poisson	geo.	Poisson	geo.	Poisson	geo.	Poisson	geo.
1	7	0.720	3.77	6.72	11.60	0.720	3.77	0.720	3.77
	10	0.200	1.76	4.84	8.83	0.200	1.76	0.200	1.76
	16	0.0150	0.469	2.88	5.78	0.0150	0.469	0.0150	0.469
2	10	1.72	6.46	9.45	14.40	0.501	3.36	0.645	3.52
	15	0.342	2.80	6.17	10.30	0.0515	1.11	0.0768	1.23
	21	0.0396	1.03	4.02	7.39	0.00364	0.287	0.00578	0.334
5	20	2.01	6.63	10.06	14.00	0.0304	1.21	0.208	1.94
	25	0.617	3.79	7.92	11.10	0.00289	0.413	0.0248	0.802
	33	0.0624	1.52	5.92	8.13	0.00000	0.0600	0.00081	0.165
10	33	2.54	7.06	11.20	13.50	0.00106	0.439	0.137	1.53
	40	0.720	3.83	8.39	10.70	0.00080	0.0672	0.00860	0.536
	49	0.110	1.61	5.89	8.05	0.00000	0.00559	0.00051	0.0929

A number of conclusions can be drawn from the this table. For all parameter values simulated, the ranking established above, that is, HL best, followed by ML, FIFO and LIFO, persists. As might be expected, the differences between service disciplines become in general more pronounced as the VC length increases, although the difference between HL and ML actually decreases in going from $M = 5$ to $M = 10$. Thus, contrary to the claims in [57], single-node behavior is not a good predictor for the performance gain in longer VCs. Also, with one exception, for the same VC length, the difference between FIFO and the policies HL and ML increases as the loss rates decrease.

ML, while inferior to HL, still reduces losses by about an order of magnitude compared to FIFO. As discussed in the next section, it may, however, be easier to implement than HL.

Intuition suggests that the statistics of the external arrival process have less and less influence as the VC length increases. For example, Table 3.5 shows that the

ratio of loss between geometric and Poisson arrivals decreases from about 5.2 to roughly 2.8 as the VC length increases from 1 to 10 (for closest deadline and FIFO service). However, for HL, no clear trend is discernible, with ratios of two orders of magnitude even for the longest VC.[8]

3.4 Scheduling Policy Performance for a Tandem System

In the last section, we covered in detail a set of experiments where the network was completely homogeneous. All packets traversed the same number of hops, all queues were equally loaded. The fraction of packets exceeding a given fixed deadline served as the figure of merit. In this section, we widen the scope of scheduling policies investigated and change the network configuration, the figure of merit and the traffic characteristics. The topology, flows and metrics duplicate that presented by Clark *et al.* [36], allowing for ready comparison.[9]. The network configuration is shown in Fig. 3.1. 22 statistically identical flows, described below, traverse the network, all flowing in the same direction. Twelve of these streams are of length one hop, four each of length two and three, and the remaining two streams reach touch all nodes for a length of four hops. Each link is shared by ten of the streams. All flows belong to the same class. Each node has a queue that can hold 200 packets.

While the last section was meant to reflect the characteristics of applications with rigid deadlines, the experiments in this section are geared more towards applications with adaptive deadlines. Thus, instead of loss caused by exceeding a fixed deadline, we measure average delay and the 99.9-percentile of the delay distribution.

[8]The results in Table 3.5 differ slightly from those in Table 3.3 as they are derived from different simulation runs.

[9]Note, however, that the delays reported in the paper are slightly too high, apparently due to a problem with the random number generator used (Lixia Zhang, private communication).

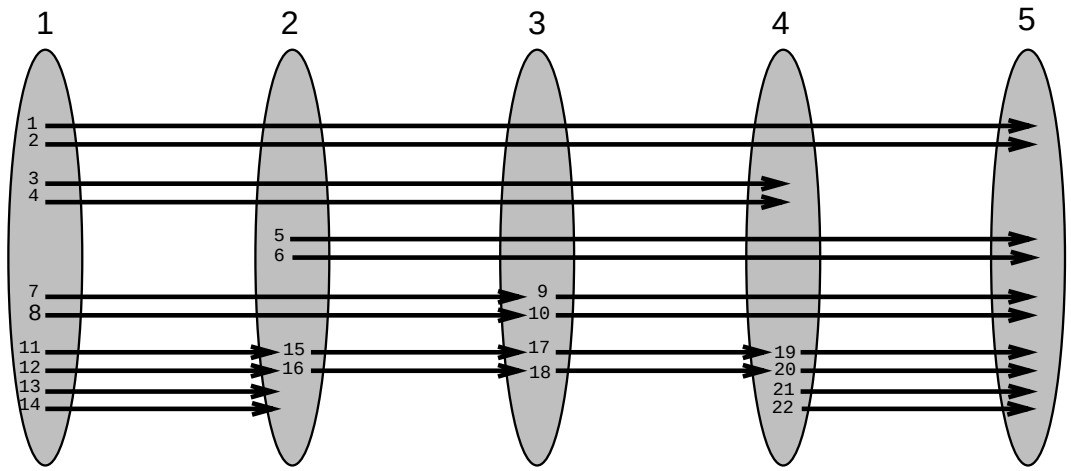

Figure 3.1. The Traffic Streams for Tandem Network

Two sources model the traffic generated by bursty real-time applications. The first source is a two-state Markov process. This source, with the same characteristics as the one in [36], alternates between idle and active periods. The idle period is exponentially distributed with a mean of 29.4 ms, while packets arrive with a fixed rate of 170 packets/second during the active period. The first packet arrives 5.88 ms after the beginning of the active period. The number of packets during an active period is geometrically distributed with a mean of 5 packets. Thus, an average active period last 29.4 ms, yielding an average packet rate of 85 packets/second. Note that an active period may contain zero packets and that the minimum packet separation is 5.88 ms. Each packet has a fixed length of 1000 bits. The source is filtered with a (r, b) token bucket regulator. Here, the average rate r is set to 85 packets/second, the token bucket depth to 50 packets. The regulator discarded roughly 2% of the generated packets, so that the original per-node load of 0.85 was reduced to about 0.83.

The second source is derived from a packet trace recorded during a 52 minute, 15 second transatlantic packet voice conversation across the Internet between Dr. Towsley and a colleague in France held on August 17, 1992. The conversation was recorded using the NEVOT network voice terminal described later (Section 4.4).

Each packet of length 180 bytes (plus headers) represented 22.5 ms of 8-bit μ-law audio sampled at 8,000 Hz. Note that for experimental reasons, the packet length was increased to 300 and 600 bytes, including headers. A total of 1643 talkspurts and silence periods consisting of a total of 50468 audio packets were transmitted, for an average rate of 16.1 packets per second or an audio rate of 23.1 kb/s at the original packet size. The minimum interarrival time was measured at 0.7 ms, the maximum at 19.8 seconds. The average interarrival time was 62.1 ms, with a 99.9 percent below 985 ms. Each silence period had an average duration of 1.23 seconds, while each talkspurt consisted, on average, of 30.69 packets. To ensure approximate statistical independence of flows, the traces were read with different initial offsets, i.e., one flow would read the trace starting at the first packet, the next at packet 2000, and so on.

The last section in this chapter contains some observations on modeling voice sources, pointing out the potential inadequacy of currently accepted voice models.

Table 3.6. Results for tandem network of Fig. 3.1; for Markovian (M) and voice (V) sources; results from [36] marked with "M (Clark)"

		path length							
		1		2		3		4	
source	policy	mean	99.9%	mean	99.9%	mean	99.9%	mean	99.9%
M (Clark)	WFQ	2.65	45.31	4.74	60.31	7.51	65.86	9.65	80.59
	FIFO	2.54	30.49	4.73	41.22	7.97	52.36	10.33	58.13
	FIFO+	2.71	33.59	4.69	38.15	7.76	43.30	10.11	45.25
M	FIFO	2.60	27.65	4.99	36.15	7.28	45.25	9.84	54.45
	FIFO+	3.23	35.55	4.98	36.35	6.44	36.05	7.76	35.65
	HL 30	4.87	30.85	4.94	32.75	4.99	35.25	4.03	39.35
	HL 50	5.29	39.85	4.93	38.45	4.83	35.75	3.35	30.75
	HL 100	5.50	59.25	4.92	56.06	4.82	53.15	3.10	27.25
	TP	4.47	59.85	4.54	59.35	5.05	57.55	4.26	31.15
	LHF	5.18	69.95	5.20	68.55	5.59	67.25	4.31	33.35
	ML	4.14	58.45	4.88	101.8	5.43	119.4	5.21	128.0
	LIFO	2.74	202.9	5.07	254.2	7.37	287.3	9.83	307.2
V	FIFO	10.04	344.7	19.06	340.3	20.18	359.2	32.18	407.6
	HL 30	10.80	344.7	19.06	340.3	19.27	472.2	34.28	622.6
	HL 50	11.35	273.4	17.85	315.3	18.60	471.2	31.78	614.4
	HL 300	15.25	322.5	17.97	327.0	14.62	372.4	16.25	392.8

For this set of simulation experiments, the network link speed was assumed to be 1 Mb/s. Together with packet sizes of 125 bytes, the per-packet transmission time computes to 1 ms.

Discrete-event simulations covering 62 minutes of simulated time were run. The first two minutes were discarded to reduce the effect of the initially empty system on the performance measured. For the FIFO+ policy, a first-order recursive filter with a weighing constant of 0.001 estimated the mean queueing delay. (The weighing constant was determined during pilot experiments; sensitivity experiments still need to be completed.) For FIFO+, we could only achieve reasonable results coming close to the values published [36] by combining it with transit priority.[10]

The queueing delays, i.e., *not* including transmission times or propagation delays, of our simulations are displayed in Table 3.6. For HL, the laxity parameter chosen is shown in parentheses. All delay values are shown in milliseconds. The traffic source is indicated by the letters 'M' and 'V', representing the Markovian two-state source and the conversational voice source, respectively. The first three rows show the values reported in the paper by Clark *et al.* [36]. The parameter following the HL designation denotes the end-to-end deadline measured in milliseconds.

For easier comparison, the 99.9% percentiles for the delay are also graphed in Fig. 3.2 and Fig. 3.3, using the data for the two-state source. The first graph shows the scheduling policies with generally lower delay percentiles than FIFO, while the second graph shows those with higher delays.

First, we note that, as expected for work-conserving disciplines, the average waiting times are roughly equal, albeit distributed differently between long and short paths for the different policies. Of primary concern, however, is the delay

[10]It is conjectured that that may have to do with the fact that, at least for M/M/1, for all loads a majority of packets are going to experience delays less than the average delay and are thus going to be disadvantaged with respect to new arrivals. For M/M/1, the fraction of packets experiencing no more than average delay is always greater than $1 - 1/e = 0.6321$.

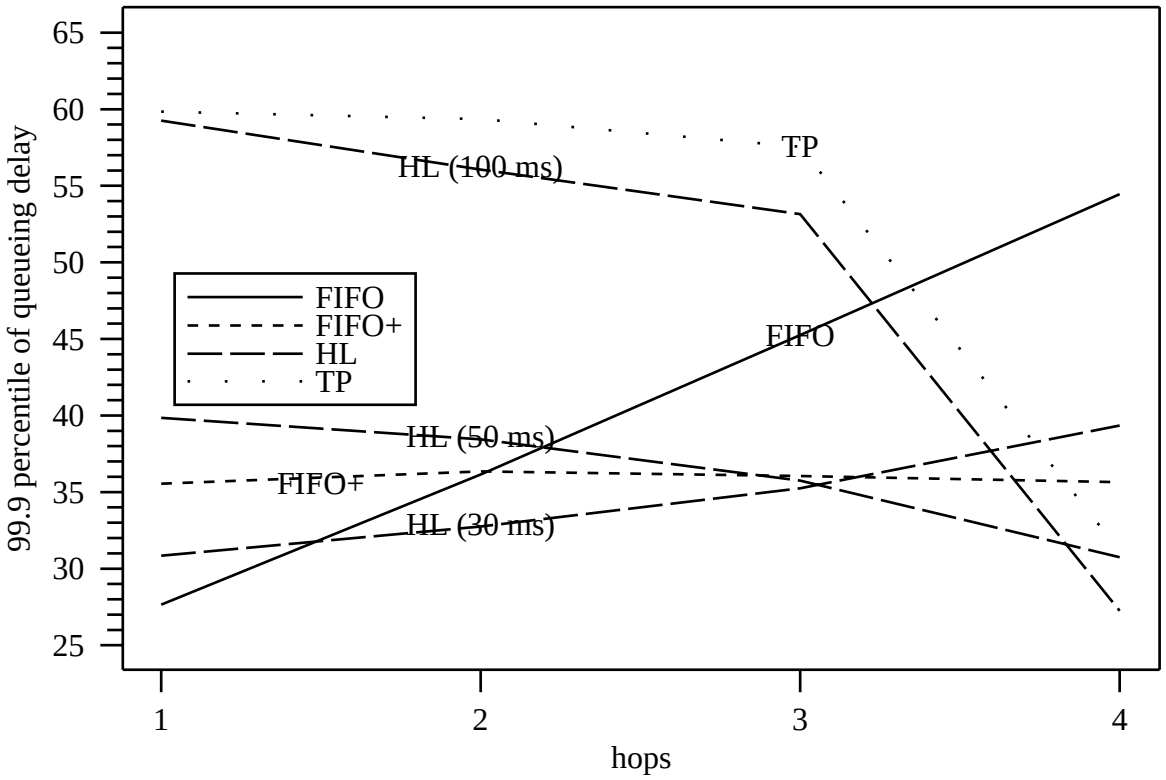

Figure 3.2. 99.9-percentile values for the queueing delays (low-delay policies)

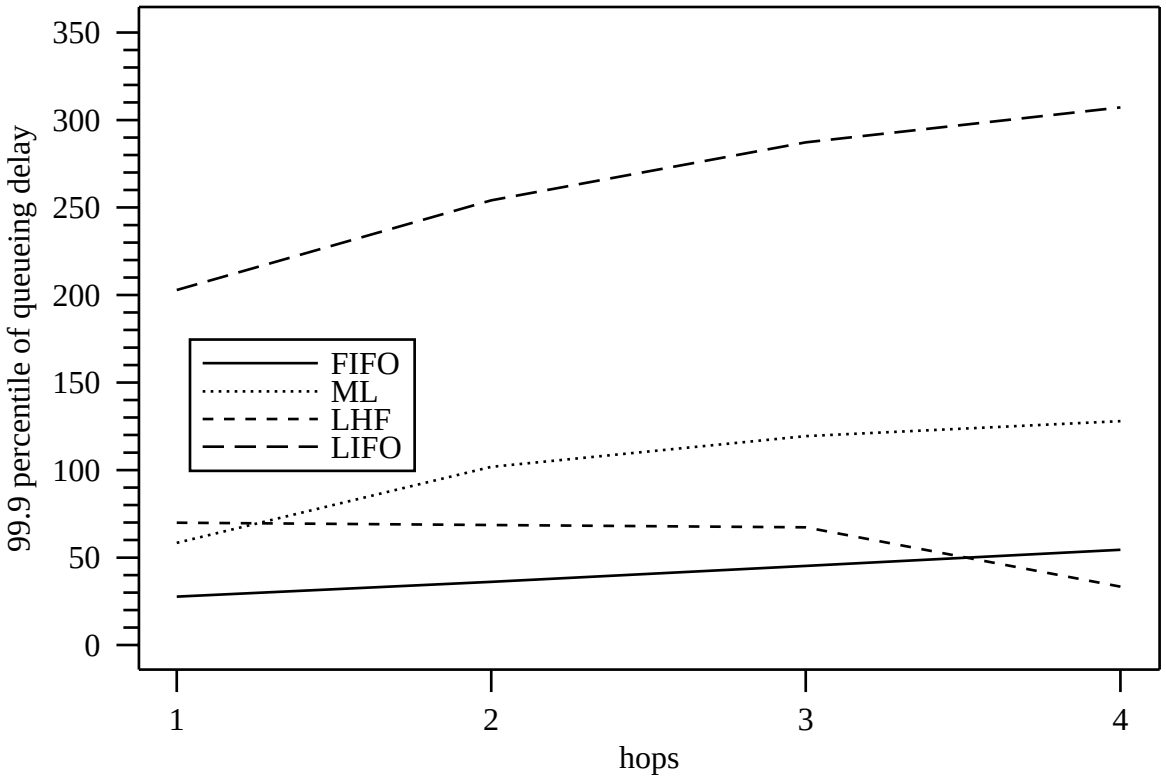

Figure 3.3. 99.9-percentile values for the queueing delays (high-delay policies)

percentile. Immediately, LIFO, LHF and, surprisingly, ML can be ruled out from further consideration. Note that the delays for LIFO do not include any necessary reordering delays.

The simple transit-priority (TP) policy shifts too much of the delay from the long-haul paths to the short-haul paths. Note that this effect is probably partially explained by the particular topology, as the 4-hop traffic in this example never has to yield to other traffic, as at its entry point at node 1, all other traffic is also entering the network. The parametrized family of hop-laxity (HL) policies can be tuned to keep the delay almost constant or even decreasing, independent of the path length. Queueing delays decreasing with hop count may be desirable to compensate for longer propagation delays. For long paths, the queueing delay is almost cut in half. Unfortunately, the delay performance of HL as a function of path length seems to strongly depend on the value of the laxity parameter.

With the caveat about the priority of packets entering the network noted earlier, FIFO+ seems to perform reasonably well without as much concern about choosing an appropriate laxity parameter.

FIFO+ and HL require roughly the same amount of computation, with a slight advantage for FIFO+ as it only requires one comparison per packet while searching through the queue for the next candidate. Also, it can perform its computation when enqueueing packets, while that reduces performance for HL.

From the data, it is apparent that limiting delays appears to be inherently difficult, in particular since we can only try to compensate for past delays by increased priorities, without knowledge about queue states seen at future nodes.

Thus, pending further investigation, we can tentatively state that for applications with fixed deadlines, where choosing the laxity value is not an issue, the HL policy appears promising, while for the case where delay percentiles are used as the performance metric, FIFO+ suggests itself. FIFO+ has the property that delays are balanced rather than depending on a parameter choice of the end user. Thus, if

loading increases, all flows in a class share in the increased delay, while for HL, the tendency to assign one's own traffic lower laxity values would have to be balanced by appropriate admission or charging policies.

HL offers a straightforward way to offer different classes of service to different real-time streams according to their urgency; this can only be indirectly achieved through a strict scheduling priority mechanism in FIFO+.

From the above discussion, a number of areas for future research seem indicated. For FIFO+ and the other policies, the performance under the uniform conditions discussed in the last section needs to evaluated. Also, the sensitivity to load changes and the estimation parameter requires further study. For the HL policy, a heuristic for choosing the laxity parameter is called for. For all policies, the performance in networks with bottlenecks, probably the more typical scenario[11], needs careful study. It is expected that the policies that try to compensate for queueing delays suffered at past nodes by increased priority down stream will not do as well, since there is less chance for compensation once the bottleneck has been cleared.

3.5 Notes on Modeling Voice Sources

In this section, we closely analyze the conversational voice source introduced on page 50, as it allows us (by negative example) to gauge the effectiveness of the two-state voice source model commonly found in queueing analyses.

The traditional voice model of roughly exponentially distributed talkspurt and silence periods dates back to the mid-sixties [65–67].

Fixed-rate voice codecs, which generate audio packets at fixed intervals during talkspurts, are the dominant form of audio codecs in use, even though a number of researchers have proposed variable-rate codecs suitable for packet voice networks

[11]The introduction of the 45 Mb/s backbone in the Internet, for example, typically introduces two transitions from high-speed local networks to the slow speed access into the backbone and vice versa.

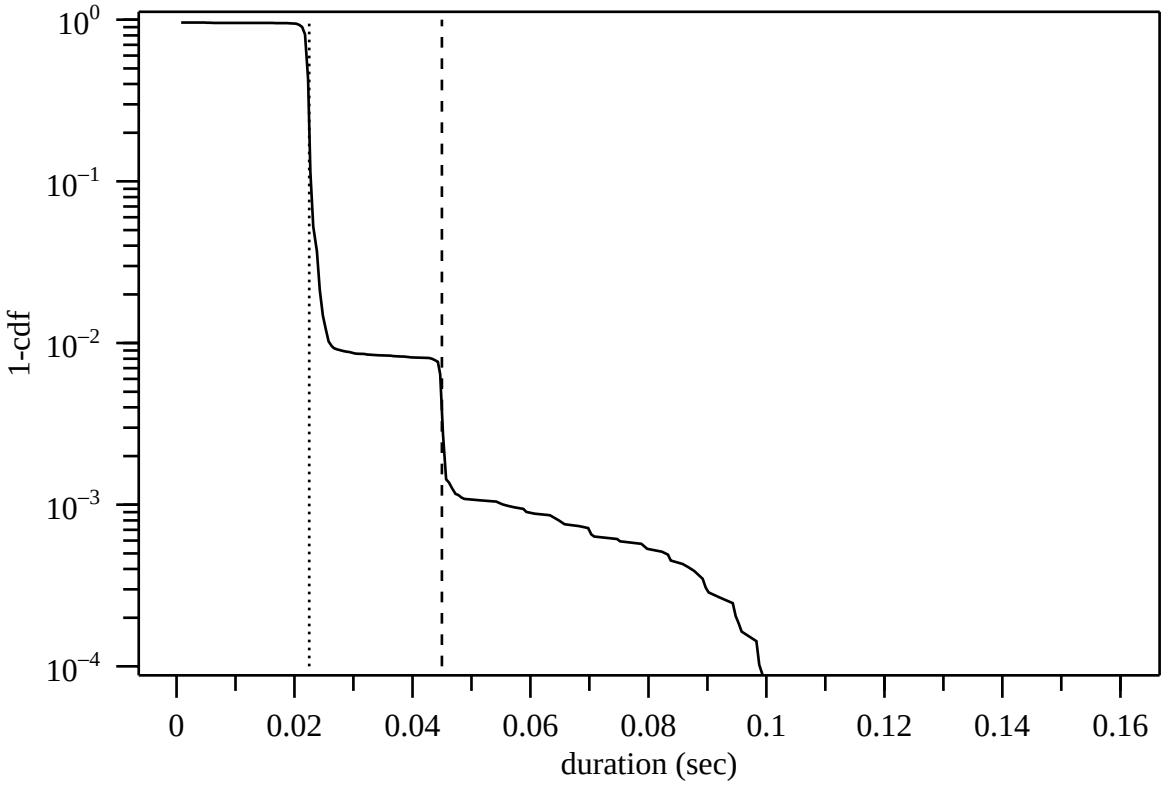

Figure 3.4. The interarrival time distribution for packet audio

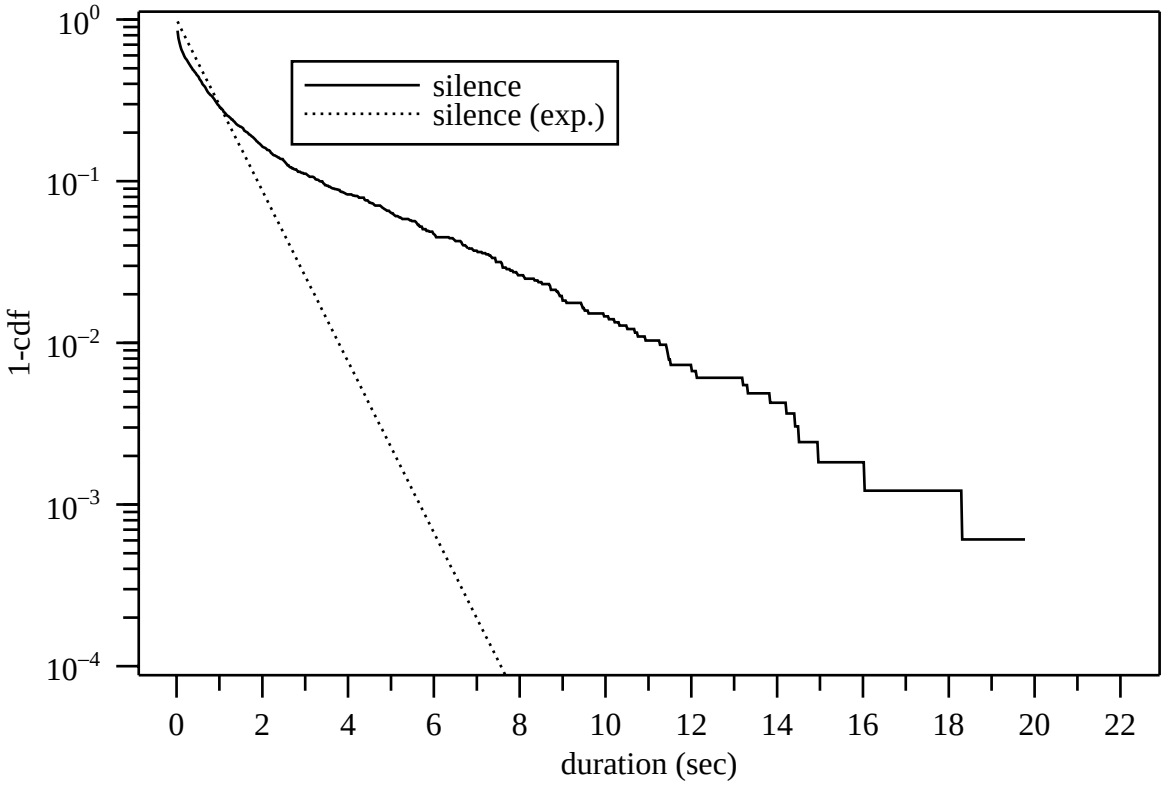

Figure 3.5. The silence duration distribution for packet audio

[68–74][12]. The "traditional" packet spacing (influenced by the frame size used by the linear predictive coder employed during the early low-bit-rate packet voice experiments within the ARPANET) is 22.5 ms, equivalent to 180 bytes at 8,000 samples per second. However, the packet stream appearing on the network is not as regularly spaced, as shown by the distributions in Fig. 3.4 and 3.5 computed from the voice conversation described in Section 3.5. There are three effects that make packet voice a very bursty source:

1. The terminal does not transmit packets deemed to contain silence or background noise. Silence suppression is particularly important for audio conferences with a large number of participants[13]. It significantly reduces the overall network load if mixing is done by the end systems, as every conferee is active only a small fraction of the time, but also prevents that the combined background noise from all conference locations is heard by each conferee. The latter part assumes greater importance as the availability of audio and video capable workstations moves teleconferences out of acoustically dampened teleconference rooms into regular offices. Breaking up the audio stream into talkspurts also allows easy playout delay adaptation and compensation for sampling clock drift between sender and receiver.

   Silences can be roughly divided into two categories. The first category with a small mean duration is caused by brief pauses between words and sentences, while much longer pauses reflect pauses while listening [65–67, 75–79]. Silence suppression accounts for the long tails in the interarrival time distribution. In a lecture with audience participation, the maximum silence period is bound to be even longer.

[12]Also, any fixed-rate coder can be transformed into a variable-rate coder by entropy-encoding the output symbols, for example by a Huffman code.

[13]The DARTnet research community routinely conducts teleconferences with more than a dozen participants.

Fig. 3.5 shows by example that an exponential distribution with the same mean does not fit the measured silence distribution well. The dotted line shows the inverse distribution $(1 - F(t))$ of the silence period duration, which has far shorter tails than the actual distribution.

2. The second reason accounts for variable interarrival times at the beginning of talkspurts, where the silence suppression algorithm commonly used also explains the extremely short interarrival times seen, down to 0.7 ms. Energy-based silence detection typically misses the important onset of particularly unvoiced speech, introducing audible clipping [80–83]. To reduce this so-called front clipping, the packet voice terminal buffers a small number of voice packets during silence periods and then transmits on the order of 40 ms of stored audio before the first packet containing sufficient energy to be declared speech.

3. The third reason for non-uniform interarrival times affects every packet within the talkspurt and has not been widely discussed. It is caused by the scheduling delays introduced by a non-real-time operating system, where a variable amount of time can elapse between the time a packet's worth of audio has been sampled and the time the voice application actually submits a packet to the network. As shown in Fig. 4.5, this can introduce a jitter of up to 3.5 milliseconds, even if the system only serves the audio application and some routing and clock synchronization daemons. Additional user processes could be expected to drastically increase this jitter.

   Fig. 3.4 shows the inverse distribution $(1-F(t))$ of the packet interarrival time. While most of the interarrival times are concentrated around the nominal value of 22.5 ms (shown as a dotted line) and twice that (shown as a dashed line), the interarrival times span the range from 0.7 ms to 171 ms, with times up to 100 ms occurring with non-negligible probability. The extremely long interarrival times are probably due to process lockouts caused by window resizing or movement.

Details on the implementation of a voice terminal within the Unix operating system can be found in Section 4.4.

One experiment was performed to gauge whether the multiplexing of such voice sources obeys the two-state models commonly used [84,85]. The source was modeled by a two-state process characterized by an average burst size of 30.69 packets, a mean silence duration of 1.23 seconds and an interpacket spacing of 22.5 ms during the talkspurt. The voice sources were offered to the same five-node network topology as in Fig. 3.1. To allow easy comparison with actual network experiments over DARTnet, the network speed was set to 1.344 Mb/s and packets were scaled to a fixed length of 600 bytes. The results shown in Table 3.7 demonstrate that the two-state model significantly underestimates both the average delay (by roughly a factor of two) and the tail of the delay distribution. The agreement between the models appears to improve slightly for long-haul flows, as their queueing delay is influenced less by the original source characteristics and more by the shaping applied by the queues themselves.

Table 3.7. Comparison of queueing delays experienced by voice traffic and equivalent two-state model

	path length							
	1		2		3		4	
traffic	mean	99.9%	mean	99.9%	mean	99.9%	mean	99.9%
voice trace	10.0	345	19.1	340	20.2	359	32.2	407
two-state model	5.0	229	9.6	238	12.3	281	17.5	319

Based on this negative example, we claim that the traditional two-state Markov model [16] where talkspurts with arrivals that are spaced at constant time intervals alternate with exponentially distributed silence periods does not capture the complexity of this traffic generation process. Firm conclusions certainly require more extensive traces. It should also be emphasized that the results will depend strongly on the type of silence detector used, in particular, on whether the detector ignores

short inter-word pauses or not. A suitable statistical model, preferably one that is analytically tractable, to capture these effects needs to be developed and tested. Brochin and Thomas [86] present a three-state model that might make a good initial candidate. (The paper [86] does not address how to fit the parameters to observed data or how well the three-state model can indeed predict, say, queueing delays.)

The conversational and lecture audio sources (the latter will be covered in Section 4.3) also point out the difficulty of policing audio with a token bucket (also called a "leaky bucket") [87–91] unless the content of the traffic carried is known in advance. Since we usually cannot accurately predict how much of the time a speaker will be active during a conference, we can only set the regulator to very near the peak rate.[14]

The difficulty of policing mean rate even if the mean rate is assumed to be known can be illustrated by the example as well. If the conversational voice source is regulated to half the peak rate, i.e., assuming that each party participates equally in the conversation, the regulator drops about 4% of the packets if the token reservoir holds ten average talkspurts worth of tokens (here, 307 tokens). The two-state model described later also shows a similar behavior if subjected to the regulator.

Finally, some indication of clock drift can also be derived from this example. A comparison of sequence numbers with trace time stamps showed that over the roughly one hour duration of the conversation, the sampling clock fell behind real time by about 360 ms, thus indicating the need for both inter-media (lip-synch) synchronization [92–100] and clock drift synchronization between sender and receiver.

---

[14]On the other hand, some may consider degraded quality for those dominating a conference a feature, not a problem.

Chapter 4

Experiments in Traffic Scheduling

This chapter describes an implementation of hop-laxity scheduling within an operational router. Hop laxity scheduling was described in Section 3.2 and evaluated by simulation experiments in Section 3.4. Measurements performed on the kernel implementation allow us to gauge the effect of protocol overhead and queue processing costs on the overall algorithm performance, providing a more realistic picture of the achievable performance gains. It also establishes how readily non-FIFO disciplines can be integrated into a "standard" Unix-based operating system and existing network protocols. Issues of set-up, largely unaddressed by simulation experiments, are also discussed in some detail, as they drastically affect the usability of the proposed scheduling mechanism.

In Section 4.1, we describe the experimental network used for the experiments. The hop-laxity scheduling algorithm requires some additions to existing network protocols, which are discussed for the examples of IP and ST-II in Section 4.2. That section also describes some design trade-offs, the initial distribution of algorithm parameters and the changes to the kernel network modules that implement the protocol. Section 4.3 then covers the traffic sources used in the experiments, drawn mainly from actual audio and video network applications, including a videotaped lecture and a transatlantic packet voice conversation.

A network voice terminal written by the author served as one application used to generate trace data. Some of its features and its design are summarized in Section 4.4. Another set of tools, described briefly in Section 4.5, then used these traces to regenerate network traffic. The trace-based traffic generator allows us

to reconstruct network traffic, even when the voice or video codec is not available at the remote site. Within some limitations, it also assures repeatability of the performance measurements.

The actual experimental results are summarized and discussed in Section 4.6. In the beginning of that section, we also illustrate the difference between the performance predicted by simulation and that seen in an actual network and strive to illuminate some of the potential causes for these differences.

4.1 The Experimental Network

All network experiments were performed within DARTnet, the DARPA-sponsored research network [101] spanning the continental United States. This network does not carry regular Internet traffic and can be reserved for the exclusive use of individual research groups. Sun SPARCstation 1+ workstations, currently running a modified SunOS BSD 4.3-based kernel, serve as routers, thus allowing modification of packet scheduling, instrumentation, implementation of new routing algorithms, etc., while providing a high-level operating environment. [1]

All nodes within DARTnet are connected by T1-lines running at 1.344 Mb/s[2]. The logical topology of the network is shown in Fig. 4.1; the links are annotated with the estimated per-link propagation delays measured in milliseconds. The node locations are listed in Table 4.1. A POP designates a point of presence, i.e., telephone company provided equipment closet, where the router connects to the physical transmission facility.

[1] The kernel was modified to support IP multicasting [102–104] [105, p. 281f] and the ST-II stream protocol. In addition, it incorporates network performance enhancements, such as larger packet buffers and other algorithmic improvements, mostly for TCP, contributed by Van Jacobsen. He also wrote the device driver for the high-speed serial interface board, that we modified for our packet transmission discipline.

[2] The speed has been reduced below the nominal rate of 1.536 Mb/s to accommodate the deficiencies of the HSI/S serial interfaces.

Table 4.1. DARTnet node locations (as of August, 1992)

name	institution	geographic location
parc	XEROX Parc	Palo Alto, CA
ames	NASA, Ames	Sunnyvale/Mountain View, CA
sri	SRI	Menlo Park, CA
lbl	Lawrence Berkeley Laboratory	Berkeley, CA
isi	Institute for Information Sciences	Marina del Ray, CA
udel	University of Delaware	Newark, DE
bbn	Bolt, Beranek and Newman	Cambridge, MA
mit	Massachusetts Institute of Technology	Cambridge, MA
dc	unattended POP	downtown Washington, D.C.
la	unattended POP	downtown Los Angeles, CA
sun	Sun Microsystems	Sunnyvale, CA
bell	Bell Communications Research (Bellcore)	Morristown, NJ
darpa	Defense Advanced Research Projects Administration	Reston, VA

The T1 lines are connected through CSU/DSUs[3] that perform electrical signal conversion to four-channel serial interface boards (called HSI/S) attached to the SPARCstation system bus.

4.2 Protocol and Operating System Support for Laxity-Based Scheduling

The hop-laxity algorithm requires two pieces of information: the time remaining to expiration (laxity) and the number of hops until reaching the destination host. The laxity information must be carried with every packet, while the hop count remains invariant throughout the life of a connection and can thus be part of the connection state.

The scheduling mechanism was implemented under both the Internet Protocol (IP) [106] and the experimental stream protocol called ST-II [107]. In this section, we describe how the protocol data units were modified for each protocol to accommodate the new scheduling algorithm and how the kernel had to be adapted. For both protocols, the basic protocol operation was unaffected. Applications that

[3]channel service unit/digital service unit

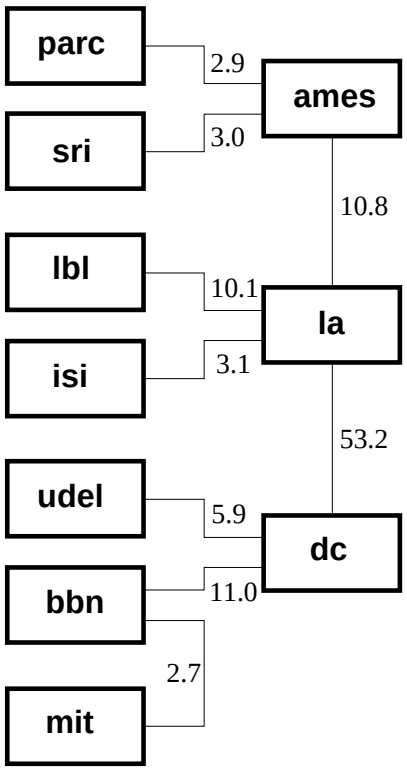

Figure 4.1. The DARTnet topology, with link round-trip propagation delays

do not wish to participate in the new scheduling algorithm or are unaware of its existence are unaffected, although they receive lower-priority service. In general, the kernel modifications were limited to a small number of routines within the BSD networking code and the ST-II control message and data packet routines.

4.2.1 IP Implementation

4.2.1.1 Extensions to the Internet Protocol

The standard IP protocol does not support time stamp or hop count information, but is extensible through options. Each IP datagram may carry zero or more options, up to a total length of 44 bytes (that is, the maximum header length of 64 bytes minus the fixed header length). There are a small number of predefined one-byte options, while the remainder of the option types have a fixed self-describing structure shown in Fig. 4.2. The first two bytes of an IP option consist of the option code octet and

the option length, measured in bytes. The option code octet is broken into three parts. If the copy bit is set, the node has to copy the option into fragments if the IP datagram has to be split into fragments when being forwarded. If the copy bit is zero, the option is only copied into the first fragment. The 2-bit option class determines whether the option is used for datagram and network control or debugging and measurement. Finally, the 5-bit option number determines the interpretation of the remaining option bytes according to a list of assigned well-known numbers. Given the length field and copy bit, a node that does not understand a particular option code can just copy the option into the outgoing packet without needing to know its interpretation.[4]

Since none of the existing options convey the needed information, we define a new experimental IP option with a code of 10_{10}, the lowest currently unused option code within the datagram control class. The format of the option is shown in Fig. 4.2. For this scheduling application, fragments should be treated as the original packet and thus the copy bit is set.

The newly defined option encompasses a total of eight bytes. The option code (value $8a_{16}$) and length (value 8), each one octet, are followed by a *flag* octet, which modifies the behavior of the scheduling algorithm. If the least significant bit of the flag octet is set, packets that have exceeded their laxity (i.e., when the laxity value has become negative) are dropped. If that bit is zero, scheduling priority increases with negative laxity. The second least significant bit switches from hop-laxity to a simple minimum laxity algorithm by not decrementing the remaining hop count from its initial value of one.

[4]There is, however, a set of options defined in [106] that have to be handled appropriately by all IP agents.

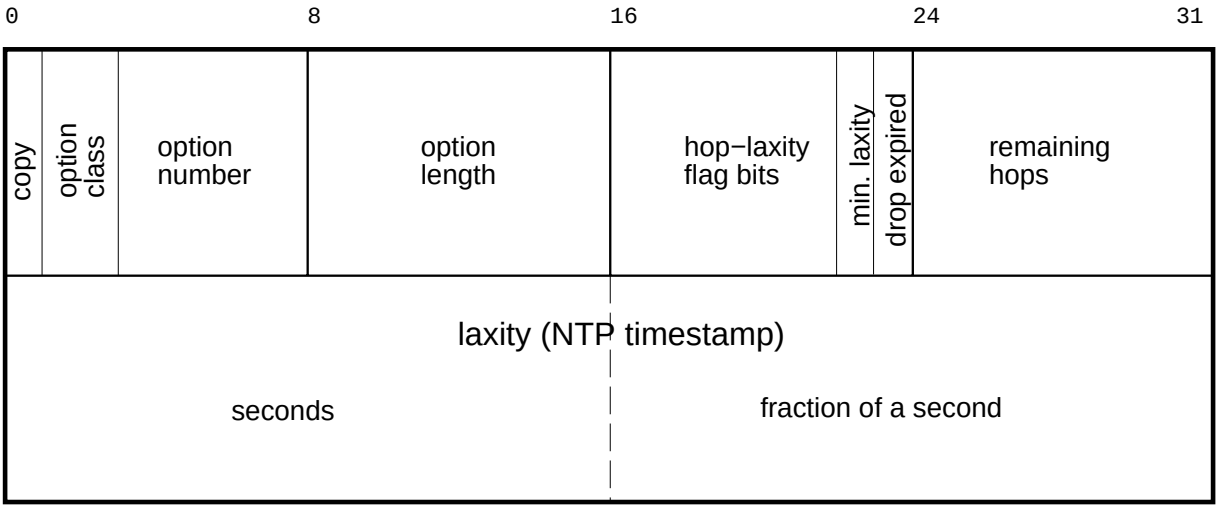

Figure 4.2. The hop-laxity IP option

4.2.1.2 Hop Count Information

The fourth octet within the option contains the number of hops remaining for this datagram. It is decremented by one at each router and considered as a signed quantity. (As an alternative, we could disallow negative hop counts and discard packets with zero remaining hops, as it indicates incorrect path length measurements. This may not be desirable, however, since the route may change during the life of a transport association. If packets with invalid hop counts are discarded, the receiving end would have no way of distinguishing this route change from normal packet loss or loss of network connectivity. This drawback could be avoided by sending an ICMP (Internet Control Message Protocol) [108] message of type 11 (Time Exceeded for a Datagram) in that case, possibly with a different code value indicating the cause than the currently defined codes.)

It has been proposed to augment the connectionless Internet protocol with "soft state" at each router. Soft state would be set up implicitly by (options within) the first data packet in a flow or by explicit set-up packets. The source address/destination address pairs, possibly augmented by port information from the transport layer protocols or a flow/class identifier carried within the IP header,

would be used to map an incoming datagram to a soft state record. Alternatively, the already defined stream identifier option (option number 8), currently considered obsolete, could be used to identify the flow, allowing flows to span several destination addresses.

Unlike a true connection-oriented model, only scheduling information, not the next hop, would be drawn from the soft state record. Thus, packets would continue to flow, albeit without the benefit of traffic-class specific scheduling, even if the soft state record got lost due to router failure and recovery. The use of soft state has been suggested, for example, for maintaining resource allocations within routers [109]. Clearly, such a soft state record could also hold the remaining hop count.

In a connectionless protocol such as IP, the path length (hop count) can only be approximately determined, since by definition, there is no guarantee that any two packets travel the same route through the network. However, for most existing networks, the hop count and even the route change infrequently. Thus, it appears to be sufficient to initially establish the path length when setting up a transport association and to then reevaluate the path length based on ICMP messages received, as described above, or the actual remaining hop count found in arriving messages. In the latter case, the receiver would have to notify the sender of the change in path length. Unfortunately, current networking implementations based on the Berkeley (BSD) distribution lack a mechanism for a user process to be notified about ICMP messages or to inspect the IP options found in an arriving IP packet.

The initial path length can be determined in a number of ways. A simple, though not completely foolproof method, implemented uses the ICMP echo request (ICMP type 8 and 0) and the IP time-to-live (TTL) field. Each node along the path decrements the TTL field by one[5]. Since ICMP apparently uses the maximum TTL

[5]The IP specification [106] requires that a gateway decrements the TTL by the number of seconds the packet has resided in that gateway, with a minimum decrement of one. Given current network speeds and implementations, the time-based rule appears to be rarely if ever used.

for outgoing IP packets, the TTL value of the incoming packet can be used to deduce the number of intervening hops. Since the TTL value to be used for ICMP messages is only required to be sufficiently large to reach the destination, this method may not work reliably for all host implementations.

As an alternative, ICMP echo requests with the IP record route option can be used. However, only 40 bytes are available for that option, allowing a maximum of 10 hops, as each IP address takes up four bytes. Within DARTnet, this is not a serious limitation, but paths across the Internet routinely require fifteen or more hops.[6]

The scheme to determine path lengths described above can be extended to a multicast network [102] by simply using the maximum computed hop count. A suitable time period has to be defined during which ICMP echo responses are accepted. Since nodes can join and leave the multicast group without the source having any direct way of knowing about it, the maximum hop count has to be periodically redetermined. Alternatively, destinations finding that packets arrive with negative hop counts could advise the source appropriately. It should be realized, however, that in an IP multicast setting with dynamic group membership, any hop count information has to remain approximate. The details of determining hop counts within multicast groups are left to future research.

4.2.1.3 Timing and Deadline Information

Our hop-laxity algorithm requires information not only about the hop count, but also about the time remaining to expiration. Depending on network properties and desired scheduling algorithm properties, a number of alternatives as to the information carried by the packet can be considered:

source time: The timestamp encodes the departure time of the packet from the source. This time may not necessarily be the actual time the packet reaches

[6] The path from `erlang.cs.umass.edu` to `cub.isi.edu` spans 19 hops, for example.

the network interface, but could be a time meaningful to the application, for example, the time a video or audio frame was generated. This requires that the nodes maintain information about the permissible laxity and that laxity remains the same for each packet within a stream.

deadline: The timestamp encodes the desired arrival time of the packet. Thus, if higher level demultiplexing is desired, packets can easily be marked as more or less urgent. With this encoding, the nodes need to keep no state information.

queueing delay: The timestamp contains the permissible sum of all queueing delays. Each node then subtracts the actual queueing delay experienced. This requires that each packet be accurately timestamped on arrival to the node or, if generated locally, when it is passed down from the socket layer. (The BBN ST-II implementation provides such functionality.) Hardware and software interrupt latencies are not accounted for in this case. In this method, only queueing delays, not propagation or transmission delays enter into the scheduling decision. Also, the delays accrued by routers not participating in the scheduling algorithm are not accounted for.

All but the last timestamp encoding require clocks to be synchronized to within a few milliseconds. This is well within the current state-of-the art, as evidenced by the sub-millisecond synchronization found within DARTnet [110].

Any timestamp that includes the transmission time may lead to packet reordering within the same stream if two packets with different lengths follow each other closely.

We chose to use the second approach, that is, carrying the deadline within the packet, mainly because we wanted to eliminate any state information (here, laxity) that a node would have to maintain, as it would force us to implement and design a flow setup and look-up mechanism. Since the number of flows may be large, this could also add non-trivial packet processing overhead. It also allows us to

compensate by scheduling for different propagation delays, thus providing a quality of service less affected by physical distance.

The deadline timestamp is carried within the final four bytes of the option, encoded in network byte order (most significant byte first or big endian). The timestamp uses a shortened version of the network time protocol (NTP) [111, 112] timestamp, extracting the middle 32 bits from the original 64 bits. The shortened timestamp encodes the least significant 16 bits of the number of seconds since zero hours (Universal Time) January 1, 1990 plus a 16-bit binary fraction of a second, resolving time to about 15.2 μs. The NTP timestamp format and its shortened version have the advantage that addition and comparison are efficient, requiring no explicit carry, in contrast to the standard Unix time values represented as seconds and microseconds. The conversion from the Unix system clock to NTP timestamp is relatively simple:

```
ntp = ((tv.tv_sec + 2208988800) << 16) +
      (((unsigned)tv.tv_usec * 0x10c6
    + (((unsigned)tv.tv_usec * 0xf7a) >> 12)) >> 16);
```

Here, `tv` holds a Unix time value, with `tv.tv_sec` denoting the number of seconds since January 1, 1970 and `tv.tv_usec` denoting the number of microseconds within the current second. The conversion is required once per call to the output dequeue routine.

The shortened NTP timestamp wraps around roughly every 18 hours. Thus, during a window of a few hundred milliseconds, packets would be stamped with a deadline just above zero, while the transit hosts' shortened NTP clocks have not yet reached the rollover point. During this window of vulnerability, packets would be treated as if they had an extremely long laxity and scheduled appropriately. Since the occurrence of this event is sufficiently rare and the consequences bearable, the expense of adding explicit checks was judged to be unwarranted.

4.2.1.4 Kernel Modifications

Before describing the actual kernel implementation, we briefly outline the procedures a data packet traverses before leaving the network interface. A general description of the BSD networking code can be found in [113]. The data flow is also depicted in Fig. 4.3.

As we will modify the data structures used to hold packet data, we briefly describe the data structures used in BSD Unix. Network packets in the BSD networking code are stored in so-called mbuf's. Mbuf's are linked blocks of memory of 128 bytes each, which either directly contain packet data or point to larger (2048 byte) so-called "clusters". The first twelve bytes of each mbuf contain information about the amount of packet data contained in the mbuf, an offset into the mbuf that indicates where packet data begins, the data type and a pointer that is used to chain together all mbuf's within a chain. If headers are added to an mbuf, the offset is decremented and the length incremented. If not enough space is available, another mbuf is chained to the front or back. Chains of mbufs can be linked with another pointer field. This scheme allows dynamically growing data structures with easy storage reclamation and no memory fragmentation.

For our purposes, we are only concerned with outputting an IP packet. (While there is a queue that holds incoming packets from the device driver waiting to be processed by the `ipintr()` software interrupt handler, it is expected to be short, as IP input processing should not require large amounts of CPU time. IP input processing mainly consists of parsing options and determining the outgoing interface. Thus, it is probably inefficient to add any form of scheduling to the IP input queue.)

A packet can reach the output routine, `ip_output()` on two different paths. If it reaches the node from an external network, it is processed by the IP input routine

The actual implementation of the scheduling algorithm consists of two functions. The first function, `ip_hl_opt()`, translates the IP option described above into a data structure residing at the very beginning of the first packet. It is called at the

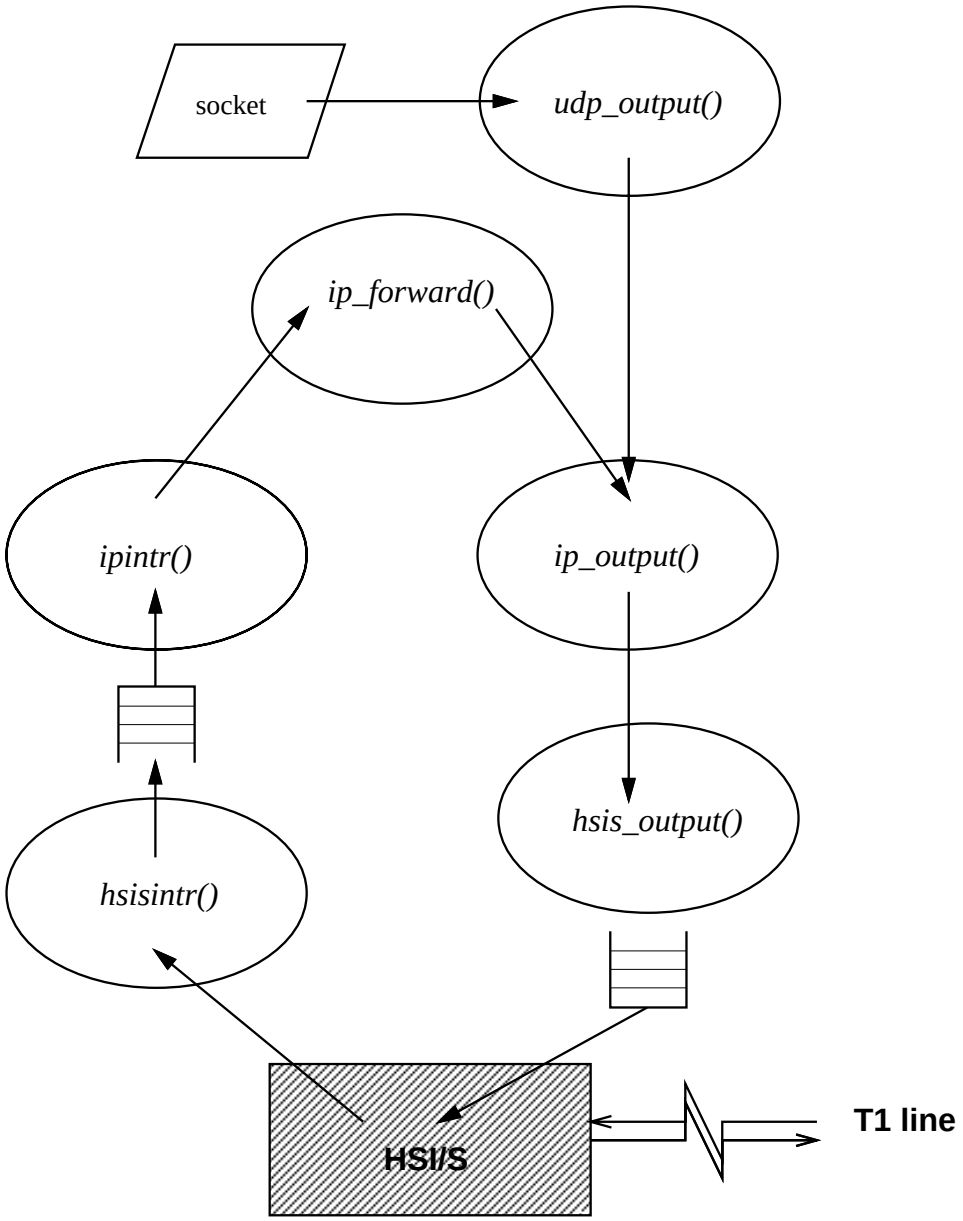

Figure 4.3. Data flow in BSD kernel for IP

beginning of `ip_output()`. The first 16-bit word of the mbuf data area designates the resource control family, so that multiple scheduling policies (e.g., virtual clock, weighted fair queueing, FIFO+ and other mechanisms [114] [15] [46] [115] [36]) can be supported.

The translation routine of that mbuf is then changed to `MT_DATASORTED`, a newly defined mbuf type. This translation takes place in the `ip_output` routine; it requires a linear through all IP options within the packet to determine if the scheduling option is present. The code module also decrements the hops-to-go field.

Instead of transcribing the IP option in `ip_hl_opt()`, we could have the dequeueing routine parse the IP options directly. This, however, has three severe disadvantages. First, note that not every packet is "seen" by the output scheduler. The packet is directly copied to the device buffer if the device is idle when the device driver is invoked. Thus, any hops-to-go adjustments need to take place outside the dequeueing routine. Secondly, it is undesirable to embed knowledge about the network layer header format into a device driver. By divorcing the IP layer representation and the data structure seen by the driver dequeueing routine, several network layer protocols (e.g., ST-II, OSI CLNP or X.25) could transparently use the same queueing mechanism. Third, since each packet may have to be inspected several times while residing in the queue, access to the scheduling information has to be as fast as possible.

While the scheduling information is inserted into the first mbuf of a packet, further link-level processing may prepend another mbuf to make room for link-level headers. Thus, the dequeueing routine searches for an mbuf of type `MT_DATASORTED` within the first two mbufs. Packets that are not marked do not enter the priority computation and are thus served with lower priority.

Within Berkeley Unix, the actual packet scheduling takes place when the device driver completes an output operation and pulls the next packet off the output queue.

4.2.1.5 Scheduling Overhead Considerations

The hop-laxity scheduling algorithm imposes a significantly greater processing burden at packet dequeueing time. However, for the serial interface used in the experiments, channel utilization should not be affected, for the following reason. The HSI/S device driver manages two fixed-size output buffers (of size 1500 bytes). While one output buffer is being output to the serial line by the on-chip DMA controller, the other buffer can be filled with the next packet to be transmitted. Thus, line output overlaps with dequeueing and copying from system memory to board buffer. For long queues of very short packets, however, it is possible that selecting the next packet to be output may take longer than the packet transmission time, thus idling the channel.

Measurements were made to estimate the cost of hop-laxity scheduling on the DARTnet routers. The cost consists of the delay incurred each time the output dequeue routine is invoked plus the sum of the processing times for each packet in the queue. The first component consists of obtaining the current system time, converting it into the shortened NTP timestamp, which takes a total of about 10 μs, and the linked list manipulation to extract the selected packet from the queue. The latter cost would also have to be borne by a FIFO queue, except for some slight penalty for potentially having to extract a packet from the middle of the queue.

The dominant cost for hop-laxity scheduling, however, is caused by having to compute the scheduling priority of each packet in the queue, entailing a division operation, and then determining the minimum priority value. For the SPARC architecture, the cost of arithmetic operations differs from common perceptions. The CPU chip lacks integer division and multiplication instructions, but does have a fast floating point unit. The execution times of various arithmetic operations of interest here are shown below. Note that these times encompass all necessary data access, not just the arithmetic operation itself.

operation	time (μs)
`r = c >> 2`	0.37
`r = c / 4`	5.64
`r = c * 4`	6.28
`r = c / 4.0`	3.65
`r = c * 4.0`	1.39

The table shows that integer multiplication and division should be avoided in favor of shifts and floating point multiplication. The scheduling code takes note of that and also makes use of the fact that the divisor, i.e., the remaining hop count, is typically a small number. It treats each hop count up to twelve separately, evaluating divisions by 2, 4 and 8 by right shifts, other divisors by floating point multiplications with a constant. As another minor performance improvement, the timestamp conversion and minimum search is performed only if at least two packets reside in the queue.

With these optimizations, each packet within the queue takes about 8 μs to process during each call to the dequeueing routine. With long queues, this can clearly be a significant burden for the processor, particularly if it has to attend to a number of busy interfaces. It should be noted, however, that typical implementations of bandwidth management mechanisms such as fair queueing or virtual clock also require searching on average half the queue for each output packet to determine the proper insertion point. However, typically, no arithmetic beyond a simple comparison is required.

As pointed out earlier, the dual-buffering scheme prevents the processing overhead from reducing channel utilization under normal circumstances. If short packets prevent a complete parallelization of copying and output, the scheduling operation could be shifted to the input queue, by computing the priority value and using it to insert the newly arriving packet at the appropriate location. The dequeueing routine would then only have to pick off the first element from the queue. Since the laxity computation does not take the queueing delay at the node into consideration, the algorithm performance is anticipated to deteriorate as compared to the current

implementation. As another alternative, instead of searching the whole queue for the packet with the lowest priority value, the search could be limited to the first n packets, under the assumption that the most urgent packets will most likely be found there [116]. The impact of these implementation alternatives on both performance and computational cost is left for further study.

4.2.2 ST-II Implementation

The ST-II implementation is very similar to the IP implementation described at length above. Thus, we will focus only on those parts that differ substantially. For the experiments, we made use of the implementation of ST-II for SunOS written by Charlie Lynn of Bolt, Beranek and Newman (BBN), Cambridge, Massachusetts.

ST-II is a connection-oriented internet protocol. The protocol description states [107, p. 7]:

> ST has been developed to support efficient delivery of streams of packets to either single or multiple destinations in applications requiring guaranteed data rates and controlled delay characteristics. ST is an internet protocol at the same layer as IP. ST differs from IP in that IP, as originally envisioned, did not require routers (or intermediate systems) to maintain state information describing the streams of packets flowing through them. ST incorporates the concept of streams across an internet. Every intervening ST entity maintains state information for each stream that passes through it. The stream state includes forwarding information, including multicast support for efficiency, and resource information, which allows network or link bandwidth and queues to be assigned to a specific stream. This pre-allocation of resources allows data packets to be forwarded with low delay, low overhead, and a low probability of loss due to congestion. The characteristics of a stream, such as the number and location

> of the endpoints, and the bandwidth required, may be modified during the lifetime of the stream. This allows ST to give a real time application the guaranteed and predictable communication characteristics it requires, and is a good vehicle to support an application whose communications requirements are relatively predictable.

As it is a common misunderstanding, it should be emphasized that ST-II provides only the data structures, but not the mechanisms and policies that allocate resources or guarantee a certain quality of service.

The major difference in scheduler implementation is caused by the fact that ST-II is connection-oriented, while IP is connectionless. Therefore, the hop count can be readily maintained as part of the connection state in ST-II. Within the experimental implementation, the one-octet hop count was inserted into the flow specification just after the flow version. Since the flow specification already specifies a limit on delay and timestamps are generated by the ST-II kernel modules at the source rather than the application, the source timestamp option suggests itself, as opposed to the deadline used for IP.

ST-II connections are established in two steps. First, the source propagates a `CONNECT` message to all target nodes along the multicast tree, which then respond with an `ACCEPT` message routed back along the tree to the source. Each intermediate node increments the node count within the flow specification included with the `CONNECT` message. As in the unmodified protocol, the transit nodes maintain the flow specification contained in the `CONNECT` message. The `ACCEPT` message contains the resource specification as modified by the target. It is passed back through the transit nodes, which now leave the hop count unmodified. On receiving an `ACCEPT` message from a target, each transit node computes the hop distance to the target in the `ACCEPT` message by subtracting the hop count found in the `CONNECT` message from that in the `ACCEPT` message. The maximum hop count is then updated as appropriate.

The hop count and timestamp information are copied to the first mbuf of a data packet's mbuf chain in the enforcement function, which is a scheduling-specific function invoked just before the output routine. The enforcement function[7] inserts the deadline computed from the sum of origin timestamp and the permissible delay into the first mbuf, in the same manner as for IP. The dequeueing function is independent of the network protocol; it was described earlier in Section 4.2.1.4.

The implementation of hop-laxity scheduling within ST-II has two advantages. First, the resulting data packet length is significantly smaller. Equivalent transport services within the IP stack require the UDP transport layer for port information. IP plus UDP require a minimum of 28 header bytes, with the hop-laxity scheduling option adding another eight bytes, for a total of 36 bytes of overhead. ST-II, on the other hand, only has an eight-byte header, plus another eight bytes for the full NTP timestamp, for a total of sixteen bytes. The second advantage concerns the computation of the hop count. Since the source explicitly adds destinations to the multicast tree, the hop counts at intermediate nodes are readily kept current.

The major disadvantage of using ST-II is the complexity of the implementation – about 17,000 lines of C code, as compared to the roughly 3,500 lines for IP with multicast extensions, plus 400 lines for UDP. Applications using ST-II also tend to be significantly more complex than those using IP/UDP only. Currently, a full implementation of ST-II is only available on SunOS. Before ST-II is widely accepted, it will probably have to be significantly streamlined and made easier to implement [117]. Also, for some applications, the connection set-up delay may be significant.

4.3 Traffic Sources

For the network experiments we ran in order to obtain packet delay data in DARTnet under the hop-laxity scheduling discipline, the traffic was generated from traces

[7]The function is used for enforcing flow rates for other scheduling mechanisms.

based on real network transmissions rather than statistical models. Three sources were used, representing two audio and one video streams.

4.3.1 Lecture Audio and Video

The sources designated 'lecture audio and video' were drawn the traces produced by encoding parts of a video tape from a distinguished lecture given by Dave Cheriton at the University of Massachusetts in 1989. The Bolter Vision video encoder was used to compress the 30 frame/second full-color NTSC video data into a bit stream with a data rate of 128 kb/s. The synchronous data provided by the codec was packetized, stripped of filler bytes and prefixed with the PVP header [56] by the **pvp** packet video processor program (written by Dave J.W. Walden at the Information Sciences Institute) to arrive at packets averaging 457 bytes and ranging in size from 16 to 534 bytes, with an effective data rate of 109,680 b/s. With UDP and IP packet headers added, the data rate reached 116.6 kb/s. The hop-laxity IP option added 1824 b/s. The frame rate was found to be 28.5 frames per second, slightly lower than the nominal 30 frames per second. A total of 3411 seconds (about 56 minutes) of video was encoded.

The lecture audio was drawn from the same lecture. The audio data, μ-law coded 8-bits/sample PCM sampled at 8000 samples per second, was packetized into blocks of 180 bytes representing 22.5 ms of sound each. Then, silent packets were removed by a level-adaptive silence detection algorithm based on frame energy estimates [118]. All audio processing was performed by Nevot, a network voice application [119] described briefly below. A trace event was generated each time a packet was submitted to the socket library. In this manner, network interface interarrival times for 100,000 frames of audio data spanning 2413 (about 40 minutes) seconds of real time were gathered. The interarrival time spanned the range from 0.71 ms to 1.86 seconds, with an average of 24.1 ms and a 99 percentile value of 65 ms. With IP and UDP overhead, the 61 kb/s audio rate measured for this test

signal creates a network load of 71.6 kb/s. The hop-laxity IP option adds another 2844 b/s. Even though audio and video data were generated at a fixed rate in the codecs used here, the non-real-time nature of the Unix operating system introduces several milliseconds of jitter due mostly to process scheduling delays.

4.3.2 Conversational Audio Source

A description of the conversational audio source can be found in Section 3.4. Where indicated, seven of the conversational audio sources were multiplexed. Rather than using several different packet traces, independence of the sources was assured by having the traffic generators read the trace files with an offset of 4000 samples each, i.e., the first generator would start reading at sample 0, the second at sample 4000, etc. To increase variability and model high-quality variable bit rate audio coding, the packet length was made to vary uniformly around the mean value of 250 bytes, $\pm$ 200 bytes. With headers, the multiplexed sources had an aggregate data rate of 260.8 kb/s.

For future work, use of the variable-rate video source described in [120] is planned.

4.4 The Network Voice Terminal

Throughout this dissertation, packet voice traffic is repeatedly cited as an example for real-time traffic. Despite its relatively low bandwidth, as compared to video, for example, its delay and continuity requirements are fairly strict.

Research in transmitting voice across a packet network dates back to the early ARPAnet days. Cohen [121] refers to cross-continental packet voice experiments in 1974. According to [122], low-bit rate voice conferences very carried out in 1976. The early '80s saw experiments of transmitting low-bitrate voice across mobile radio [77, 123, 124] and satellite [125] packet channels. The first Internet packet voice protocol was specified formally in 1977 [126], and a packet video standard followed in

1981 [56]. The CCITT standard G.PVNP [127] was published in 1989. Currently, an effort is underway within the Internet Engineering Task Force to develop a transport protocol (called RTP) suitable for packet voice and video as well as other real-time applications [128, 129]. Due to its ample bandwidth and relatively short albeit variable delays, packet voice across local networks such as Ethernet [130], token ring [131, 132] or slotted ring [133] has drawn considerable implementation work. Note that packet audio/video should be set apart from the approach to voice/data integration that provides fixed-bandwidth circuits on multiple access networks [134–136].

Interest in packet audio has increased recently as more and more workstations come equipped with built-in toll (telephone) quality or even CD-quality audio hardware support. There exist a fair number of simple programs that utilize the SPARCstation audio hardware to communicate between two workstation on a local net, for example vtalk (Miron Cuperman, OKI) or PhoneTalk (Patrik Nises and Joakim Wettby, Royal Institute of Technology, Stockholm). Another example for LANs is LANBRETA-DTS [137], part of a LAN-based teleconferencing system. Programs designed for multiple-party connections across wide-area networks include VT [138] and `vat` (Van Jacobsen and Steve McCanne, LBL). A number of commercial products use medium-bitrate packet voice to more effectively utilize leased private lines, extending the concept of the traditional data-only multiplexer [139]. System implementations of packet voice terminals are described in [122, 140–142]. Packet radio experiments are featured in [143]. Surveys on packet voice performance are presented in [140, 144].

To aid in the research efforts on aspects of real-time services reported in this dissertation, the author wrote a complete packet voice application known as Nevot ("NEtwork VOice Terminal") described in more detail in [119, 145, 146]. Its design was guided by two goals: first, to offer extensive tracing and performance monitoring functions, secondly, to be easily extendible to accommodate new network, real-time

and session control protocols, inter and intra-media synchronization algorithms, fixed and variable rate audio encodings and user interfaces.

For this dissertation research and follow-on work, NEVOT was used:

- as a traffic source, both interactively and by providing trace data for network tests and simulations
- as a quality-of-service measurement tool, allowing the measurement of packet loss and end-to-end delay as actually perceived by the user, after delay adaptation and loss reconstruction
- as a test bed for improved delay adaptation, intra-media synchronization algorithms and protocols for real-time services [128, 129]

NEVOT has also served as a demonstration tool for real-time applications across both wide-area packet switched networks such as the Internet as well as corporate private networks. It has demonstrated both the feasibility of implementing real-time services within a non-real time operating system such as BSD Unix, and also highlighted the shortcomings of current operating systems in supporting real-time applications.

The main function of a packet voice terminal is the reconstruction of the synchronous audio stream produced by the remote source, in the face of random and nonstationary network and operating system delays. The audio device generates samples at a fixed rate, typically 8,000 samples per second, which are then gathered into packets of around 20 ms. (Currently, 180 samples representing 22.5 ms is most commonly used; this value dates back to the frame size of the 2,400 bit/second LPC vocoder used in early experiments.)

The packet voice receiver compensates for the delay variabilities introduced by the sender, the network and the network receiver processing by adding a variable delay between the arrival time and the time the audio data is submitted to the audio

output device. [147] [148] [149] [150]. In general, packets within a talkspurt have to be played out consecutively, so that playout delay adaptation can only take place at the beginning of a talkspurt. Since low end-to-end delays are highly desirable for interactive voice communication, adaptive rather than worst-case playout delays are typically used. NEVOT models the variable delays as a Gaussian random variable and adjusts the playout delay of the talkspurt beginnings as a constant factor times the current delay variance estimate. (An additional delay buffer on the order of 20 to 30 ms has to be added for workstations within the audio device to compensate for operating system scheduling delays.)

NEVOT is meant to serve four purposes:

- as a demonstration tool for Internet audio conferences,
- as a measurement tool to investigate traffic patterns and losses in packet voice applications across wide-area networks,
- as a demonstration implementation of real-time services in a distinctly non-real-time operating system (Unix)
- as a traffic source to validate and evaluate resource allocation protocols and algorithms.

The modular software architecture of NEVOT is shown in Fig. 4.4. Operational experience showed that multiple concurrent conferences or conversations are desirable, for example for private side chats within a conference. Within a packet voice setting, multiple conversations are much easier to support than with traditional telephony. Audio mixing and multiple audio channels make it possible to listen in on several conferences.

Since NEVOT is expected to operate over a wide range of networks, from FDDI to modem SLIP lines, it supports audio encodings spanning the range from 48 kHz sample rate, 16 bits/sample CD quality through standard telephony toll-quality

64 kb/s PCM[8], 32 and 24 kb/s ADPCM[9] [151] to low-bit rate vocoder quality 4 kb/s LPC. Support for variable rate encodings (e.g., by Huffman-encoding the ADPCM coefficients) is planned. The variety of audio encodings used makes the use of application-layer gateways necessary. NEVOT implements a gateway by making use of the multiple stream design, forwarding input for each stream, after transcoding, to all other streams. Privacy of conversations and conferences is assured by DES encryption.

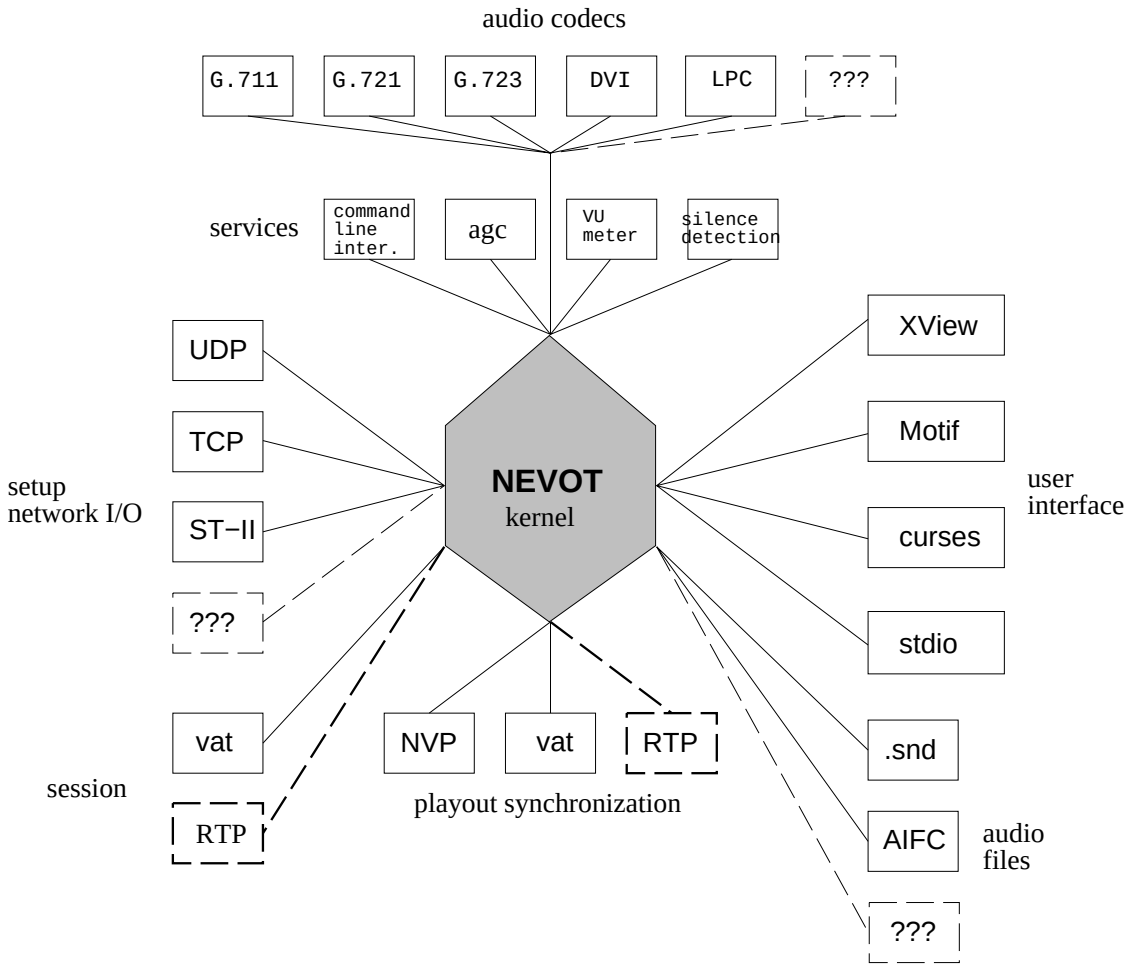

Figure 4.4. NEVOT Structure Overview

[8]pulse code modulation

[9]adaptive differential pulse code modulation

NEVOT interacts with the user through the Open Look™ graphical user interface on X-capable workstations or a simple command-based interface. A version with almost identical functionality geared towards ASCII terminals was also written. but is not described here. Nevot is currently implemented on Sun SPARCstations, the SGI Indigo workstation and the Personal DECstation.

4.5 A Trace-Based Traffic Generator

From the traces, network traffic was generated by a special-purpose program running on each network source node. Trace generators rather than actual audio or video sources were used since the remote site offered no easy way to remotely execute codecs. Also, the overhead of processing actual audio and video data would have made it impossible to run number of sources required to load the network.

The traffic generator operates as follows. It reads a packet arrival time from the trace file, shifts the timestamp appropriately (the traces contain absolute times) and then schedules a packet transmission at the next instant specified through the timeout parameter of a `select()` system call. Since the scheduling is not exact, the program keeps track of the actual packet send time and shortens the interarrival time to the next packet appropriately, assuring that the average rate is maintained, albeit at the cost of greater interarrival variability. The graph in Fig. 4.5 shows that the scheduling inaccuracy is limited to about 3.5 ms. It remains to be determined whether this scheduling jitter significantly affects performance measurements. Also, the effect of several concurrent processes on scheduling accuracy remains to be determined.

4.6 Experiments

After laying the groundwork by covering the sources (Section 4.3) and network topology, we now present the results from a set of experiments within DARTnet.

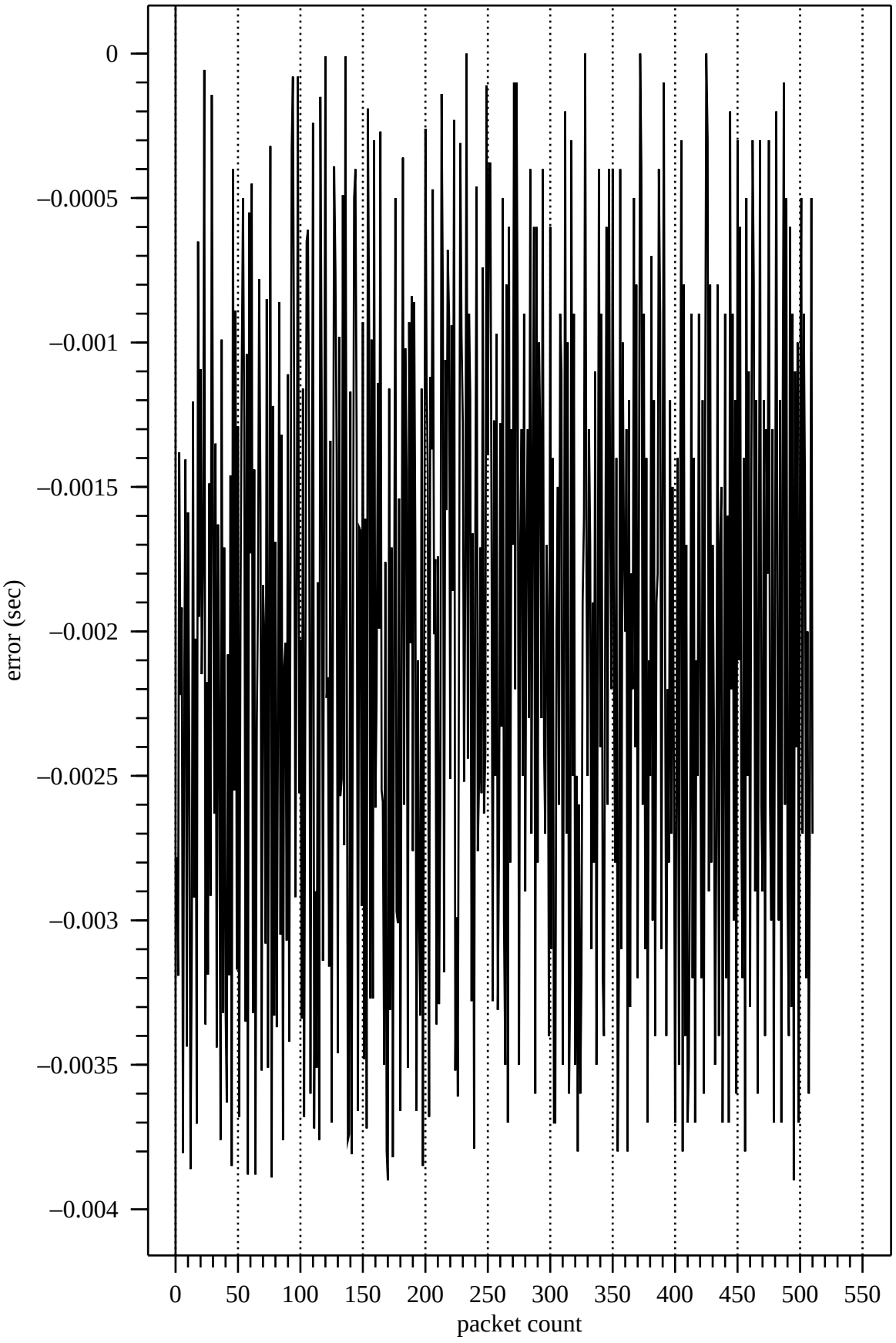

Figure 4.5. Scheduling jitter for trace-based traffic generator

First, in Section 4.6.1, we show that the accuracy of simulations in predicting actual network performance, even with appropriate corrections, is limited. With that caveat, we then summarize the results of network measurements providing insight into the performance of the hop-laxity (HL) queueing policy as compared to the standard FIFO policy. All DARTnet experiments described in this section ran the IP version of the kernel described earlier in Section 4.2.1

4.6.1 Simulation as Predictor of Network Performance

To gain an understanding of how well simulations can predict actual network performance, we mapped the topology used in Section 3.4 into DARTnet, with nodes 1 through 5 represented by DARTnet nodes `parc`, `ames`, `la`, `dc` and `bbn`, respectively. For both simulation and network measurements, only streams arriving at node 5 (or DARTnet node `bbn`, respectively) are taken into account.

In order to transmit an equal number of bits during the simulation and network experiments, the header overhead incurred by the DARTnet protocol stack needs to be taken into account. The HSI/S serial interface supports HDLC as its data link layer. HDLC adds two bytes of framing and 2 bytes for a cyclic redundancy check (CRC) to each frame. We ignore the overhead caused by bit-stuffing since the data portion of the packets used in the experiments is zero and thus not affected by bit-stuffing. The point-to-point protocol (PPP) serves as the media access layer and adds four bytes, two of which are fixed-value HDLC address and control bytes. IP, without options, and UDP add another 20 and 8 bytes of header, respectively, for a grand total of 36 header bytes.

Another factor to be taken into account when comparing simulation and network measurements is that most network interfaces cannot keep the channel busy continuously due to interrupt latencies, register set-up time and the like. By carefully following a UDP packet train submitted at maximum rate as it progressed through the network interface, we could estimate the latencies involved as well as the achiev-

able channel utilization. Since the latencies depend on other system activity and the packet length, only rough estimates are possible. For 600 byte packets, the time between packet transmissions was found to average about 3.67 ms, while pure transmission time at a rate of 1.344 Mb/s would only require 3.571 ms. The idle time of 98 μs per packet translates to a roughly 2.7% loss of channel capacity. Throughout the simulation experiments reported in this subsection, we thus model the channel as having a capacity of 1.308 Mb/s. The achievable `ftp` throughput measured at 1.318 Mb/s (including IP and TCP headers), but with significantly longer (1500 byte) packets, lends some credence to this estimate.

At low load, i.e., without queueing, a packet would experience a latency of approximately 460 to 600 μs, composed of the following components. The hardware interrupt triggered by receiving a frame at the serial interface was measured to take about 75 μs; then, 18 μs would elapse until the software interrupt posted by the hardware interrupt was serviced. The software interrupt itself would add another 150 μs. 160 μs elapsed until the output handler was started, the time being occupied by IP processing. The actual transmission then began about 160 μs later.

The delay measurements within DARTnet are by necessity end-to-end, including all propagation and transmission delays. We used two methods to estimate these fixed delays, both yielding results within 2 ms of each other. The first method used a `ping` program with enhanced timing resolution to measure the round-trip delay between sending ICMP echo request packets and receiving the corresponding ICMP echo reply packet. Propagation delay was estimated at half the minimum round-trip delay measured, ignoring the transmission time for the short (about 60 byte) packets. The results of these measurements are unaffected by clock synchronization and are shown in Fig. 4.1. The second method simply takes the minimum delays measured during the experiment as an indication of the combined propagation and transmission delays.

Table 4.2 compares the performance estimated by the simulation and actual network measurements. Two sets of experiments were performed to gauge the relative impact of low and high loads on how well simulation results predicted actual network performance. In the first, labeled S1 and N1 in Table 4.2, the packets, with headers, were sized at 300 bytes, while the experiment labeled S2 and N2 in that table, consisted of packets of length 600 bytes. Note that the system load for the low-load and high-load experiments is roughly the same. Except for additional time needed to copy longer packets between kernel and user space, the overhead imposed by tallying arriving packets and generating new arrivals is the same for both long and short packets.

For both low and high network load, the simulation predicts the average queueing delays within the current clock synchronization spread. However, the delay percentiles are consistently underestimated by the simulation, for both low and high loads. The differences between simulated and measured performance are seen to be increasing with longer paths and higher load. Fortunately, for the high-load case, the error is still only roughly 5 to 10%. Since per-packet penalty should decrease with increasing packet size, the load-dependence of the error points to additional queueing within the kernel. The only additional queue in the IP forwarding path is that between the receive hardware interrupt and the software interrupt (see Fig. 4.3). Thus, if the processing of the software interrupt is delayed due to other system activity, packets could queue up there. Simply measuring the queue occupancy, however, will not suffice, as packets queued on input would probably been queued on output even in the absence of input processing delays.

Another factor, probably of minor importance, is the background traffic induced by the multicast group management protocol IGMP and the clock synchronization protocol ntp. Given that the system is operating in a high-load region (due to the burstiness of the traffic, the average utilization is only about 60%), even small

amounts of additional traffic could have a disproportionate impact on queueing delays.

Table 4.2. Comparison of queueing delays estimated by simulation (S) and network measurements (N) for FIFO scheduling

	path length							
	1		2		3		4	
traffic	mean	99.9%	mean	99.9%	mean	99.9%	mean	99.9%
S1	0.5	5.0	1.0	6.7	1.4	8.8	1.8	10.8
N1	0.8	1.0	1.0	12.7	1.7	13.8	1.9	17.7
S2	10.2	346	19.0	444	26.5	511	37.3	528
N2	10.7	362	20.8	492	28.7	545	38.3	559

It should be noted that the measurements have not been compensated for the clock offset between `bbn.dart.net` and the various source nodes. From information provided by the clock synchronization control program `ntpq`, the clock offset can be estimated at about ± 1 ms.

4.7 DARTnet Experiment: First Topology

This section discusses the first experiment comparing the measured network performance of the hop-laxity (HL) and FIFO policies. The network flows are depicted in Fig. 4.6. The thick arrows originating at node `parc` and terminating at node `mit` indicate the flow of interest, while the thinner arrows represent cross traffic. Each flow consists of seven multiplexed conversational audio traces, as discussed earlier in Section 4.3.2. The assignment of flows to links assures equal loading of each link with the equivalent of 21 voice sources.

The flows were assigned a laxity of 500 ms. In the experiment labeled 'HL 5' in Table 4.3, each packet started out with a hop count of five, regardless of the actual number of hops the flow it belonged to was to traverse. This was meant to model a larger network, where the interfering traffic continues to other destinations beyond

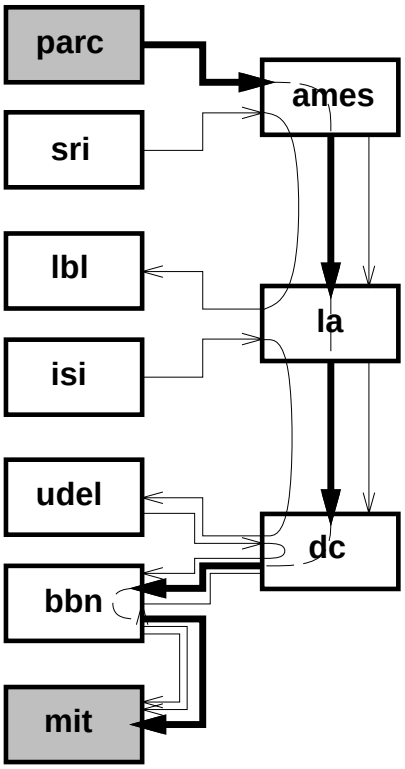

Figure 4.6. Traffic Flows for DARTnet Experiment

Table 4.3. End-to-end delay performance measured in DARTnet (topology of Fig. 4.6

	mean	delay percentiles		
	delay	0.90	0.99	0.999
policy	(ms)	(ms)	(ms)	(ms)
FIFO	60.5	80.1	162.5	243.0
HL 5	77.0	125.9	312.0	423.4
HL	55.5	64.8	99.9	133.5

the scope of the limited DARTnet topology. In the second experiment, the flows were marked with their "true" hop count, i.e., all interfering flows had initial hop counts of three and one, while the `parc` to `mit` flow started with a hop count of five. As one would expect, the latter provides a more favorable treatment to the long-distance flow, somewhat reducing its mean delay and significantly reducing the delay percentiles for all measurable and interesting percentile values. The 'HL 5' experiment indicates that some flows may be considerably worse off when hop laxity scheduling is introduced.

4.7.1 DARTnet Experiment: Second Topology

In a second experiment, DARTnet was used to model the topology described in [36], with the same flows and conversational voice traffic as discussed in Section 4.6.1. Here, all packets had a length of 600 bytes, including headers.

Table 4.4. End-to-end delay performance measured in DARTnet (topology of Fig. 3.1)

	path length							
	1		2		3		4	
traffic	mean	99.9%	mean	99.9%	mean	99.9%	mean	99.9%
FIFO	18.6	370	57.6	529	73.5	590	87.1	608
HL (200 ms)	26.3	515	59.9	632	59.9	632	81.4	661
HL (400 ms)	26.4	500	56.9	584	65.9	553	74.7	599

The end-to-end delays (including transmission and propagation delays) of the streams reaching the node `bbn`, are summarized in Table 4.4. 'HL (x)' refers to hop-laxity scheduling with a laxity value of x. From the table, we conclude that the performance between FIFO and HL does not differ markedly, indicating that the performance gains attributable to delay-sensitive scheduling are outweighed by the additional processing costs and packet header overhead. It appears doubtful that other laxity values would yield significantly better results. It should be noted that

the goal of delay equalization has indeed been reached, by reducing the delay spread between long-haul and short-haul flows from 238 for FIFO to 99 ms for HL.

A third set of experiments used several lecture voice and video streams multiplexed at each source, confirming the basic results shown here. Further details are relegated to a technical report in progress.

In summary, the network measurements generally confirm the conclusions drawn from the simulation experiments discussed in Section 3.4, but indicate that the gains attributable to hop-laxity scheduling are further diminished by the additional processing and packet header overhead. Further studies are needed before it could be recommended to expend the computational effort needed to implement HL in routers.

4.8 Conclusion and Future Work

The experiments performed so far have opened a range of possibilities for future explorations. We start by summarizing in Section 4.8.1 how the BSD networking code could be modified to ease implementation of resource-control algorithms. This complements the suggestions offered in [117]. Based on our experience, suggestions on the appropriate use of simulations and network measurements are made in Section 4.8.2.

4.8.1 Supporting Non-FIFO Scheduling in BSD Kernels

The networking code within Berkeley Software Distribution Unix (BSD) and the commercial Unix systems derived from it (such as SunOS or DEC Ultrix) have the flexibility to incorporate new protocols at both the transport and network layer, such as ST-II or OSI-based protocols, as well as the integration of new devices, even without access to the operating system source code. However, the support for alternate process scheduling and resource control is weak. Adding the hop-laxity scheduler or resource-reservation methods requires modifying the network

kernel and device driver sources, which are often not available to the experimenter. For example, the enqueueing and dequeueing routines within the device drivers are actually macros, so that relinking cannot be used to replace them with modified versions.

The combination of different scheduling methods is even more difficult since there is no standardized way of providing the necessary information to the scheduling routines. It would, however, be relatively easy to add a fairly general facility that would allow easy experimentation with a variety of scheduling methods. First, each network interface definition (contained in the `ifnet` structure defined in `net/if.h`) should have function pointers to the enqueueing and dequeueing routines. For the other layers, hooks should be added to the `protosw` structure (defined in `sys/protosw.h`) to access an enforcement or translation function that would do the mapping from protocol-specific scheduling information to the generic data structures understood by the device-level schedulers.

Less readily solved is the question of how to encapsulate the necessary per-packet and per-association "out-of-band" scheduling information. As described earlier, in our implementation, we simply mark special mbufs as containing the scheduling information within the first few bytes of their data area. This works reasonably well as long as there is enough space so that no additional mbuf has to be prepended. Alternatively, a new mbuf could always be used, but then all device drivers would have to be aware that certain types of mbufs are not meant to be transmitted. Keeping scheduling information within the same mbuf chain as the actual packet data simplifies argument passing and queueing, but may require searching a number of mbufs for the desired type.

4.8.2 Simulation and Network Measurements

Clearly, any proposed protocol, queueing policy etc. must finally prove itself in real networks, with realistic traffic and under the software and hardware constraints

of real routers and workstations. The experiments conducted on DARTnet highlight some of the difficulties in conducting performance measurements in a real network:

Kernel programming is tedious. Changing the kernel, even if all sources are available, is far more tedious and time consuming than programming a simulation. Debugging support for Unix kernels is rather limited, so that in many cases the cause for failure has to be divined by deciphering optimized assembly code. This is particularly true if the hardware used in the network is not available locally, as was the case for the work reported here. We used a tracing facility with timestamps to keep track of significant events as a packet crossed a router. As an example, adding time stamps recording progress along the path traversed by a packet is trivial in a simulation program, requires extensive programming effort and incurs nonnegligible delays when translated to IP packet handling code.

Measurements affect the outcome. Any timing measurements delays packet transmission and may reduce channel utilization. Packet receiver processing may impose additional delays.

Clocks are not synchronized. Even though DARTnet runs a state-of-the-art clock synchronization protocol, clock differences of several milliseconds, subject to drift, cannot be ruled out, thus making accurate delay measurements difficult. (It is possible to get statistics about actual clock differences from the clock daemon, but the effort is substantial.)

A channel is never idle. Even though DARTnet is dedicated to experiments, there are a number of services that need to be running during many experiments, such as clock synchronization and the multicast group management protocol (IGMP). Fortunately, the background traffic is usually insignificant.

Experiments are hard to reproduce. Since the traffic is generated by a non-real-time operating system, two experiments never produce the same packet timing. The disturbances introduced by background traffic can usually be ignored.

Network measurements generally take longer to set up, run and evaluate than equivalent system experiments. In our experience, the five-node network mapped onto DARTnet could be simulated in significantly less time than it took for the actual network experiment, even though five processors were contributing during the network experiment. Also, trace and statistics files have to be gathered after each experiment.

The topology is very restricted. Although topology constraints did not impact our work significantly, the current DARTnet topology has no alternate paths to a destination, thus making routing experiments difficult.

Thus, network measurement experiments should only be undertaken after extensive simulation experiments. However, limited network experiments can be useful in deriving information about computational overhead that can flow back into realistic simulation experiments.

CHAPTER 5

DISTRIBUTION OF THE LOSS PERIOD FOR QUEUES WITH SINGLE ARRIVAL STREAM

5.1 Introduction

For soft real-time communication systems, the packet loss rate rather than the average packet delay becomes the critical performance measure. Interactive voice, video and distributed measurement and control are examples of such systems that impose delay constraints, but also show a certain tolerance for lost packets. Most previous studies of real-time systems only measure loss as a time-average fraction of missing packets [5, 30, 58, 152–154]. In order to judge the "quality" of loss-prone communication, however, it is important to know not just how *many* packets are being lost, but also whether losses occur in clusters or randomly. The importance of accounting for correlated losses has long been recognized in specifiying acceptable performance of data circuits. "An errored second is declared when one or more bits in that second is found in error. [155]" This leads to the metric of the percentage of error-free seconds (EFS).

Papers on the replacement of lost packets in packet voice communication systems typically assume random, uncorrelated occurrence of packet loss [156, 157]; as we will show, this assumption might be overly optimistic. The work by Shacham and Kenney [158, 159] provides another example. They observed that loss correlation in a finite-buffer system could completely eliminate the advantage of three orders of magnitude predicted for forward-error correction under the assumption of independent (Bernoulli) losses.

The investigation of loss correlation has a long history in the context of bit errors in data communication [160], but has only recently attracted stronger interest in the area of packet networks.

A number of authors have quantified the influence of loss correlation for network performance. For example, it has been shown [161–164] how the throughput of go-back-N ARQ increases with positive loss correlation. Similar results for the N-packet transfer time and go-back-∞ and blast protocols are presented in [165]. All papers[1] assume a two-state Markovian error model with geometrically distributed run lengths for losses and successes, which will be seen to be appropriate for some, but not all queueing systems studied below. Thus, our results can be used in conjunction with the work cited here to directly predict ARQ performance.

In some simulation studies, cell loss correlation has been investigated [166, 167] and methods for the compensation of correlated cell loss have been proposed [159, 168]. Biersack [169] uses simulation to evaluate the effect of burst losses on the performance gains achievable by forward error correction.

In this chapter, we consider a single FCFS queue with infinite buffer in isolation in which an arriving customer[2] which reaches the server after waiting h or more units of time is considered "lost" (even though it still receives service). Chapter 6 will tackle more complex arrival statistics.

We characterize the stochastic properties of the *time-out loss period*, the uninterrupted interval of time during which the virtual work in the queue exceeds the given threshold h. Given this time-based quantity, we also arrive at measures of customer-based quantities such as the distribution of the number of consecutive customers which each spends more than an allotted time waiting for service. We show that the assumption that each customer is lost independently from the previous

[1] except for [163], which allows a more general Markovian error model

[2] The terms "customer" and "packet" will be used interchangeably.

one leads to a significant underestimation of the duration of such loss periods. Using elementary methods of probability, we also prove that for certain classes of queues the duration of the loss period is independent of the value of the threshold. We further show through numerical examples for other important types of queues that the influence of the threshold on this metric is minimal for interesting probabilities of loss. Our numerical results also indicate that the expected number of consecutively lost customers (for the same loss probability) varies by as much as 50% depending on the batch arrival distribution used. We also derive measures of the time between loss periods. Throughout the proposal, emphasis is placed on providing results that can be readily numerically evaluated.

A number of authors have investigated related aspects of the problem. Kamitake and Suda [170] consider a discrete-time queue in which traffic is generated by a changing number of active callers, with each active caller generating a packet in a slot according to a Bernoulli process. They compute the steady state loss rate for a given number of active callers and then consider the variation in the number of active callers in computing the amount of uninterrupted time from when this loss rate first exceeds a value ζ until it drops below another value η, with $\eta < \zeta$. Our work differs from [170] in that we directly characterize those periods of time in which arriving customers are lost, rather than characterizing loss as being "quasi-stationary" during periods of times during which the number of active sources remains constant.

Leland [171] mentions, but does not elaborate on measuring consecutive losses per connection in an ATM simulation experiment. Woodruff and Kositpaiboon [172] mention the desirability of specifying the probability and duration of periods of high cell loss rates. By rate conservation methodology, Ferrandiz and Lazar [173–175] develop a theoretical framework for studying loss correlation in queueing systems. They investigate the distribution of gaps, that is, consecutive losses, due to blocking and clipping (see below) in a multiclass $G/G/m/B$ queueing system; we discuss similarities and differences between our work and [173] in the following sections. In

general, our focus is more on deriving computable results, relying on well-known Markov chain methods.

Van Doorn [176] and Meier-Hellstern [177,178] characterize the overflow process from a finite Markovian queueing system. As pointed out earlier, [159] underlines the importance of taking loss correlations into account, but investigates their effect on forward-error correction only through simulation. A large body of literature analyzes the overflow process of blocked-calls-cleared systems, but the results do not seem directly applicable to our problem. Norros and Virtamo [179] use the average loss probability during congestion as an indicator of loss correlation. They assume that the total rate has a Gaussian distribution and that congestion occurs when the total instantaneous rate exceeds the channel capacity.

The chapter is organized as follows. After defining more precisely the systems and measures of interest in the section below, continuous-time queues are investigated in Section 5.2. In Section 5.3 we then apply similar methods to derive the corresponding measures for a discrete-time queue of interest in packet switching. We conclude by summarizing the work presented and pointing out some issues to be investigated. Additional details and examples can be found in a companion technical report [180].

5.2 Clip Loss in Continuous Time ($G/M/1$)

5.2.1 Performance Measures

This chapter focuses on a single-server queue, where customers are processed in the order of arrival. Customers that spend more than a deterministic, fixed amount of time h waiting for service are tagged as lost on leaving the queue, but are still served. (Ferrandiz and Lazar [173] refer to this as clipping loss.) This definition of loss is motivated by considerations of traffic with soft real-time constraints, where packets that are excessively delayed are worthless to the receiver. The loss as defined

here differs from that studied in our earlier work [152], where excessively delayed customers depart before occupying the server.

A *loss period* (LP) is an uninterrupted interval during which all arriving customers would experience a waiting time exceeding h. For infinite queues with FCFS service, the loss period equals the interval during which the virtual work in the queue is greater than the threshold h. Loss periods and busy periods are related in that a busy period is a special case of a loss period, with threshold value $h = 0$. Also, both busy periods and loss periods start with the arrival of a customer. They differ, however, in that a busy period ends with a departure of a customer, while the end of a loss period is not connected with a customer departure. A *noloss period* is the interval between two loss periods. For $h = 0$, noloss periods correspond to idle periods of the queue.

While the loss period is of independent interest, we are particularly concerned with measuring the number of consecutively lost customers, called *loss run* for brevity. Note that the number of customers arriving in a loss period is not identical to the number of consecutively lost customers. In particular, the first customer triggering a loss period, i.e., the customer that pushes the virtual work above h, is itself not lost. Thus, loss periods consisting of a single arrival do not contribute to customer loss. Note that there may also be several noloss periods interspersed between two customer losses if each of the loss periods separating the noloss periods consists of only the arrival triggering a loss period. Similar to loss runs, *success runs* denote the number of consecutive customer without any loss.

Fig. 5.1 depicts a virtual work sample path and the measures defined above. Time proceeds along the abscissa from left to right, while the ordinate shows the virtual work at a given instant. In the figure, arrivals that will be tagged as lost on departure are marked with $\otimes$, the others with $\odot$. The extent of loss periods (LP) and busy periods (BP) are indicated by horizontal bars. Bold arrows represent *initial jumps*, the amount of virtual work above h at the beginning of a loss period.

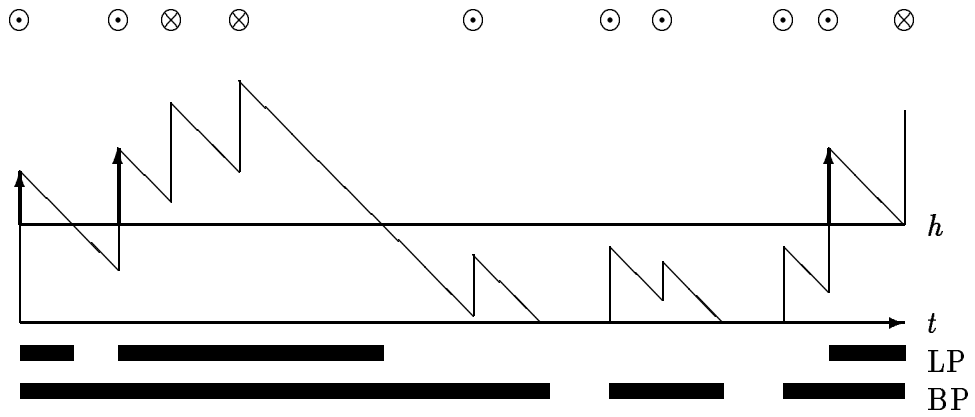

Figure 5.1. Virtual work sample path

The height of the vertical jumps indicates the amount of work brought to the system by an arriving customer. The unfinished work decreases at a unit rate as the server processes jobs.

Let us briefly touch upon other measures of loss behavior for queueing systems. For finite queues with m deterministic servers and bulk arrivals, at least m of the arriving packets are always served. For an individual source, the probability of consecutive losses depends on its position within the bulk, which might be fixed, uniformly distributed or a time-varying stochastic process, depending on the buffer management policy. This system was treated extensively by Li [181]. Li defines as blocking those states where the buffer is full prior to service completion.

We may also look at loss-correlation through a frequency-domain perspective. As an example, the first-order autocorrelation coefficient of intervals between losses was measured experimentally for a five-stage virtual circuit model with bounded waiting times. Using the von-Neumann statistic [182] as a robust estimator, it was found that the intervals between losses were essentially uncorrelated. Autocorrelation information might be useful in comparing different buffer management schemes or service disciplines, but cannot readily be used to predict the performance of packet reconstruction algorithms.

The stochastic properties of the loss period or consecutive customer losses can be quantified in the usual manner, for example through its distribution or its average duration, either in terms of time or the number of customers affected. Details are discussed in the next two sections for continuous and discrete-time queueing systems of interest.

As a first model, we investigate a system with general (not necessarily i.i.d.) arrival process of rate λ and exponential service with rate μ. (The service process does not have to be independent of the arrival process.) The buffer is infinite, so that only loss due to excessive delays in the queue occurs. All packets, regardless of whether they exceed the waiting time threshold or not, are served. (A system where late packets are dropped will be treated later.) The special case of Poisson arrivals, i.e., the $M/M/1$ system, will be treated in detail since it yields closed-form expressions for the measures of interest. The $M/M/1/\infty$ system was also investigated by Ferrandiz and Lazar [173] as a special case of their $G/G/m/B$ analysis. Their analysis seems considerably more involved, does not readily yield numerical results and does not make use of the simple connection to the busy period pointed out here. Our model is applicable, if only in approximation, to systems with variable packet sizes, for example the PARIS network [183] or a variation of the Knockout switch [184].

Throughout this section, we will illustrate our calculations through a running example consisting of an $M/M/1$ queue with arrival rate $\lambda = 0.8$, service rate $\mu = 1$ (and thus a load of $\rho = \lambda/\mu = 0.8$) and a clipping threshold of $h = 3$ so that $\alpha = 1 - \rho e^{(\lambda-\mu)h} = 43.905\%$ of all arrivals experience a delay of more than 3 (are "lost"). The methods presented below apply equally well at lower loss probabilities; we have chosen this (impractically) high loss to simplify simulation verification of our results through simulation.

5.2.2 *Distribution of the $G/M/1$ Initial Jump and Loss Period*

As pointed out above, a loss period commences when an arrival causes the virtual work to cross the threshold h from below. In order to establish the distribution of the loss period, the following lemma describes the distribution that governs the initial jump, i.e., the virtual work immediately after the arrival of the first customer in a loss period.

Lemma 1 *For a $G/M/1$ queue, the initial jump has the same distribution as the service time and does not depend on the threshold h.*

PROOF Let the random variable J represent the height of the initial jump and $f_J(j)$ its density. The density $f_J(j)$ can be expressed through conditioning, where we define V as the virtual work just prior to the arrival of the customer whose new work pushes the virtual work above h. A denotes the arriving (new) work.

$$f_J(j) = \int_0^h P[V = v | V \leq h] \cdot P[A = h + j - v | A \geq h - v] \, dv \tag{5.1}$$

The second conditional probability can be rewritten as

$$\frac{\mu e^{-\mu(h+j-v)}}{e^{-\mu(h-v)}} = \mu e^{-\mu j},$$

which follows immediately from the memorylessness property of the exponential distribution.

Now we can rewrite the jump density as

$$\begin{aligned} f_J(j) &= \mu e^{-\mu j} \int_0^h \frac{P[v = y \cap y \leq h]}{P[v \leq h]} \, dy \\ &= \mu e^{-\mu j} \frac{1}{P[v \leq h]} \int_0^h P[v = y] \, dy \\ &= \mu e^{-\mu j} \frac{1}{P[v \leq h]} P[V \leq h] \\ &= \mu e^{-\mu j}. \end{aligned}$$

This shows that the jump density is indeed exponentially distributed. □

Given this property of the initial jump, the following theorem follows immediately:

Theorem 5.1 *In a $G/M/1$ queueing system, a loss period is stochastically identical to a busy period. The loss period distribution is independent of the threshold h. For $G/G/1$ queues, a loss period is stochastically identical to a busy period with special first service.*

For the $G/G/1$ case, note that busy periods with special (or exceptional) first service are covered by Wolff [185, p. 392-394].

The independence of the loss behavior from the threshold recalls a similar observation made by Li [181] regarding the buffer overflow process in a packet voice system. There, the time spent in the overload state was found to be independent of the buffer size.

Let the random variable L denote the time duration of a loss period. Then, given Theorem 5.1, busy and loss periods for the $M/M/1$ queue have the density [186, p. 215]

$$f_L(y) = \frac{1}{y\sqrt{\rho}} e^{-(\lambda+\mu)} I_1[2y\sqrt{\lambda\mu}]$$

and mean

$$E[L] = \frac{1}{\mu - \lambda} = \frac{1}{\mu}\frac{1}{1-\rho},$$

where $I_1(y)$ is the modified Bessel function of the first kind of order one. The cumulative distribution function is best computed by numerical integration, as there are no suitable closed-form expression for the integral.

For non-$G/M/1$ queues, the computation of the initial jump can be difficult. In general, the relationship

$$f_J(j) = \int_0^h \frac{w(v)b(h+j-v)}{W(h)(1-B(h-v))}\, dv$$

holds, where $b(x)$, $w(x)$, $B(x)$ and $W(x)$ are the densities and distributions of arriving work and the virtual work seen by the arrival, respectively.

However, some stochastic ordering results for busy periods with special first service and loss periods may be of interest.

Lemma 2 *The loss period for a $G/GI/1$ queue is stochastically longer than (shorter than) a busy period if the initial jump is stochastically larger (smaller) than a service duration.*

PROOF We prove the first, non-parenthesized part; the other proceeds similarly. Let the random variables X and Y denote a regular service time and an initial jump, respectively. From the coupling theorem (see [187, Proposition 8.2.2]), we know that if $Y \geq_{\text{st}} X$, then there exist random variables $\tilde{X}$ and $\tilde{Y}$, with the same distributions as X and Y, such that $\tilde{Y} \geq \tilde{X}$, i. e., $P(\tilde{Y} \geq \tilde{X}) = 1$. Without changing the duration of the loss period, we preempt the first customer after $\tilde{X}$ and then complete its remaining service time, $\tilde{Y} - \tilde{X}$, at the conclusion of the loss period. Denote the sum of the service periods of the remaining customers in the busy or loss period by S and the extension of the loss period caused by arrivals during the $\tilde{Y} - \tilde{X}$ segment as E. Since

$$\tilde{X} + S > a \Rightarrow \tilde{X} + S + (\tilde{Y} - \tilde{X}) + E > a,$$

we write

$$P[\tilde{X} + S + (\tilde{Y} - \tilde{X}) + E > a] \geq P[\tilde{X} + S > a],$$

from which the stochastic inequality for the coupled loss and busy periods follows. Finally, we uncondition to make the result apply to any busy and loss period.

Alternatively, we can argue by using [187, Example 8.2(a)], where it is shown that

$$f(Y_1, \ldots, Y_n) \geq_{\text{st}} f(X_1, \ldots, X_n)$$

for any increasing function f if $Y_i \geq_{\text{st}} X_i$ and given that $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_n$ are independent. Clearly, the length of the busy period is an increasing function (the sum) of the service periods that constitute it. □

Using this lemma, it is easy to show the following general relation between loss periods and service times.

Theorem 5.2 *If the service time has decreasing failure rate (DFR), the loss period is stochastically longer than a busy period. Conversely, for service times with increasing failure rate (IFR), the loss period is stochastically shorter than a busy period.*

PROOF Define X_t as the additional life of X from t onward, given that $X \geq t$. In [187, Proposition 8.1.3], it is shown that iff X is DFR, then $X_t \geq_{\text{st}} X$ and iff X is IFR, then $X_t \leq_{\text{st}} X$. This is true for any value of t.

Let the conditional random variable X_t denote the initial jump initiating a random loss period, given that the amount of service time needed to reach from the virtual work to h equals t. Let X' be the unconditional initial jump and X the service time. We show the proposition for the IFR case; it follows for the DFR case by reversing the relational operators.

If X is IFR, then $X_t \leq_{\text{st}} X$, so that

$$P[X_t < X] \geq P[X < x].$$

X' can be computed from X_t by removing the condition:

$$P[X' < x] = \int_0^\infty P[X_t < x]\, dF_x(t)$$

Thus,

$$P[X' < x] = \int_0^\infty P[X_t < x]\, dF_x(t) > \int_0^\infty P[X < x]\, dF_x(t) = P[X < x].$$

In other words,

$$X' \leq_{\text{st}} X,$$

from which Lemma 2 yields the proposition. □

Note that the initial jump does not equal X_t for any one t. As an example, consider the $M/D/1$ queue with unit service time. The survivor function of the residual service time is given by

$$\bar{F}_t(a) = \frac{\bar{F}(t+a)}{\bar{F}(t)} = \begin{cases} 1 & \text{for } t+a < 1 \\ 0 & \text{otherwise} \end{cases}$$

for $t \leq 1$ and undefined otherwise. Thus, the density is zero everywhere except at the point $t + a = 1$.

The initial jump, on the other hand, is given by

$$P[J = j] = P[V = h - 1 + j | V \leq h].$$

A closed-form expression for the distribution of the virtual wait for the $M/D/1$ is not available, so that the above equation cannot be further simplified. However, Theorem 5.2 tells us that the loss period will be stochastically shorter than the busy period. In particular, the mean loss period will be less than $1/(1-\rho)$. The simulation data in Table 5.1 shows that the expected initial jump and, consequently, the expected loss period depend only weakly on h for "interesting" values of h. It is conjectured that this property holds for general queueing systems. A possible justification can be sought in the exponential form of Kingman's approximation for the tail of the waiting time distribution [188, p. 45].

Table 5.1. Expected initial jump and expected loss period for $M/D/1$ queue

h	initial jump	loss period
0.0	1.000	5.00
0.2	0.815	4.07
0.5	0.588	2.93
1.0	0.567	2.82
2.0	0.483	2.39
3.0	0.467	2.32
5.0	0.465	2.31

5.2.3 Consecutive Customers Lost

While the duration of a loss period is distributed like the duration of a busy period, we recall from Section 5.2.1 that the number of consecutive customers lost does not have the same distribution as the number of customers in a busy period. Defining C_C and C_B as the number of consecutively lost customers and the number of customers in a busy period, respectively, we have

$$P[C_C = n] = \frac{P[C_B = n+1]}{P[C_B > 1]},\ n > 0$$

where $P[C_B > 1]$ is the probability that the busy period contains more than one customer.

Let us apply these results to the $M/M/1$ queue. With $P[C_B = n]$ given by [186, Eq. (5.157)]

$$P[C_B = n] = \frac{1}{n}\binom{2n-2}{n-1}\rho^{n-1}(1+\rho)^{1-2n},\ n > 0$$

we compute

$$P[C_B > 1] = 1 - P[C_B = 1] = 1 - \frac{1}{1+\rho} = \frac{\rho}{1+\rho}.$$

Thus,

$$P[C_C = n] = \frac{1}{n+1}\binom{2n}{n}\frac{\rho^{n-1}}{(1+\rho)^{2n}},\ n > 0.$$

Note that this result differs markedly from the geometric distribution postulated by Ferrandiz [173, Corollary 5.3].

Since the average number of customers per busy period is $1/(1-\rho)$, we have that for the $M/M/1$ queue the average number of consecutive customers lost is

$$E[C_C] = \frac{1}{P[C_B > 1]}\left(\frac{1}{1-\rho} - 1\right) = \frac{1+\rho}{1-\rho}. \tag{5.2}$$

This result differs markedly from that obtained under the assumption that losses occur independently as Bernoulli events with the time-average loss probability α.

In that case, the conditional probability mass function (pmf), given one loss, for the number of consecutive losses would be distributed geometrically as

$$P[\hat{C_C} = n] = \alpha^{n-1}(1-\alpha)$$

with an average value of $E[\hat{C_C}] = 1/(1-\alpha)$. For our running example, the independence assumption leads one to conclude that a loss period consists of 1.78 customers on average, while our analysis above shows that the actual number for this system is 9. Thus, customer losses are far more clustered than the assumption of independent losses would suggest.

An additional characterization of loss periods is provided by the *conditional probability of packet loss* given that the previous packet was lost, denoted here by r. It is directly related to the average loss run length, $E[C_C]$, through [173, eq. (5.1)]

$$\begin{aligned} E[C_C] &= \frac{1}{1-r} \approx 1 + r + r^2 + \ldots, \\ r &= 1 - \frac{1}{E[C_C]}. \end{aligned}$$

For the $M/M/1$ case,

$$r = \frac{2\rho}{1+\rho}.$$

The clustering of losses in a queueing system is naturally also reflected in this measure. For our $M/M/1$ example, the conditional loss probability equals 0.89, while the assumption of independent losses would result in a conditional loss probability r equal to the loss probability α, which evaluates to 0.44 for our running example.

5.2.4 Distribution of Noloss Period

The distribution of the time between loss periods is more difficult to determine. This interloss time comprises the interval between the time the virtual work W drops below h from above and the first time instance it rises above this mark again, i.e., the event $\min\{t : W(t) = h | W(0) = h\}$. The sample path with respect to time t of

this stochastic process is continuous in time and right-continuous in state, with drift of rate t and jumps of exponential height at Poisson intervals. The difficulty appears since the process is "sticky" at the zero line, with dwell time corresponding to the interarrival (or queue idle time) distribution. We are interested in the distribution of the time to absorption of this process at the $W = h$ barrier.

To make the problem tractable, we have assumed a Markovian arrival process; otherwise the duration of the noloss period would depend on the time of the last arrival during the preceding loss period. Thus, the computation of this section will be limited to the $M/G/1$ model.

Aspects of this problem or approximations of it appear in a number of applied stochastic models [189]. In *collective risk theory* [190] the insurance company starts out with some fixed capital. This capital increases through premiums at a constant rate and decreases (or increases) by claims occurring at Poisson instants. Of interest is the time until the capital reaches zero, that is, the company is ruined. To model noloss periods, the capital would represent the mirror image of the virtual work, with an initial value of zero. However, the model does not allow for the fact that the state cannot exceed h (idle system). Thus, it would tend to overestimate the duration of the noloss period and be most suitable for heavy traffic where the idle period is short. It would give exact results, however, for $t < h$ since the system cannot have reached h by that time.

We select a model involving an approximation that is based on the so-called Moran dam model [191, 192] [193, p. 336f] [194, p. 200]. In this model, the water content of a dam or reservoir is represented by a continuous or discrete-state, discrete-time homogeneous Markov process. For reasons of computability, we choose a discrete-state representation, yielding a discrete-time Markov chain (DTMC). Time is discretized in quantities of τ, a fraction of the service time, and the Markov chain tracks the virtual work W_n in the queue at epochs of integer multiples of τ. Thus, the Markov chain has $k = h/\tau + 1$ states with values from the set

$\{0, 1/\tau, 2/\tau, \ldots, h\}$. At the end of each discretization interval, at $(n\tau)^-$, the virtual work, if positive, decreases by one unit, reflecting the fact that the virtual work decreases at unit rate. Arrivals bring in an amount of work X_n, again in multiples of τ, and occur just after the beginning of the discretization interval, at $(n\tau)^+$. The noloss period ends as soon as the virtual work reaches h or state k. We model this by making state k an absorbing state and compute the duration of the noloss period as the time to absorption into state k. Given this description of the DTMC, we can write the state evolution recursively as

$$W_{n+1} = \min(k, W_n + X_n) - \min(1, W_n + X_n).$$

Let us define a_k as the probability that k units of work of size τ arrive. Also, denote the complementary cumulative distribution function g_j as

$$g_j = \sum_{i=j}^{\infty} a_j = 1 - \sum_{i=0}^{i-1} a_j.$$

The state transition matrix follows readily:

$$\mathbf{P} = \left[\begin{array}{c|ccccc} & 0 & 1 & \ldots & k-1 & k \\ \hline 0 & a_0 + a_1 & a_2 & & a_k & g_{k+1} \\ 1 & a_0 & a_1 & & a_{k-1} & g_k \\ \ldots & & & & & \\ k-1 & 0 & 0 & & a_1 & g_2 \\ k & 0 & 0 & & 0 & 1 \end{array}\right] \tag{5.3}$$

The last row stems from the fact that state $k-1$ is absorbing. The state transition probabilities are computed as

$$a_j = P[\tau j \leq X \leq \tau(j+1)], \tag{5.4}$$

where X is the amount of work arriving in a slot. We know that the accumulated work from n arrivals in a $G/M/c$ system is Erlang-n distributed with density

$$f(x) = \frac{\mu n(\mu n x)^{n-1}}{, (n)} e^{-\mu n x}.$$

and cumulative distribution function $F(x) = P(n, \mu x)$, where $P(a, x)$ is the normalized incomplete gamma function

$$P(a,x) \triangleq \frac{\gamma(a,x)}{,(a)} = \frac{1}{,(a)} \int_0^x e^{-t} t^{a-1}\, dt.$$

The distribution of arriving work needed in evaluating Eq. (5.4) is, hence, given by

$$\begin{aligned} P[X < x] &= \sum_{n=0}^{\infty} P[X < x | n \text{ arrivals}] P[n \text{ arrivals}] \\ &= e^{-\lambda\tau} + \sum_{n=1}^{\infty} P(n, \mu x) e^{-\lambda\tau} \frac{(\lambda\tau)^n}{n!}. \end{aligned}$$

The distribution of the time to absorption can be computed in two basic ways. First, since the probability of having been absorbed by the nth transition is simply the probability that the DTMC is in state k after n steps, we can use the basic state probability equation in its recursive or matrix-power form,

$$\pi^{(n)} = \pi^{(n-1)}\mathbf{P} = \pi^{(0)}\mathbf{P}^n, \tag{5.5}$$

where $\pi^{(0)} = (0, 0 \ldots 0, 1, 0)$, i.e., $k-1$ is the initial state. The matrix power in Eq. 5.5 can be evaluated as a special case of the general relationship for any functional f of a matrix, given by $f(\mathbf{P}) = \mathbf{V} f(\Lambda) \mathbf{V}^{-1}$, where $\mathbf{V}$ is the matrix of eigenvectors of $\mathbf{P}$ and the function f is applied element-by-element to the diagonal matrix of eigenvalues Λ [195, p. 8]. This eigenvalue approach may be more accurate for large values of n.

The other alternative defines $f_{il}^{(n)}$ as the probability that the system *first* enters state l after n steps, given the initial state is i. To use the first-passage formulation, the matrix $\mathbf{P}$ has to be returned to its recurrent form by replacing the last row with $(0, \ldots, 0, a_0, g_1)$. It is readily seen that this first-passage probability mass function is recursively defined for all transition matrices as

$$f_{il}^{(1)} = P_{il} \text{ for } i = 0, 1, \ldots, \tag{5.6}$$

$$f_{il}^{(n)} = \sum_{j, j \neq l} P_{ij} f_{jl}^{(n-1)} \tag{5.7}$$

Sample calculations showed that state equations, matrix computations and the approach using f_{il} yield the same numerical result to within four significant

figures, indicating that roundoff errors are not a serious problem here. Also, the computational effort is about the same.

The discretization error incurred by using a particular value of τ can be estimated by computing the expected duration of the noloss period. Since the fraction of packets lost, α, is related to the expected loss period $E[L]$ and the expected noloss period $E[N]$ by[3]

$$\alpha = \frac{E[L]}{E[L] + E[N]}, \tag{5.8}$$

the expected noloss period can be computed as

$$E[N] = E[L]\left(\frac{1}{\alpha} - 1\right).$$

Given the DTMC approximating the virtual work process, the expected noloss period (equivalent to the time to absorption) can be computed as

$$E[N] = \left(\frac{1}{\pi'_{k-1}} - 1\right)\tau \tag{5.9}$$

where π'_{k-1} is the *steady-state* probability that the return process corresponding to the DTMC is in state $k-1$. The transition probability matrix of the return process is derived from $\mathbf{P}$ defined in Eq. 5.3 by replacing the last row with all zeros, except for a one in column $k-1$. This relationship is derived in [196, p. 112, Problem 3] for the case of two absorbing states, but the result generalizes readily to any number of absorbing states (see also [197, p. 103]).

For our example, the exact value of $E[N]$ is 6.388. For the discretization with $\tau = 0.1$, Eq. (5.9) yields a value of 6.109, which improves to 6.237 and 6.327 for $\tau = 0.05$ and $\tau = 0.02$, respectively.

5.2.5 Customers per Noloss Period

It appears difficult to derive an expression for the distribution of the number of customers in a noloss period. The expected value, however, is readily available

[3]Replacing α by the load, ρ, noloss and loss periods by idle and busy periods yields the well-known relation for busy cycles, again underlining the strong connection between loss and busy periods.

since the average number of customer arrivals during loss periods, $E[C_C]$, and noloss periods, $E[C_N]$, are related in a similar fashion as the respective measures of their time duration as given in Eq. (5.8), yielding

$$\begin{aligned} \alpha &= \frac{E[C_C]}{E[C_C] + E[C_N]} \\ E[C_N] &= E[C_C]\left(\frac{1}{\alpha} - 1\right). \end{aligned}$$

The difficulty in determining the distribution arises from the fact that noloss periods do not "see" the same Poisson arrival process with rate λ as a random observer, just as the the arrival rate measured during busy periods is higher than the arrival rate measured over all customers. Thus, the conditional probability of the number of arrivals given a noloss period duration cannot be readily computed.

All results were confirmed by simulation experiments. Even for the relatively coarse quantization of $\tau = 0.1$, the noloss duration agrees quite closely with the approximate analysis. Details can be found in [180].

5.3 Clip Loss in Discrete-time Systems

We now turn our attention to a queueing model that is commonly used for packet switches and ATM-type networks [198, 199]. In this model, time is discretized, with deterministic service (of duration $\tau = 1$) and batch arrivals, which, in many instances, allow somewhat simpler solutions than their continuous-time counterparts. Batches arrive at the beginning of a slot of unit width, while at most one customer departs at the end of a slot. (Hunter [200, p. 193] refers to this as an early arrival system.) We allow the batch size, A, to have a general distribution, but require the batch sizes to be independent from slot to slot and independent of the state of the queue itself.

We will say that batch sizes are geometrically distributed if their probability mass function (pmf) is $a_n = pq^n$, with $q = 1 - p$ and an average batch size of $\rho = q/p$. We will also cover the Poisson distribution with mean ρ,

$$a_n = \frac{e^{-\rho}\rho^n}{n!} \text{ for } n \in [0, \infty),$$

and the binomial distribution with mean $\rho = \nu p$,

$$a_n = \begin{cases} \binom{\nu}{n} p^n q^{\nu-n} & \text{for } n \in [0, \nu] \\ 0 & \text{otherwise} \end{cases} .$$

While numeric solutions are possible for general batch size distributions, a queue with geometric batch sizes, i.e., the system $D^{[\text{Geo}]}/D/1$, will be shown to exhibit the property that busy periods and loss periods are equal in distribution, analogous to the result described in Section 5.2 for the continuous-time $GI/M/1$ system. Also, restricting batch sizes to be geometrically distributed will yield closed-form expressions for many distributions of interest. Due to the close relationship between busy and loss periods, we will investigate busy periods in some detail in Section 5.3.1, followed by derivations of the properties of loss and noloss periods in sections 5.3.2 and 5.3.3, respectively.

5.3.1 The Busy and Idle Period

In discrete-time systems, we define the beginning of a *busy period* to be the time that the first customer in a batch experiences no waiting, i.e., finds the server free. Note that, unlike in a continuous time system, two successive busy periods need not have an intervening period of time during which the server is idle. This occurs if the last customer in a busy period, departing in (n^-, n), is immediately followed by one or more new arrivals in (n, n^+). Thus, the first customer in that batch enters service immediately, starting a new busy period, while the server experiences no idle slot. Later, we will discuss the composite busy period which encompasses time intervals without server idling, consisting of one or more busy periods.

Let us return now to the busy period and compute its distribution. Because each customer occupies the server for one slot, the busy period length is equal to the number of customers served during the busy period in the discrete-time case. For geometric batches[4], we can compute the number of customers in a busy period by making use of Takács combinatorial arguments [201, p. 102f], [186, p. 225f]. Let B be the number served in a busy period and $\tilde{A}_n$ the number of arrivals during the service times of customers 1 through n, where these customers may belong to one or more busy periods.

The probability mass function of the number served in a busy period is given by [186, Eq. (5.166)]

$$P[B = n] = \frac{1}{n} P[\tilde{A}_n = n - 1].$$

For the case of deterministic service and batch size distribution a_n, the probability on the right-hand side can be readily derived:

$$P[\tilde{A}_n = n - 1] = P[n - 1 \text{ arrivals in } n \text{ slots}] = a_{n-1}^{n*}.$$

Here, a_n^{n*} denotes the n-fold convolution of a_n with itself. For the case of geometrically distributed batches, the convolution becomes the negative binomial or Pascal distribution with probability mass function

$$a_n^{r*} = \binom{r + n - 1}{n} p^r q^n.$$

Thus, the busy period for geometric batches is distributed according to

$$P[B = n] = \frac{1}{qn} \binom{2n - 2}{n - 1} (qp)^n \tag{5.10}$$

[4]The ensuing development requires that the work arriving during the service of each customer be i.i.d.. For batch arrivals, the arrivals that occur while the first customer is being serviced consist of the batch starting the busy period minus one. The distribution of this "shortened" batch has the same distribution as a regular batch only if batches are geometrically distributed. For other batch distributions, the following calculations can serve as an approximation.

$$= \frac{1}{n}\binom{2n-2}{n-1}\rho^{n-1}(1+\rho)^{1-2n}. \tag{5.11}$$

The last transformation makes use of the fact that the system load ρ is related to the batch distribution parameter p through $p = 1/(1+\rho)$. We recognize the last expression as the distribution of the number of customers served in an $M/M/1$ busy period [186, p. 218].

The z-transform of the number served in a busy period, $B(z)$, follows from the $M/M/1$-derivation [186, p. 218]:

$$B(z) = \frac{1+\rho}{2\rho}\left[1 - \sqrt{1 - \frac{4\rho z}{(1+\rho)^2}}\right] = \frac{1-\sqrt{1-4pqz}}{2q} \tag{5.12}$$

The expected number served (and arriving) in a busy period can be computed by evaluating an infinite series using Eq. (5.12) or directly copying the $M/M/1$ result:

$$E[B] = \sum_{n=1}^{\infty} nP[B=n] = \frac{p}{\sqrt{1-4pq}} = \frac{1}{1-\rho}.$$

The *idle period* I for general batch sizes is geometrically distributed with density $P[I=n] = a_0^n(1-a_0), n \geq 0$. Recall that a_0 is the probability of a batch having zero members. Thus, the average idle period is given by $a_0/(1-a_0)$. For geometric batch sizes, $a_0 = p$ and thus an idle period last an average of $1/\rho$ slots.

In contrast to the continuous-time case, an idle period may have zero duration. This occurs if a new batch arrives immediately after the last customer of a busy period departs. We will call a period of continuous server occupancy spanning several busy periods a *composite busy period* and identify random variables associated with it by a tilde. A composite busy period consists of ϕ busy periods with the geometric probability $P[\phi = k] = (1-a_0)^{k-1}a_0$. In the z-transform domain, the number of customers in the composite busy period, $\tilde{B}$, is determined as $\tilde{B}(z) = \phi(B(z))$, where

$$\phi(z) = \frac{pz}{1-qz}$$

is the probability generating function of the number of busy periods constituting a composite busy period.

For geometric batches, $\tilde{B}(z)$ can be expanded using Eq. (5.12):

$$\begin{aligned}\tilde{B}(z) &= \frac{p}{q}\frac{1-\sqrt{1-4pqz}}{1+\sqrt{1-4pqz}} \\ &= \frac{p}{q}\frac{1-2\sqrt{1-4pqz}+1-4pqz}{4pqz} \\ &= \frac{1}{zq}\left[\frac{1-\sqrt{1-4pqz}}{2q}\right]-\frac{p}{q}\end{aligned}$$

The bracketed fraction is recognized as $B(z)$ and thus $\tilde{B}(z)$ can be inverted easily, yielding

$$\begin{aligned}P[\tilde{B}=n] &= \frac{1}{q}\frac{1}{n+1}\binom{2n}{n}\frac{\rho^n}{(1+\rho)^{2n+1}} \\ &= \frac{1}{n+1}\frac{(2n)!}{(n!)^2}\frac{\rho^{n-1}}{(1+\rho)^{2n}} \qquad (n \geq 1).\end{aligned}$$

The expected number of customers in a composite busy period (and its expected duration in slots) is seen to be

$$E[\tilde{B}] = E[B]\cdot E[\phi] = \frac{E[B]}{a_0},$$

in general or

$$E[\tilde{B}] = \frac{1+\rho}{1-\rho}$$

for geometric batches.

5.3.2 The Loss Period

A *loss period* begins when one or more customers arriving in a batch see h or more customers already in the system (in other words, if their wait is equal to or greater than h.) Thus, a busy period is (again) a special case of a loss period with h having the value zero. A loss period ends when there are h or fewer customers left after a departure. Just as discussed above for the case of busy periods, a loss period may be followed immediately by another loss period. This occurs if the number of customers reaches h at some point n, i.e., the loss period ends, and a

batch arrives in (n, n^+), starting a new loss period. An uninterrupted interval where the number of customers in the system just prior to an arrival never drops below h (or, equivalently, where the occupancy after the arrival instants remains above h) will be referred to as a *composite loss period*. Clearly, it consists of an integral number of loss periods.[5] The random variables L and $\tilde{L}$ represent the loss period and composite loss period, respectively, while the random variable V represents the occupancy on the slot boundary, equivalent to the work in the queue seen by an arriving batch (virtual work). Because of the deterministic service time and the slotted arrivals, duration and customer count properties are the same, i.e., a loss period of l slots leads to l consecutive losses.

Fig. 5.2 depicts an example of a sample path showing a composite loss period made up of two loss periods (LPs) for a threshold of $h = 3$.

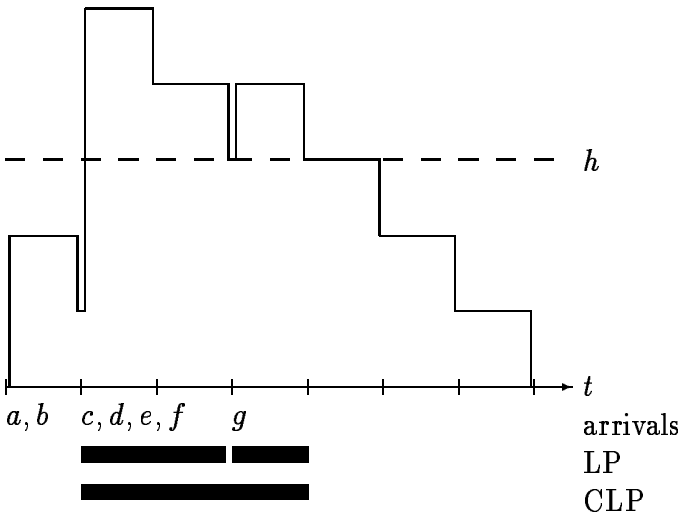

Figure 5.2. Loss periods in discrete time ($h = 3$)

Just as in the continuous-time case, we are interested in determining the distribution of the initial jump J, that is, the work load or, equivalently, the number of

[5]However, unlike in the $M/M/1$-case, we do not have to factor out the single-customer busy periods.

customers that begins a loss period. Fig. 5.2, for example, shows two initial jumps, one occurring at the arrival of batch $\{c \ldots f\}$, with a height of two, and the second at the arrival of $\{g\}$, with a height of one.

Lemma 3 *The initial jump J into a loss period has the distribution*

$$P[J=j] = \frac{\sum_{a=j}^{\min(h+j,\nu)} P[V_h = h+j-a]P[A=a]}{\sum_{a=1}^{\min(h,\nu)} P[V_h > h-a]P[A=a] + P[A>h]},$$

where the distribution of the batch size random variable A is zero outside the range $[0,\nu]$. The random variable V_h represents the occupancy seen by an arriving batch when the occupancy is less than or equal to h. The conditional occupancy distribution seen by an arriving batch is defined as

$$P[V_h = v] \triangleq \frac{P[V=v]}{P[V \le h]}.$$

PROOF We sum the probabilities of all events leading to a jump of size j (given that a jump occurred), noting that the random variables A and V are independent. Thus, we have for jumps into loss periods,

$$\begin{aligned}
&P[J=j] \\
&= \sum_{v+a=h+j} P[V_h = v \cap A = a | A + V_h > h] \\
&= \frac{\sum_{v+a=h+w} P[V_h = v \cap A = a \cap A + V_h > h]}{P[A+V_h > h]} \\
&= \frac{\sum_{v+a=h+w} P[V_h = v]P[A=a]}{\sum_{v+a>h} P[V_h = v]P[A=a]} \\
&= \frac{\sum_{a=1}^{\min(h+j,\nu)} P[V_h = h+j-a]P[A=a]}{\sum_{a=1}^{\min(h,\nu)} P[V_h > h-a]P[A=a] + P[A>h]}.
\end{aligned}$$

□

Among loss periods, we have to distinguish those that form the first loss period in a composite loss period. While an arriving batch that starts a loss period may

see between 0 and h in the system, a batch that initiates a composite loss period, i.e., the initial jump conditioned on the fact that the loss period is the first in a composite loss period, can see at most $h-1$ customers. Thus, the following lemma provides a separate expression for the jump into a composite loss period, $\tilde{J}$.

Lemma 4 *The initial jump into a composite loss period $\tilde{J}$ has the distribution*

$$P[\tilde{J}=j] = \frac{\sum_{a=j+1}^{\min(h+j,\nu)} P[V_{h-1}=h+j-a]P[A=a]}{\sum_{a=2}^{h} P[V_{h-1}>h-a]P[A=a] + P[A>h]}.$$

The derivation of $P[\tilde{J}]$ proceeds as in the proof of the previous lemma.

Finally, initial jumps of all but the first loss period within a composite loss period always start at h. Since the queue occupancy seen by an arriving batch and the batch size itself are independent, the jump into these "interior" loss periods is distributed like a regular non-zero batch.

In close parallel to the continuous-time case, the memorylessness property of the geometric distribution makes the distribution of both types of initial jump a shifted version of the batch distribution, independent of h. We formulate this more precisely in lemma 5.

Lemma 5 *For the $D^{[\mathrm{Geo}]}/D/1$ queue, the initial jump J and the initial jump into composite loss periods $\tilde{J}$ are distributed like non-zero batches, with pmf $P[J=j] = P[\tilde{J}=j] = pq^{j-1} = \rho^{j-1}/(\rho+1)^j$.*

The proof can be found in [180].

Given the characteristics of the initial jump, the length of a (composite) loss period is stochastically identical to the length of a (composite) busy period with a special first service given by the initial jump distribution. By the previous lemma, loss periods in a system with geometric batch arrivals have the same distribution as regular busy periods of the same system. Since the members of a batch that initiate a loss period also experience delays of at least h, the number of customers lost in

a loss period and the number served in a busy period with the above-mentioned special first service are stochastically identical as well.

For general batch-size distributions, the probabilities for loss periods of length one can easily be written down exactly:

$$\begin{aligned} P[L=1] &= P[J=1], \\ P[\tilde{L}=1] &= P[J=1]a_0. \end{aligned}$$

Closed-form expressions for measures of L and $\tilde{L}$ for longer durations seem difficult to obtain, however. We therefore model the queue state during loss periods as a discrete-time Markov chain with an absorbing state. Since the composite loss period is of greater practical significance, indicating the number of consecutively lost customers, we will focus on this random variable for the remainder of this section. The states of the chain indicate the amount of unfinished work above h just after a batch has arrived. For computational reasons, we truncate the transition matrix to $K+2$ states and write

$$\mathbf{P} = \left[\begin{array}{c|cccccc} & 0 & 1 & 2 & \dots & K & K+1 \\ \hline 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ 1 & a_0 & a_1 & a_2 & \dots & a_K & 1-\sum_{j=0}^{K} P_{1j} \\ 2 & 0 & a_0 & a_1 & \dots & a_{K-1} & 1-\sum_{j=0}^{K} P_{2j} \\ \vdots & & & & & & \\ K+1 & 0 & 0 & 0 & \dots & a_0 & 1-a_0 \end{array}\right].$$

The choice of K depends on the accuracy requirements and available computational resources; it can be selected by increasing K by small increments until the solution does not change appreciably. Note that the zero state is absorbing since the loss period ends when the amount of unfinished work above h reaches zero. We therefore obtain the duration of a composite loss period by evaluating the chain's absorption probabilities over time, given the initial state probabilities determined by the distribution of the initial jump, also truncated to $K+2$ values.[6]

[6]For the tail of the loss-period, this random walk with drift and one absorbing barrier may be represented by the corresponding Brownian motion [202, p. 437].

Since the distribution of the composite loss probability typically has a very long tail, computing its expected value through summation of the weighted probabilities was found to be numerically inaccurate and computationally expensive. However, an alternate approach exists [203, p. 425]. Let d_j be the expected time to absorption into state 0, starting at state j. By the law of total probability, we can write a system of linear equations

$$d_j = \sum_{k=1}^{K+1} \mathbf{P}_{jk} d_k + 1.$$

In matrix from, the linear system consists of the $\mathbf{P}$ matrix with its first row and column removed. All results were derived using this approach.

5.3.3 The Noloss Period

The state evolution during a noloss period can also be modeled by a discrete-time transient Markov chain with initial state h. Unlike the continuous-time case, this model is exact. Since the number of possible states is limited to $h + 1$ (including the absorbing state $h + 1$ representing a new loss period), no truncation error is incurred.

We track the number of customers in the system just after arrivals, at n^+. Since we cannot easily accommodate zero first-passage times, we compute the conditional probability mass function of the length of a noloss period given that it lasts at least one slot. The pmf of the unconditional loss period is then simply the conditional pmf scaled by a_0. We denote the respective random variables by N' and N.

The transition probability matrix is similar to the one used to approximate the noloss period for the continuous-time case, Eq. (5.3), but since there are no departures when the system is empty, the first and second row are identical.

$$P = \left[\begin{array}{c|cccccc} & 0 & 1 & \dots & h-1 & h & h+1 \\ \hline 0 & a_0 & a_1 & & a_{h-1} & a_h & g_{h+1} \\ 1 & a_0 & a_1 & & a_{h-1} & a_h & g_{h+1} \\ \dots & & & & & & \\ h-1 & 0 & 0 & & a_1 & a_2 & g_3 \\ h & 0 & 0 & & a_0 & a_1 & g_2 \\ h+1 & 0 & 0 & & 0 & 0 & 1 \end{array}\right]$$

where

$$g_k = \sum_{j=k}^{\infty} a_j = 1 - \sum_{j=0}^{k-1} a_j.$$

The expected length of the noloss period can be obtained as in the continuous-time case by evaluating the steady-state probabilities of the return process. The state transition matrix of the return process is derived from the matrix $\mathbf{P}$ by replacing the last row $(h+1)$ with zeros except for a one in column h (see Eq. (5.9)).

The expected number of consecutive successful customers can be computed as discussed in Section 5.2.5.

5.3.4 Numerical Examples

For a first impression of the behavior of the loss period, we plot the mean value of the composite loss period as a function of the system load ρ in Fig. 5.3. As expected, the curves follow the characteristic pattern of loss probability and delay curves for queueing systems, with a gradual rise up to a "knee" point, followed by a region of high sensitivity to ρ for high loads. Exhibiting a pattern that will be seen in the following curves as well, the geometric case is clearly separated from the other distributions, which track each other closely in mean composite loss period.

We stated in Lemma 5 that the distribution of the initial jump and (composite) loss period of the $D^{[\mathrm{Geo}]}/D/1$ queue are independent of the threshold h. It seems therefore natural to investigate to what extent these quantities depend on h for other batch distributions commonly used for modeling data communication systems. As an example, consider the Poisson distribution and binomial distribution[7] with an

[7]The value of $\nu = 2$ used here is the smallest non-trivial value. As ν increases, the behavior should approach that of the Poisson distribution.

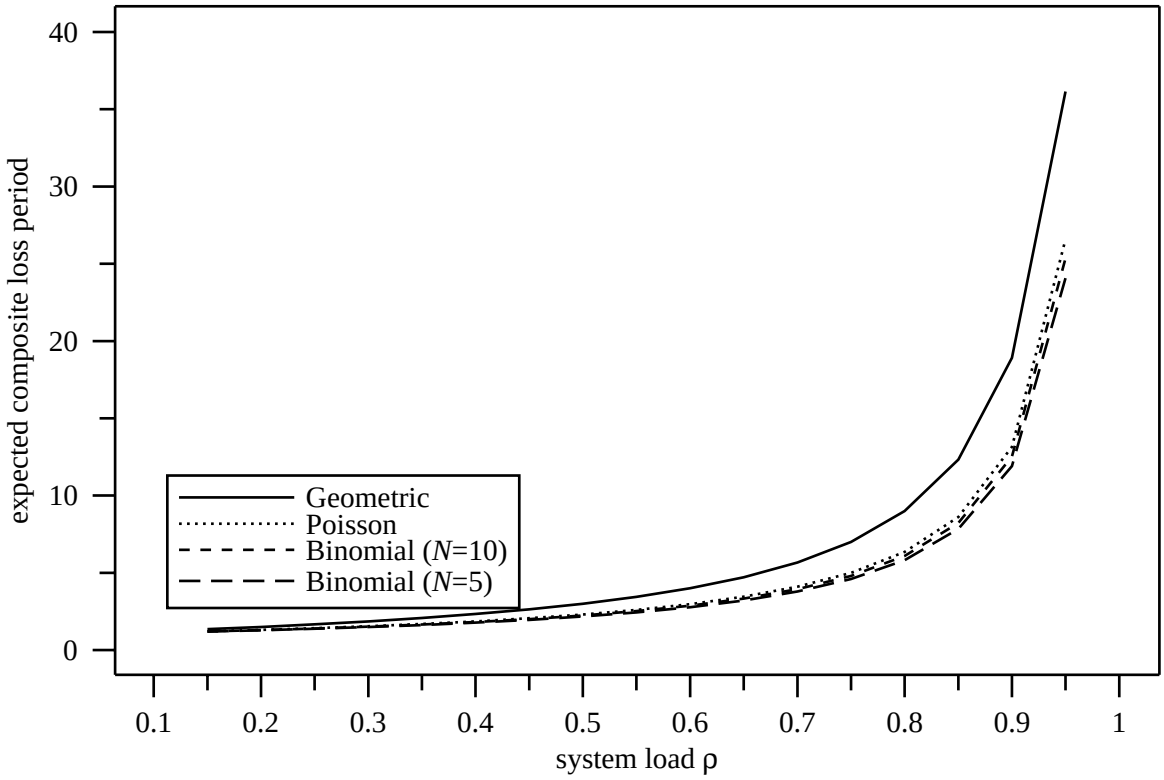

Figure 5.3. Expected composite loss period as a function of system load for $h = 5$

average batch size of $\rho = 0.8$. For values of h ranging from 3 on up (corresponding to losses of about 30% and less), Table 5.2 shows that h plays no significant role in $E[\tilde{J}]$ and consequently $E[\tilde{L}]$. (The same observation also holds for the distribution, not shown here.) It other words, for loss probabilities of practical interest, the loss period is basically independent of the threshold.

The distribution of the composite loss period is shown in Fig. 5.4. It is seen here that the distributions differ little for small composite loss periods, with most of the difference in expectation caused by the divergence in the tail of the distribution. The loss period for geometric batches tails off significantly faster than those for the either the Poisson or the binomial distribution.

Table 5.2. Probability of loss, expected composite loss period and jump for Poisson batches as a function of h for $\rho = 0.8$

	Poisson			Binomial ($\nu = 2$)		
h	α	$E[\tilde{J}]$	$E[\tilde{L}]$	α	$E[\tilde{J}]$	$E[\tilde{L}]$
0	1.0000	1.453	7.264	1.0000	1.25	6.25
1	0.6936	1.304	6.520	0.5556	1.00	5.00
2	0.4569	1.275	6.376	0.2469	1.00	5.00
3	0.2974	1.272	6.358	0.1097	1.00	5.00
4	0.1933	1.272	6.358	0.0488	1.00	5.00
5	0.1256	1.272	6.358	0.0217	1.00	5.00
6	0.0816	1.272	6.358	0.0096	1.00	5.00
7	0.0531	1.272	6.358	0.0043	1.00	5.00
8	0.0345	1.272	6.358	0.0019	1.00	5.00
10	0.0146	1.272	6.358	0.0004	1.00	5.00

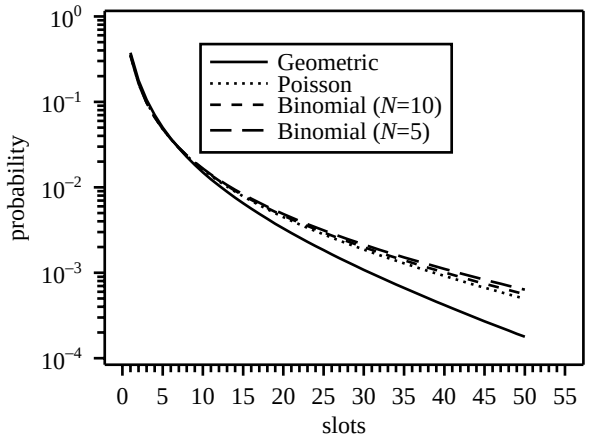

Figure 5.4. Probability mass function of the composite loss period for $\alpha = 0.1$ and $h = 5$

5.4 Buffer Overflow in Single-Stream Discrete-Time Queues

5.4.1 First-Come, First-Served

In this section, we will derive properties of the loss correlation for a class of discrete-time queues with restricted buffer size. As our queueing model, we consider a FIFO single-server discrete-time queue where arrivals occur in i.i.d. batches of general distribution with mean batch size λ. Each arrival requires exactly one unit of service. For short, we will refer to this system as $D^{[G]}/D/1/K$ [204]. Let K denote the system size, that is, the buffer capacity plus one. Arrivals that do not find space are rejected, but once a customer enters the system, it will be served. This buffer policy will be referred to as rear dropping in section 5.4.2. Arbitrarily, arrivals are fixed to occur at the beginning of a time slot and departures at the end, creating, in Hunter's terminology [200], an early arrival system. This model is used to represent the output queue of a fast packet switch, for example [205].

The waiting time and loss probability for this model have been analyzed by a number of authors [30, 198, 204–208]. For Poisson-distributed batches, Birdsall *et al.* [206, p. 392] compute the conditional probability of a run of exactly n slots in which one or more arrivals are rejected given that an arrival was rejected in the preceding slot. We will call this probability $P[C_R = n]$. The quantity is seen to be the product of the probability that two or more arrivals occur during the next $n-1$ slots and the probability of zero or one arrivals occurs in the terminating interval,

$$P[C_R = n] = e^{-\lambda}(1+\lambda)\left[1-(1+\lambda)e^{-\lambda}\right]^{n-1}.$$

Birdsall *et al.* [206, Eq. (11)] also compute the probability that exactly d arrivals are rejected in the next slot, provided that one or more was rejected in the previous slot. Their result is related to a relation we will derive later (Eq. (5.15)).

We define Q_k to be the event that the first customer in an arriving batch sees k customers already in the system and q_k to be the probability of that event. For

general batch size probability mass function (pmf) a_k, the q_k's are described by the following recursive equations [205]:

$$\begin{aligned} q_1 &= \frac{q_0}{a_0}(1 - a_0 - a_1) \\ q_n &= \frac{1}{a_0}\left[q_{n-1} - \sum_{k=1}^{n} a_k q_{n-k}\right], \ 2 \leq n < K \\ q_0 &= 1 - \sum_{n=1}^{K-1} q_n = \left[1 + \sum_{n=1}^{K-1} q_n/q_0\right]^{-1} \end{aligned}$$

The probability that a packet joins the queue, $P[J]$, is given by

$$P[J] = \frac{1 - q_0 a_0}{\lambda},$$

since $1 - q_0 a_0$ is the normalized throughput. Note that q_n, $n = 1, 2, \ldots$, depends on the buffer size only through the factor q_0, i.e., q_n/q_0 is independent of the buffer size [200, p. 236].

For later use, let us compute the probability $P[S]$ of the event S that one or more losses occurs during a randomly selected time slot. By conditioning on the system state k (which occurs with probability probability q_k), we can write

$$P[S] = \sum_{k=0}^{K-1} q_k P[S|Q_k] = \sum_{k=0}^{K-1} q_k \sum_{j=K-k+1}^{\infty} a_j = \sum_{k=0}^{K-1} q_k \left(1 - \sum_{j=0}^{K-k} a_j\right)$$

Let the random variable C_C be the number of consecutively lost customers. The distribution of loss run lengths follows readily,

$$\begin{aligned} P[C_C = n] &= \sum_{s=1}^{K} P[s \text{ spaces available} \mid \text{loss occurs in slot}] \cdot a_{n+s}, \\ &= \frac{1}{P[S]} \sum_{s=1}^{K} q_{K-s} a_{n+s} \end{aligned} \tag{5.13}$$

as the number of arrivals in a slot is independent of the system state.

The expected number of consecutive losses can be computed from Eq. (5.13) or directly by observing that loss runs are limited to a single slot since the first customer in a batch will always be admitted. The expected loss run length is simply

the expected number of customers lost per slot, given that a loss did occur in that slot. The expected number of customers lost in a slot is given by $\lambda(1 - P[J])$, so that

$$E[C_C] = E[\text{losses per slot} \mid \text{loss occurs in slot}] = \frac{\lambda(1 - P[J])}{P[S]} = \frac{\lambda - 1 + q_0 a_0}{P[S]}. \tag{5.14}$$

Numerical computations show that influence of K on the distribution of C_C is very small (see Table 5.3). The table also shows that $E[C_C]$ is roughly a linear function of λ.

Losses that occur when a batch arrives to a full system, i.e., a system with only one available buffer space, are independent of the system size and can thus be used to approximate the distribution of C_C quite accurately. For $K = 1$, n consecutive losses occur if and only if $n + 1$ packets arrive during the slot duration, conditioned on the fact that two or more packets arrived. Thus,

$$P[C_C = n] \approx P[C_C = n | K = 1] = \frac{a_{n+1}}{1 - a_0 - a_1} \tag{5.15}$$

$$\begin{aligned} E[C_C] &\approx \frac{1}{1 - a_0 - a_1} \sum_{n=1}^{\infty} n a_{n+1} \\ &= \frac{1}{1 - a_0 - a_1} \left[\sum_{n=2}^{\infty} n a_n - \sum_{n=2}^{\infty} a_n \right] \\ &= \frac{1}{1 - a_0 - a_1} \left[\lambda - a_1 - (1 - a_0 - a_1) \right] \\ &= \frac{1}{1 - a_0 - a_1} [\lambda - 1 + a_0]. \end{aligned} \tag{5.16}$$

As can be seen readily, the above agrees with Eq. (5.13) and Eq. (5.14) for $K = 1$.

Also, by the memorylessness property of the geometric distribution, Eq. (5.15) and Eq. (5.16) hold exactly for geometrically distributed batches with parameter p and evaluates to

$$P[C_C = n] = \frac{p(1-p)^{n+1}}{1 - p - p(1-p)} = p(1-p)^{n-1}$$

$$E[C_C] = \frac{1}{p} = 1 + \lambda$$

(Regardless of what system occupancy an arriving batch sees, the packets left over after the system is filled are still geometrically distributed.)

Table 5.3. Expected loss run length ($E[C_C]$) for $D^{[G]}/D/1/K$ system

a_k	K	$\lambda = 0.5$	$\lambda = 0.8$	$\lambda = 1$	$\lambda = 1.5$
Poisson	1	1.18100	1.30397	1.39221	1.63540
	2	1.15707	1.27511	1.36201	1.60574
	3	1.15707	1.27158	1.35911	1.60403
	4	1.15226	1.27153	1.35910	1.60403
	5	1.15238	1.27159	1.35914	1.60404
	6	1.15242	1.27160	1.35914	1.60404
	∞	1.15242	1.27160	1.35914	1.60404
Geo	any	1.5	1.8	2.0	2.5

5.4.2 Influence of Service and Buffer Policies

It is natural to ask how the burstiness of losses is affected by different scheduling and buffer management policies. As scheduling policies, FIFO (first-in, first-out) and non-preemptive LIFO (last-in, first-out) are investigated. For either policy, we can either discard arriving packets if the buffer is full (*rear discarding*) or push out those packets that have been in the buffer the longest (*front discarding*). Note that this dropping policy is independent of the service policy. From our viewpoint, LIFO serves the packet at the rear of the queue. Obviously, only systems with K greater than one show any difference in behavior.

The analysis of all but FIFO with rear discarding appears to be difficult. Let us briefly discuss the behavior of FIFO and LIFO, each either with front or rear discarding.

FIFO with rear discarding: The first customer in an arriving batch always enters the buffer and will be served eventually. Thus, a loss run never crosses batch boundaries.

FIFO with front discarding: Here, a batch can be completely lost if it partially fills the buffer and gets pushed out by the next arriving batch. However, if a batch was completely lost, the succeeding batch will have at least one of its members transmitted since it must have "pushed through" until the head of the buffer.

LIFO with rear discarding: The first packet in a batch will always occupy the one empty buffer space and be served in the next slot. Again, loss runs are interrupted by packet boundaries.

LIFO with front discarding: A run of losses can consist of at most than one less than arrive in a single batch since the last customer in the batch will be served during the next slot. A loss run never straddles batch boundaries.

For all four systems, indeed over all work-conserving disciplines, the queue length distribution (and, thus, the loss probability) are the same [209–211]. The mean waiting time results favoring front dropping agree with those of [211] for general queueing systems. Clare and Rubin [210] show that the minimum mean waiting time for non-lost packets is obtained using LCFS with front dropping (referred to as preemptive buffering in [210]).

For all systems, a batch arrival causes the same number of lost packets. If there are q packets in the buffer ($0 \geq q < K$) and a arrive in a batch, $[(q + a) - K]^+$ will be lost.

For rear dropping, at least the first packet in the batch will always enter the system, interrupting any loss run in progress. Thus, we have:

Lemma 6 *The same packets (as identified by their order of generation) will be dropped for all work-conserving service policies and rear dropping.*

Here, we visualize packets within the same batch generated sequentially in the interval $(t, t+0)$.

Lemma 7 *The distributions of loss runs for FIFO with rear and front dropping are the same.*

PROOF We number packets in the order of generation; packets within an arrival batch are numbered arbitrarily, but such that packets that are served are served in increasing sequence number.

We note first that the buffer for front dropping always contains an uninterrupted sequence of packets: Assume that the buffer contains an uninterrupted sequence. A service completion removes the first element of the sequence, without creating an interruption. A batch arrival that is accepted completely will also not create a sequence number gap within the buffer. A batch that pushes out some of the customers likewise will continue the sequence. Finally, a batch that pushes out all customers certainly does not create a gap within the buffer. Note that this property does not hold for rear dropping, as the example of two successive batch arrivals with overflows demonstrates.

As pointed out before, the loss runs for rear dropping are confined to a single arriving batch and comprise $[q+a-K]^+$ packets. For front dropping, the losses are made up of packets already in the buffer and possibly the first part of the arriving batch. By the sequence property shown in the preceding paragraph, all these form a single loss run (again of length $[q+a-K]^+$), which is terminated by serving the next customer. Thus, while the identity of packets dropped may differ, the lengths of the loss runs are indeed the same for both policies. □

The issue is more complicated for front dropping and general service disciplines. A number of simulation experiments were performed to investigate the behavior of the four combinations of service and buffer policies, with results collected in

Table 5.4. (The rows labeled "theory" correspond to the values for FIFO and rear dropping, computed as in discussed in the previous section.) These experiments suggest the following conjecture:

Conjecture 1 *The distribution of loss runs is the same for all combinations of FIFO, LIFO and rear and front dropping. The sample path of loss run lengths is the same for all systems except for LIFO with front dropping.*

Table 5.4. Performance measures for geometric and Poisson arrivals, $\lambda = 1.5$, $K = 4$, 90% confidence intervals

arrivals	service	dropping	$E[W]$	$1 - P[J]$	$E[C_C]$
geometric	theory		1.985	0.3839	2.500
	FIFO	rear	1.984...1.987	0.3833...0.3840	2.496...2.502
		front	1.349...1.351	0.3833...0.3840	2.496...2.502
	LIFO	rear	1.984...1.987	0.3833...0.3840	2.496...2.502
		front	0.867...0.869	0.3833...0.3840	2.495...2.500
Poisson	theory		2.417	0.341	1.604
	FIFO	rear	2.416...2.418	0.340... 0.341	1.602...1.604
		front	1.582...1.584	0.340... 0.341	1.602...1.604
	LIFO	rear	2.416...2.418	0.340... 0.341	1.602...1.604
		front	0.549...0.553	0.340... 0.341	1.603...1.604

Let us briefly outline a possible approach to a more formal analysis. We focus on one batch and construct a discrete-time chain with the state

$$(i, j) = (\text{left in buffer}, \text{consecutive losses in batch}) \qquad i \in [1, K-1];\ j \in [0, \infty)$$

The initial state, that is, the state immediately after the arrival of the batch of interest, is determined by the batch size and the system state and should be computable. The states $(0, j)$ are absorbing.

The transition matrix is given by:

$$
\begin{array}{lcll}
(i,j) \rightarrow (i,j) & = & a_1 & \forall i,j; \\
(i,j) \rightarrow (i-1,j) & = & a_0 & j > 0; \\
(i,j) \rightarrow (i-1,j+1) & = & a_2 & i,j > 0; \\
(i,j) \rightarrow (i-2,j+2) & = & a_3 & i > 1, j > 0; \\
\ldots & & & \\
(i,j) \rightarrow (i-k,j+k) & = & a_{k+1} & i > k, j > 0; \\
(i,j) \rightarrow (0,i+j) & = & \sum_{k=i+1}^{\infty} a_k & i,j > 0; \\
(0,j) \rightarrow (0,j) & = & 1 &
\end{array}
$$

All other entries are zero.

The probability of j losses given initial state (i_0, j_0) is the probability distribution of being absorbed in state $(0, j)$.

Intuitively, random dropping, i.e., selecting a random packet from among those already in the buffer, should reduce the loss run lengths, particularly for large buffers. However, this policy appears to be difficult to implement for high-speed networks. Simulation results for geometric arrivals support this result, as shown in Table 5.5 as a front or rear dropping would result in an average loss run length of 2.5 for $\lambda = 1.5$ and 1.8 for $\lambda = 0.8$.

Table 5.5. Effect of random discarding for system with geometrically distributed batch arrivals

	$\lambda = 1.5$			$\lambda = 0.8$		
K	$E[W]$	$1 - P[J]$	$E[C_C]$	$E[W]$	$1 - P[J]$	$E[C_C]$
3	0.913	0.4148	2.304	0.750	0.1731	1.669
4	1.509	0.3833	2.175	1.120	0.1213	1.573
6	2.842	0.3534	2.008	1.771	0.0659	1.451
8	4.304	0.3415	1.915	2.300	0.0386	1.382
10	5.851	0.3365	1.849	2.721	0.0234	1.328
15	9.884	0.3331	1.755	3.399	0.0071	1.238

It should be noted that average run lengths exceed a value of two only for extremely heavy load. Thus, on average, reconstruction algorithms that can cope with two lost packets in a row should be sufficient. Also observe that the loss

correlation has a noticeable dependence on the buffer size K, although the change in $E[C_C]$ is far less dramatic than that in the loss probability, $1 - P[J]$.

5.5 Summary and Future Work

In the preceding sections, we have developed probabilistic measures for the behavior of losses in single-server queues commonly used in analyzing high-speed networks and fast packet switches. These measures should be useful in the design and test of packet recovery systems, which face a much harder task than predicted by the optimistic assumption of independent losses.

We found that for certain important queues, the distribution of the loss period is independent of the threshold value used, while for all discrete-time batch distributions investigated the threshold value has very little influence on the loss period. It remains to be seen whether there exists a certain value of h above which the initial jump (and, hence, the loss period) changes no further with increases in h.

For $G/M/1$ and $D^{[\mathrm{Geo}]}/D/1$ queues, a busy period can be regarded as a special case of a loss period. Thus, computation of the distribution of busy periods is of particular interest in studying loss phenomena. Our computation of the busy period using combinatorial arguments applied strictly only to geometric batches, but it might provide a readily computable approximation for other distributions.

We also found in section 5.4, somewhat surprisingly, that for a discrete-time queue with general batch arrivals, LIFO and FIFO combined with different discarding policies exhibit the same loss correlation.

CHAPTER 6

LOSS CORRELATION FOR QUEUES WITH MULTIPLE ARRIVAL STREAMS

6.1 Introduction

As was emphasized already in the introduction to the previous chapter, loss probability and loss correlation supersede average waiting time and throughput as the performance measures determining quality of service for many real-time applications. Bursty traffic, as generated for example by multimedia applications, has made concerns about loss correlation more urgent.

This chapter generalizes the work described in the previous chapter, extending it to derive the loss correlation seen by packets of a selected stream in queues with *several* arrival streams. The goal of this work is to provide insights as to the structure and magnitude of loss correlation, so that the protocol and system architecture designer can judge when loss correlation will significantly effect system performance. The designer can judge whether either ameliorating measures are called for (in the case of real-time sequence reconstruction) or the benefits of loss correlation (in the case of retransmission algorithms) can be reaped.

Throughout this chapter, our interest in broadband ISDN and ATM-like systems motivates a discrete-time perspective, with packet losses caused by buffer overflow[1] We denote the event that packet n is lost as $L(n)$. Loss correlation is measured here by the conditional loss probability, the conditional probability that a packet from a stream is lost given that its immediate predecessor was also lost. The conditional

[1]It appears to be generally accepted that buffer overflow will be the dominant loss mode in wide-area high speed networks.

loss probability will be denoted as $P[L(n)|L(n-1)]$. The expected loss run length $E[C_L]$, that is, the average number of consecutively lost packets from a stream, is related to the conditional loss probability by

$$E[C_L] = \frac{1}{1 - P[L(n)|L(n-1)]}.$$

The chapter is divided into two major sections. First, in section 6.2, we focus on a system where a foreground stream with random or deterministic interarrival time competes with a batched background stream for buffer space. We allow general batch arrivals for the background stream and general, but i.i.d., interarrival times for the foreground stream. This model allows us to judge the dependence of the loss correlation as a function of the contribution of the foreground stream, the burstiness of the background stream, the buffer size and the space priority order of the streams. We find first that the buffer size has no appreciable influence for all but the smallest buffers. Also, it appears that as long as the contribution of the foreground stream to the total traffic is sufficiently small, the losses of that stream can be considered independent.

Then, in section 6.3, we address the issue of the influence of traffic correlation or burstiness on the loss correlation. Here, the input traffic consists of a superposition of identical sources, so that the total input traffic may be correlated. Since they are commonly used to model voice and video traffic, we choose the superposition of interrupted Poisson processes in the $N \cdot \mathrm{IPP}/D/c/K$ queue. We find that buffer size plays almost no role in the value of the loss correlation, while the burstiness affects both loss probability and conditional loss probability in a similar way.

6.2 Superposition of Periodic and Random Traffic

Up to this point, we have assumed that the stream analyzed is the only one arriving at a queue. Clearly, this is not realistic in many network situations. As a simple

first model closer to actual network traffic, we again consider a discrete-time system similar to the one studied in [212]: single, first-come, first-serve queue with batch arrivals at every time slot and unit service time. The stream of interest is modeled as periodic, superimposed on background traffic with general, but i.i.d., batch arrivals. The background traffic is envisioned as the superposition of a large number of sources. For brevity, the two streams are abbreviated as "foreground traffic" (FT, stream 1) and "background traffic" (BT, stream 0). If the foreground traffic is periodic, it is also referred to in the literature as constant bit-rate (CBR) [213] or continuous bit-stream oriented (CBO) [214, 215] traffic. This traffic model is motivated by voice and video applications, where the source of interests emits packets periodically during a talk spurt, a video scan line or even a whole image[2]. To gain insight into the loss behavior, we require transient rather than steady-state results.

A number of authors have investigated the issue of superimposing different traffic streams. Kaplan [216] develops steady-state waiting time results in the transform domain for a continuous-time system with a single deterministic server with infinite buffer used by a primary, deterministic (periodic) stream and a secondary or background stream with Poisson characteristics. A more general continuous-time model is treated in [217], where the primary stream can have any random interarrival and service distribution and the secondary stream is a sequence of deterministic arrivals whose interarrival times are not necessarily identical. The steady-state distribution of the virtual waiting time for the infinite-buffer case is described by an integral equation. For a secondary stream with random arrivals, consult [218]. A number of authors [219–221] analyze the superposition of independent periodic arrival streams at a single server with infinite waiting space. Bhargava *et al.* analyze the same system using the ballot theorem [215]. In general, the queue length survivor

[2]e.g., video sources with compression on whole image and/or smoothing buffer

function for infinite queues can be used to approximate the cell loss probability for finite queues for sufficiently low (order of 10^{-9}) loss probabilities [221].

Closely related to the issue of superimposed heterogeneous traffic streams is that of overflow problems, covered extensively in the literature. While many overflow models, motivated by call set up in the switched telephone network, assume loss systems, i.e., no buffer, (among others, [222, 223]), a number of authors consider delay systems. For example, Matsumoto and Watanabe [224] derive approximations for the individual and combined mean waiting time and overflow probability in a finite system with two overflow streams, modeled as interrupted Poisson processes (IPPs), and one Poisson stream, which is a special case of the more general model of a $MMPP/M/c/c+k$ queue analyzed by Meier-Hellstern [178]. In most cases, the distribution of the number of busy servers in an infinite-server system accepting the overflow input is evaluated. In [225], the Laplace transform of the inter-overflow time matrix probability density function is provided. In [226], the related model of alternating voice and data input to a finite buffer is treated.

Kuczura [227] treats infinite queues (in continuous time) with one Poisson and one renewal input, in particular the $GI + M/M/1$ and $GI + M/M/\infty$ systems. In the context of discrete-time single-server queues with infinite buffer, correlated inputs and unity service time, Gopinath and Morrison [228] include an example of a two-class model with different arrival statistics.

Superimposed traffic streams of the same statistical description are found in the analysis of packet voice multiplexers [229], usually in the form of approximations [17, 84].

In the following sections, we set out to analyze in some detail three implementations of the model described at the beginning of this section, increasing in generality as we progress through the section. The first two models assume a constant, deterministic interarrival time for the foreground traffic. The first model (Section 6.2.1) assumes that the foreground traffic arrives at periodic intervals. It

always either attempts to enter the queue before all background traffic arriving during the same slot (*first-admittance system*) or attempts to enter after all background traffic has competed for buffer space (*last-admittance system*). These policies provide bounds when the system practices source discrimination, i.e., has sources compete for buffer space in a specific order as is often the case in packet switches. The more general model of section 6.2.2 allows these two admittance policies, but also the more interesting case where the foreground packet is positioned randomly among the cells from the background stream. Finally, the third model in section 6.2.3 straightforwardly extends the analysis to the case where the interarrival time of the foreground stream is a random variable.

6.2.1 Constant Period, Fixed Position: First and Last Admittance Systems

For simplified analysis and to obtain best or worst-case results, we first assume that the cells that are part of the periodic stream arrive as either the first (early arrival) or last cell (late arrival) during a slot. The case when the FT cell always arrives first is of no particular interest as it will always get admitted. To evaluate performance for the pessimistic late-arrival assumption, we embed a discrete-time Markov chain at the (imagined) instant just after the arrival of the background traffic and just before the arrival of the cell from the periodic stream.

We first consider the simple $D^{[X]}/D/1/K$ system with batch size pmf a_k and transition matrix Q embedded after the arrivals, with states labeled 0 through K.

$$\mathbf{Q} = \begin{bmatrix} a_0 & a_1 & a_2 & \dots & a_{K-1} & 1-\sum_{j=0}^{K-1} q_{0j} \\ a_0 & a_1 & a_2 & \dots & a_{K-1} & 1-\sum_{j=0}^{K-1} q_{1j} \\ 0 & a_0 & a_1 & \dots & a_{K-2} & 1-\sum_{j=0}^{K-1} q_{2j} \\ 0 & 0 & a_0 & \dots & a_{K-3} & 1-\sum_{j=0}^{K-1} q_{3j} \\ \dots & & & & & \\ 0 & 0 & 0 & \dots & a_1 & 1-\sum_{j=0}^{K-1} q_{K-1,j} \\ 0 & 0 & 0 & & a_0 & 1-\sum_{j=0}^{K-1} q_{Kj} \end{bmatrix}$$

Given this transition matrix for the background traffic only, the transition matrix for the combined system, embedded just before arrivals of the periodic traffic with

period τ, is computed by taking the τth power of $\mathbf{Q}$ and defining $h_{i,j}$ as element i,j of that matrix.

$$\tilde{\mathbf{Q}} = \begin{bmatrix} h_{10} & h_{11} & h_{12} & \dots & h_{1,K-1} & 1-\sum_{j=0}^{K-1}\tilde{q}_{0j} \\ h_{20} & h_{21} & h_{22} & \dots & h_{2,K-1} & 1-\sum_{j=0}^{K-1}\tilde{q}_{1j} \\ \dots & & & & & \\ h_{K,0} & h_{K,1} & h_{K,2} & \dots & h_{K,K-1} & 1-\sum_{j=0}^{K-1}\tilde{q}_{K-1,j} \\ h_{K,0} & h_{K,1} & h_{K,2} & \dots & h_{K,K-1} & 1-\sum_{j=0}^{K-1}\tilde{q}_{K-1,j} \end{bmatrix}.$$

Note that the last two rows are identical since the customer from the periodic stream will not be admitted if the system is already in state K prior to its arrival.

After the usual computations, we arrive at the vector of steady-state probabilities as seen by a periodic arrival, $\gamma = (\gamma_0, \dots \gamma_K)$. Then, the loss probability of periodic arrivals is simply γ_K and the expected waiting time for admitted customers is

$$E[W_1] = \frac{\sum_{k=0}^{K-1} k\gamma_k}{1-\gamma_K}.$$

Finally, $\tilde{q}_{K,K}$ is the conditional loss probability (given that the previous customer was lost). The expected number of consecutively lost periodic customers is a function of the conditional loss probability and given by the properties of the geometric distribution as

$$E[C_C] = \frac{1}{1-\tilde{q}_{K,K}}.$$

As an example, consider a system under overload, with a geometrically distributed batch size of mean 0.7, a period τ of 3 (for a combined average load of 1.033) and a system size of 3. The periodic traffic experiences a loss of 15.74% and an average wait of 0.875. The conditional loss probability is 20.03%, with an average number of 1.25 consecutively lost customers (loss run length). If losses were independent, the average loss run length would be 1.187. Thus, even under heavy load and a periodic stream that contributes a significant fraction of the total traffic, the loss run lengths in that system are very close to one. The calculations were confirmed by simulations.

6.2.2 Constant Period, Random Position

In this variation, called the random-arrival system, the periodic cell is randomly interspersed with the background traffic. Again, a periodic arrival occurs every τ slots. We will refer to a sequence of τ slots commencing with a slot with a periodic arrival as a *cycle*. The first slot in a cycle bears the index zero.

6.2.2.1 FT Loss Probability and Waiting Time

The probability that k BT cells enter the queue before the FT cell is given by

$$b_k = \sum_{j=k}^{\infty} \frac{a_j}{j+1} \text{ for } k = 0, 1, 2, \ldots, \tag{6.1}$$

since the cell has an equal chance to be at any position of the total batch arriving to the queue. The expected number of cells entering the queue prior to the FT cell can be computed by rearranging the order of summation as

$$\begin{aligned} E[B] &= \sum_{k=1}^{\infty} k b_k = \sum_{k=1}^{\infty} k \sum_{j=k}^{\infty} \frac{a_{j+1}}{j} = \sum_{j=1}^{\infty} \frac{a_j}{j+1} \sum_{k=1}^{j} k = \sum_{j=1}^{\infty} a_j \frac{j(j+1)}{2(j+1)} \\ &= \frac{1}{2} \sum_{j=1}^{\infty} j a_j = \frac{1}{2} E[A], \end{aligned}$$

as intuition suggests. Here, $E[A]$ denotes the average batch size. By virtue of its random position, the FT cell is (in distribution) followed by as many BT cells as precede it. However, these two quantities are not independent within a single slot.

Through the remainder of this section, we will see that the results for the late-arrival system follow from those of the random-arrival system by simply replacing b_k by a_k.

As a preliminary step in the transient analysis, we write down the Markov chain for the background traffic only, embedded just before the first arrival in a batch. Here, the state space encompasses states 0 through $K-1$.

$$P = \begin{bmatrix} a_0 + a_1 & a_2 & a_3 & \ldots & a_{K-1} & 1 - \sum_{j=0}^{K-2} p_{0j} \\ a_0 & a_1 & a_2 & \ldots & a_{K-2} & 1 - \sum_{j=0}^{K-2} p_{1j} \\ 0 & a_0 & a_1 & \ldots & a_{K-3} & 1 - \sum_{j=0}^{K-2} p_{2j} \\ \ldots & & & & & \\ 0 & 0 & 0 & \ldots & a_0 & 1 - \sum_{j=0}^{K-2} p_{K-1,j} \end{bmatrix} \tag{6.2}$$

Similar to the case of fixed position arrivals, we now embed a Markov chain with state transition probability matrix $\tilde{\mathbf{P}}^{(j)}$ just prior to the first arrival in the jth slot of a cycle. This first arrival could be from either the periodic or the background stream. Since we embed the chain just after a departure, the state space reaches up to $K-1$, not K. The matrix $\tilde{\mathbf{P}}^{(j)}$ is given by

$$\tilde{\mathbf{P}}^{(j)} = \begin{cases} \mathbf{P}^{\tau-j}\mathbf{P}'\mathbf{P}^{j-1} & \text{for } 0 < j \leq \tau \\ \mathbf{P}'\mathbf{P}^{(\tau-1)} & \text{for } j = 0. \end{cases} \tag{6.3}$$

Here, $\mathbf{P}'$ is the transition matrix computed with the shifted batch size distribution that takes account of the single foreground arrival: $\mathbf{a}'$, $a'_j = a_{j+1}, j = 1, \ldots, K$, $a'_0 = 0$.

The steady state probabilities seen by the first customer in the jth slot of a cycle are derived from $\tilde{\mathbf{P}}$ and are denoted by the vector $\tilde{\boldsymbol{\pi}}^{(j)} = (\tilde{\pi}_0^{(j)}, \ldots, \tilde{\pi}_{K-1}^{(j)})$. Define the random variable B as the number of customers that arrive before the selected customer in the batch (i.e., the customer position). The random variable S describes the state seen by an arrival from the periodic stream, i.e, where $j = 0$. The steady-state probability of loss for periodic arrivals is given by the convolution of position and state distributions,

$$\begin{aligned} P[L_1] &= P[B + S \geq K] = \sum_{k=1}^{K} \tilde{\pi}_{K-k}^{(0)} P[B \geq k] = \sum_{k=1}^{K} \tilde{\pi}_{K-k}^{(0)} \left(1 - \sum_{j=0}^{k-1} b_j\right) \\ &= 1 - \sum_{k=1}^{K} \tilde{\pi}_{K-k}^{(0)} \sum_{j=0}^{k-1} b_j, \end{aligned} \tag{6.4}$$

since position B and state S are independent. Alternatively, given the above measures, the distribution of the state seen by a periodic arrival is given by

$$\pi_k = \sum_{j=0}^{k} \tilde{\pi}_j^{(0)} b_{k-j}.$$

Then, the loss probability $P[L_1]$ for periodic traffic is seen to be equal to π_K. $P[L_1]$ and π_k for the last-admittance system can be computed by replacing b_k by a_k. The first-admittance system has $b_0 = 1, b_k = 0$ for $k > 0$, so that $\pi_k = \tilde{\pi}_k^{(0)}$.

An arriving periodic customer has to wait for those already in the queue at the beginning of the slot and those customers within the same batch that enter the queue ahead of the periodic customer. Since these two components are independent, the distribution of the waiting time, $P[W = k]$, for the periodic traffic is given by the convolution

$$P[W = k] = \frac{1}{1 - P[L_1]} \sum_{j=0}^{k} \tilde{\pi}_j^{(0)} b_{k-j} = \frac{\pi_k}{1 - P[L_1]} \text{ for } k = 0, \ldots, K-1.$$

Again, the last-admittance and first-admittance quantities are computed by replacing b_k as described above.

6.2.2.2 BT Loss Probability

The loss probability for BT cells, $P[L_0]$, is computed indirectly through the channel utilization. The channel idle probability, without regard to traffic type, is computed as

$$P[\text{idle}] = \frac{a_0}{\tau} \sum_{j=1}^{\tau-1} \tilde{\pi}_0^{(j)}$$

since the slot steady-state probabilities seen by an arriving batch differ for each slot within a cycle of length τ. The term for j equal to zero is omitted since the FT arrival assures that the channel is never idle. From the idle probability, the loss probability for the combined stream, called $P[L]$, follows from flow-conservation as

$$P[L] = \frac{1 - P[\text{idle}]}{\lambda_0 + 1/\tau}.$$

Background traffic constitutes a fraction of

$$\frac{\lambda_0}{\lambda_0 + 1/\tau} = \frac{1}{1 + 1/(\lambda_0 \tau)},$$

while foreground traffic contributes

$$\frac{1/\tau}{\lambda_0 + 1/\tau} = \frac{1}{1 + \lambda_0 \tau}.$$

Given these fractions, the total loss probability can be expressed as the weighted sum of the stream loss probabilities, $P[L_0]$ and $P[L_1]$, as

$$P[L] = \frac{P[L_0]}{1 + 1/(\lambda_0 \tau)} + \frac{P[L_1]}{1 + \lambda_0 \tau}.$$

Then, the loss probability for the background traffic can be solved for:

$$P[L_0] = (1 + 1/\lambda\tau)\left[P[L] - \frac{P[L_1]}{1 + \lambda\tau}\right] \tag{6.5}$$

The same results for the BT loss probability $P[L_0]$ can also be obtained by direct methods, similar to those used to compute $P[L_1]$. First, we need to compute the probability mass function of the number of cells that arrive in the batch prior to a randomly selected customer. Since arrivals occur in batches, we are dealing with a random incidence problem. For all but slot zero in a cycle, we have that the distribution of the batch size that a random customer finds itself in is given by

$$\tilde{a}_n = \frac{n a_n}{\lambda_0}$$

Thus, the probability that a random customer finds k other customers ahead of it in its batch, β_k, is found by observing that a random customer is equally likely to occupy any position within the arriving batch. For all but the first slot (index j) within a cycle, we have

$$\beta_k^{(j)} = \sum_{i=k+1}^{\infty} \frac{\tilde{a}_i}{i} = \frac{1}{\lambda_0} \sum_{i=k+1}^{\infty} a_i, \qquad 0 < j < \tau - 1$$

Similarly, for slot zero in a cycle, the BT cell is equally likely to occupy positions 1 through $n + 1$ given that the batch it arrived in had size n. Thus,

$$\beta_k^{(0)} = \frac{1}{\lambda_0} \sum_{j=k}^{\infty} \frac{j a_j}{j + 1}$$

The loss probability for BT cells, $P[L_0]$, is then given by a relation analogous to Eq. (6.4), except that we average over all τ slots in a cycle:

$$P[L_0] = \frac{1}{\tau} \sum_{j=0}^{\tau-1} \left(1 - \sum_{k=1}^{K} \tilde{\pi}_k^{(j)} \sum_{i=0}^{k-1} \beta_i^{(j)}\right) = 1 - \frac{1}{\tau} \sum_{j=0}^{\tau-1} \left\{\sum_{k=1}^{K} \tilde{\pi}_k^{(j)} \sum_{i=0}^{k-1} \beta_i^{(j)}\right\} \tag{6.6}$$

Both methods of computing the BT loss, Eq. (6.5) and Eq. (6.6) require approximately the same computational effort that increases linearly in τ, assuming matrix powers are computed through eigenvalue methods.

It is tempting to embed a Markov chain just before FT arrivals as it would yield both the unconditional and conditional loss probability for FT packets. This, however, is difficult for random FT arrivals since the number of BT arrivals after the FT arrival depends on the number that have arrived before (in the same slot). The system state seen by the FT arrival in turn depends on the number of BT arrivals that precede the FT arrival.

6.2.2.3 Conditional Loss Probability

In order to compute the conditional loss probability, we note that the state at the next slot after a FT loss is always $K-1$. Also, all BT cells arriving within the same slot and following a FT cell that is lost are also lost and can thus be ignored. Thus, the dependence of the number of arrivals after the FT cell on the state is no longer a problem. Then, the conditional loss probability is given by

$$P[L_1(n)|L_1(n-1)] = \left[\hat{\mathbf{P}}^{\tau-1}\mathbf{B}\right]_{K-1,K},$$

where $L_1(n)$ indicates the event that FT arrival n is lost (n is the packet sequence number within the FT stream.) P denotes the transition matrix defined in Eq. (6.2), but padded to K rows and columns by zero elements. The K by K transition matrix $\mathbf{B}$ describes the state transition caused by the arrivals preceding the FT cell in the same slot. For randomly placed FT cells, the elements b_k are computed according to Eq. (6.1).

$$\mathbf{B} = \begin{bmatrix} b_0 & b_1 & b_2 & \dots & 1-\sum_{j=0}^{K-1} b_j \\ 0 & b_0 & b_1 & \dots & 1-\sum_{j=0}^{K-2} b_j \\ 0 & 0 & b_0 & \dots & 1-\sum_{j=0}^{K-3} b_j \\ \dots & & & & \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

For late-arrival FT, the matrix $\mathbf{B}$ contains the batch probabilities:

$$\mathbf{B} = \begin{bmatrix} a_0 & a_1 & a_2 & \dots & 1 - \sum_{j=0}^{K-1} a_j \\ 0 & a_0 & a_1 & \dots & 1 - \sum_{j=0}^{K-2} a_j \\ 0 & 0 & a_0 & \dots & 1 - \sum_{j=0}^{K-3} a_j \\ \dots & & & & \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

After having evaluated the conditional loss probability algebraically, we briefly highlight two structural properties. For this queue, the independence of loss correlation and buffer size observed earlier [212] holds exactly for certain (common) combinations of τ and K.

Theorem 6.1 *The conditional loss probability for FT packets is independent of K for values of $K \geq \tau$.*

Intuitively, this result holds since for $\tau \leq K$, the buffer that was full on a FT arrival will not empty until the next FT arrival slot, if at all. In other words, there will be a departure at every slot until the next FT arrival after a FT arrival has been lost. The probability that the next FT arrival again sees K in the system depends only on the arrival probabilities, not on the system size. We prove the result formally by sample path arguments.

PROOF We track the number of available spaces rather than the system occupancy. After a FT loss, there will be zero available slots. Thus, the conditional loss probability can be viewed as the probability of reaching state zero at the next FT arrival given that the (lost) FT arrival left state zero behind. At the end of each slot, the number of available spaces increases by one. Here, we make use of the fact that the condition $\tau \leq K$ guarantees a departure. At the beginning of each slot, it decreases by the number of BT arrivals, but not below zero. This behavior is completely independent of the system size. □

Theorem 6.2 *(i) The conditional loss probability is greater than or equal to the unconditional loss probability. (ii) The conditional loss probability decreases monotonically in K.*

PROOF The following argument shows part (i). For any customer, the probability that it is lost depends only on the state after the arrivals of the previous slot ("state") and the number of arrivals that precede the arrival during the same slot ("arrivals"). The latter quantity is by system definition independent of the state of the queue. Clearly, the loss probability for the next customer increases with increasing state[3]. For the loss probability, we consider a random customer. The state lies somewhere between 0 and K. In contrast, for the conditional loss probability, we only consider those customers that follow a lost customer of the same stream. Their state is known to be K. Since, in particular, some non-zero number of random customers will see states of less than K, their loss probability will be lower than those seeing K as state.

Part (ii) follows from a sample-path argument. Consider two systems that are distinguished only by their buffer size: one has a buffer of K slots, the other $K+1$. Since we are interested in the conditional loss probability, we assume that a loss has occurred in both systems. The arrival process in the following slots is independent of the loss event. Consider the state just after a batch arrival. As long as the $K+1$ system occupancy stays above zero, the state of the $K+1$ system is simply that of the K plus one. In that case, the probability that the next FT arrival finds the system full is the same for both systems. (This is the argument used in the proof of theorem 6.1.)

On the other hand, if the $K+1$ system reaches zero, we know that the K system will also be at zero since it must have been at zero at the previous slot already. (The occupancy can only decrease by at most one in this single-server system). Thus, from that point on the two systems share the same occupancy. The next FT arrival will only be lost if the occupancy reaches K or $K+1$, respectively. Since the arrivals are the same, the probability of that event for the smaller system is larger. □

---

[3]Formal proof by sample path arguments.

6.2.2.4 Numerical Examples

To gain a better understanding of the dependence of conditional and unconditional loss probabilities on system structure and parameters, we graph and discuss a number of numerical examples below. All graphs display unconditional and conditional loss probabilities for FT packets as well as the unconditional loss probability for the BT stream. (The conditional loss probability for BT is less interesting since we assumed that the BT stream is the aggregate of several streams.)

In Fig. 6.1, the loss behavior for a BT stream consisting of either geometrically and Poisson distributed batches (mean batch size of 0.8) is shown as a function of the system size K. The FT period is held constant at $\tau = 10$. We see that the unconditional loss probabilities for the BT and FT stream differ, at least for larger values of K, by a roughly constant factor. That factor is larger for geometric than for Poisson arrivals, as the burstiness of the latter is smaller (and thus closer to that of the periodic traffic). In comparing unconditional loss probabilities across BT batch distributions, we note that the Poisson arrivals are associated with loss probabilities that decrease much more rapidly with increasing system size, again reflecting the lower burstiness. As expected, the conditional loss probability monotonically decreases as K increases, but is independent of K for $K \geq \tau$, as predicted by Theorem 6.1 and shows very little dependence for values of $K \geq \tau/2$. (This behavior seems to be valid across the range of K and τ, judging from sampling experiments not shown here.) For interesting loss probabilities, the conditional loss probability is several orders of magnitude larger than the unconditional loss probability, but the average number of consecutively lost packets is still very close to 1 (1.049 for geometric arrivals and the given set of parameters).

Plotting loss against FT period τ, as in Fig. 6.2, yields similar conclusions. In addition, we note that the conditional loss probability approaches the unconditional loss probability from above as τ increases, reflecting the fact that as the FT arrivals

are spaced further and further apart, the congestion experienced by one arrival is dissipated by the time of the next FT arrival. The asymptotic behavior as τ approaches infinity will be discussed shortly, in Section 6.2.2.5. In the figure, the asymptotic loss probability is marked with a dot. It is worth noting that a single 64 KBit/second voice connection on a 150 MBit/second data link corresponds to a value of τ of approximately 2300. At that value, losses are virtually uncorrelated (according to our definition of loss correlation.)

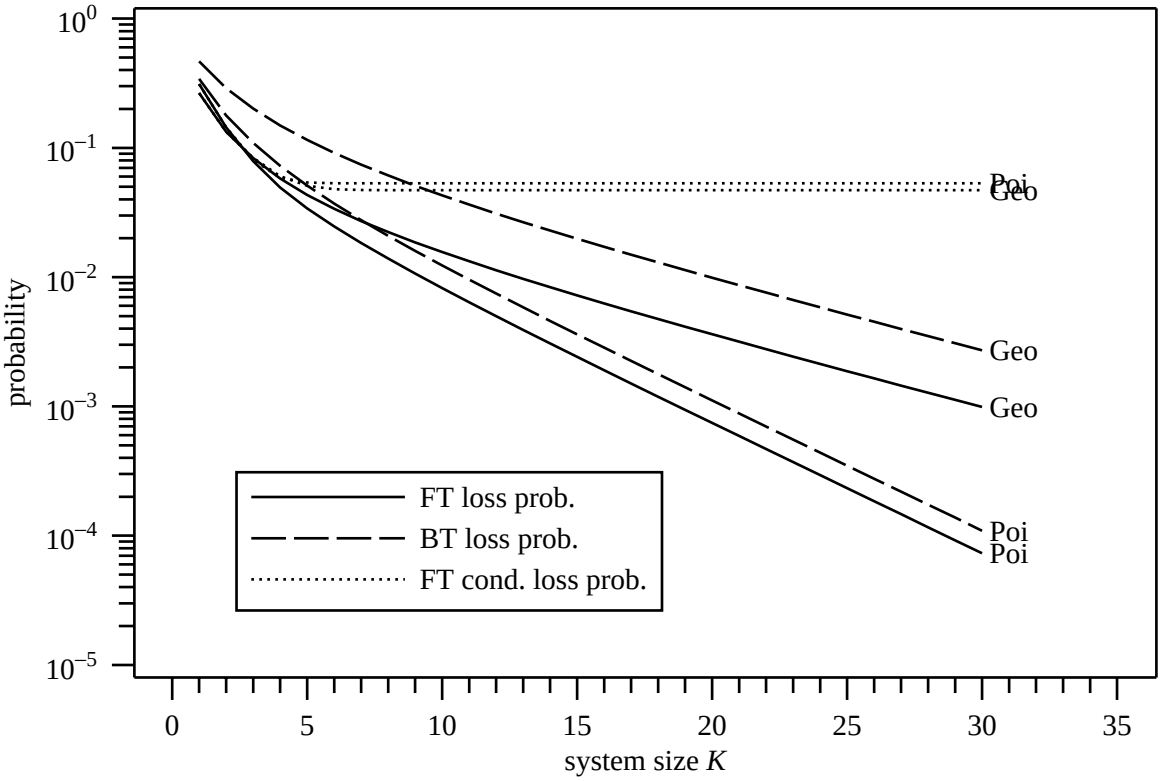

Figure 6.1. Loss probability and conditional loss probability for Poisson or geometrically distributed background traffic BT with $\lambda_0 = 0.8$ and periodic traffic FT with period $\tau = 10$, as a function of system size K

Rather than comparing batch size distributions, Fig. 6.3, Fig. 6.4, and Fig. 6.5 illustrate the connection between loss and the buffer admission policy for FT cells. Placing the FT arrivals randomly among the BT arrivals yields a conditional and

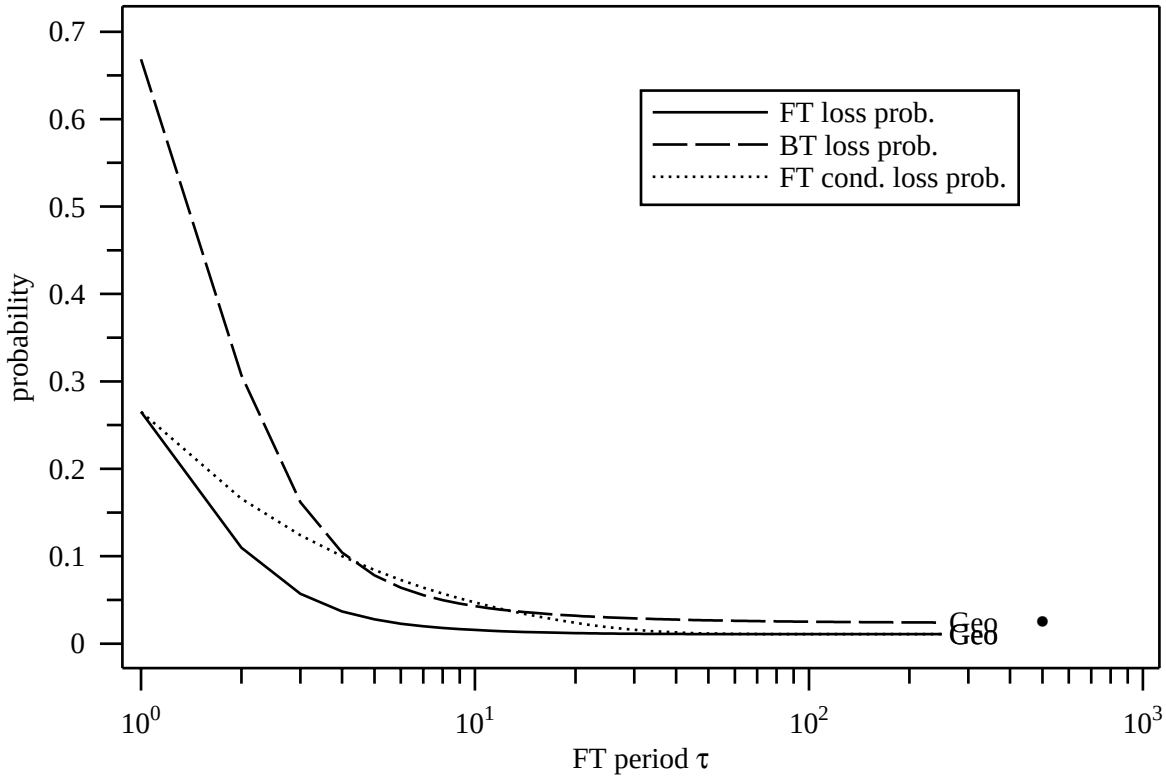

Figure 6.2. Loss probability and conditional loss probability for Poisson or geometrically distributed background traffic BT with $\lambda_0 = 0.8$ and periodic traffic FT (random arrival), as a function of τ; system size $K = 10$

unconditional probability that is better by factor of about five than that for the late-admittance system. The factor stays rather constant for all but the smallest values of K. Except for values of τ of less than, say, ten, the BT loss probability is largely unaffected by the treatment of FT arrivals. Beyond a certain value of τ, the overall system load and therefore the FT loss probability change relatively little.

In Fig. 6.6, the FT period τ is varied while the overall system load remains constant at $\lambda = 0.8$. As the contribution of periodic traffic diminishes and random traffic increases, the increased overall burstiness leads to increased loss probability. At the same time, the increased spacing of FT arrivals has the conditional loss probability approach the unconditional one as losses become independent. Again,

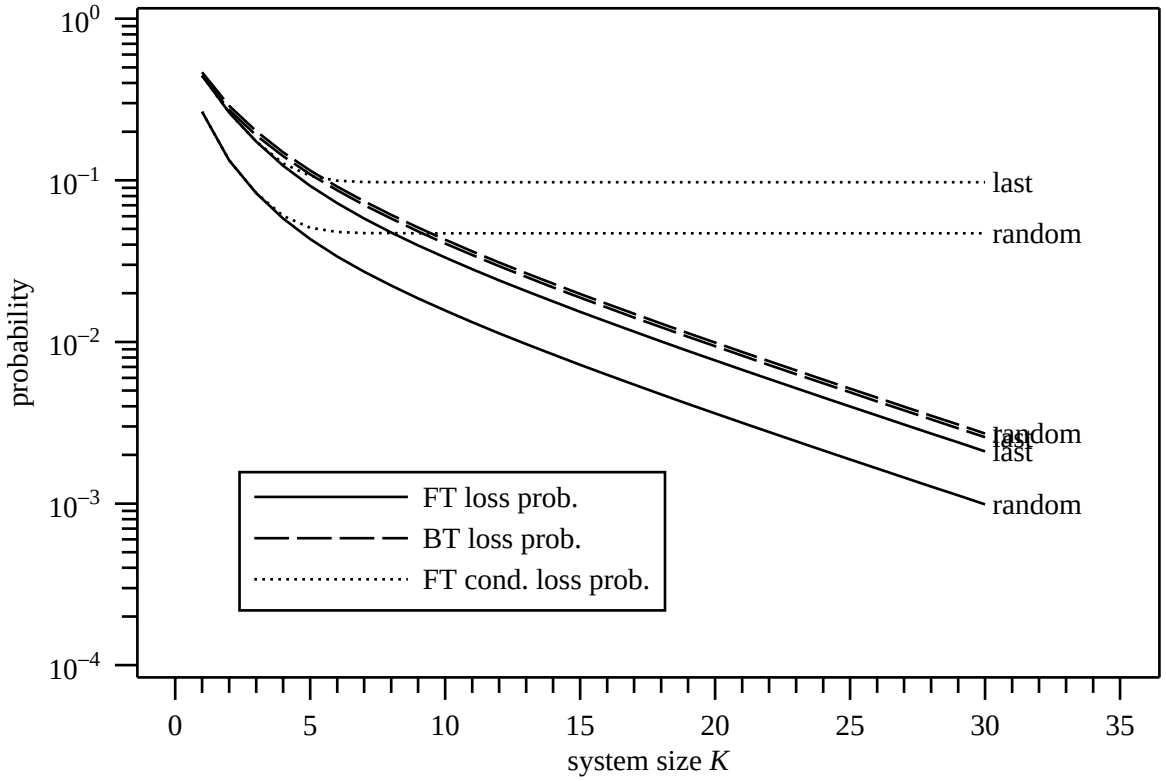

Figure 6.3. Loss probability and conditional loss probability for geometrically distributed background traffic BT with $\lambda_0 = 0.8$ and periodic traffic FT with period $\tau = 10$, as a function of system size K

the spacing by a constant factor of roughly five in both conditional and unconditional loss is observed between the two FT arrival patterns.

Almost all queueing systems show a roughly exponential rise in loss probability as the system load approaches one. This behavior is also found in Fig. 6.7 for the loss probabilities of both traffic types as the background traffic is varied from $\lambda = 0.2$ to 0.9. The buffer size and FT period are held constant at $K = 10$ and $\tau = 10$, respectively.[4] The flattening of the loss curve seen here is also common to all finite-buffer systems. The more interesting observation in our context, however, is that the expected loss run length varies relatively little, staying close to the value

[4]Thus, the expected run length would remain the same for all values of K greater or equal to 10.

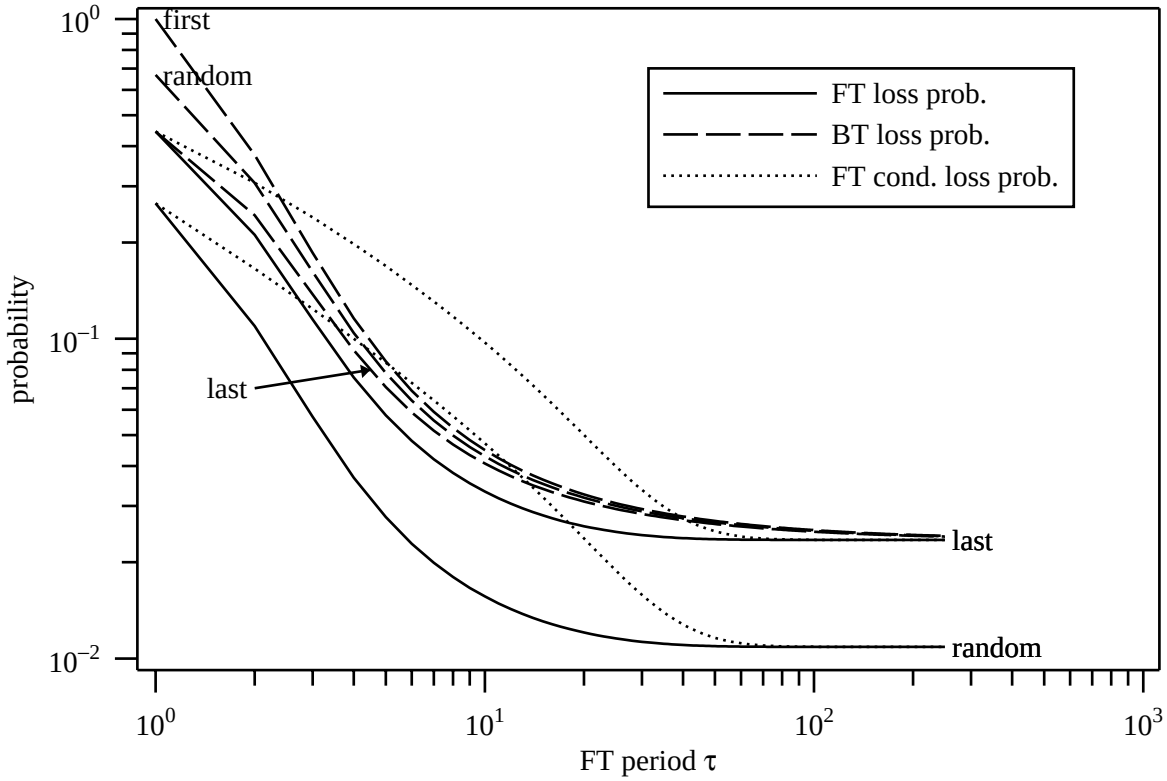

Figure 6.4. Loss probability and conditional loss probability for geometrically distributed background traffic BT with $\lambda_0 = 0.8$ and periodic traffic FT, as a function of τ; system size $K = 10$

of one associated with random loss. Note the vastly different scales of the left and right ordinate in Fig. 6.7. While the loss probability ranges over eleven orders of magnitude, the mean loss run length changes by less than ten percent.

Finally, Fig. 6.8 provides a graphic illustration of the effect of the conditional loss probability r on the distribution of loss run lengths. Since the loss run lengths are geometrically distributed, the tail of the run length distribution decays rapidly even for values of r significantly larger than "typical" loss probabilities.

6.2.2.5 Asymptotic Analysis in τ

It is instructive to take a closer look at the case when τ approaches infinity. We will show in this section that all three performance measures (loss of FT and BT,

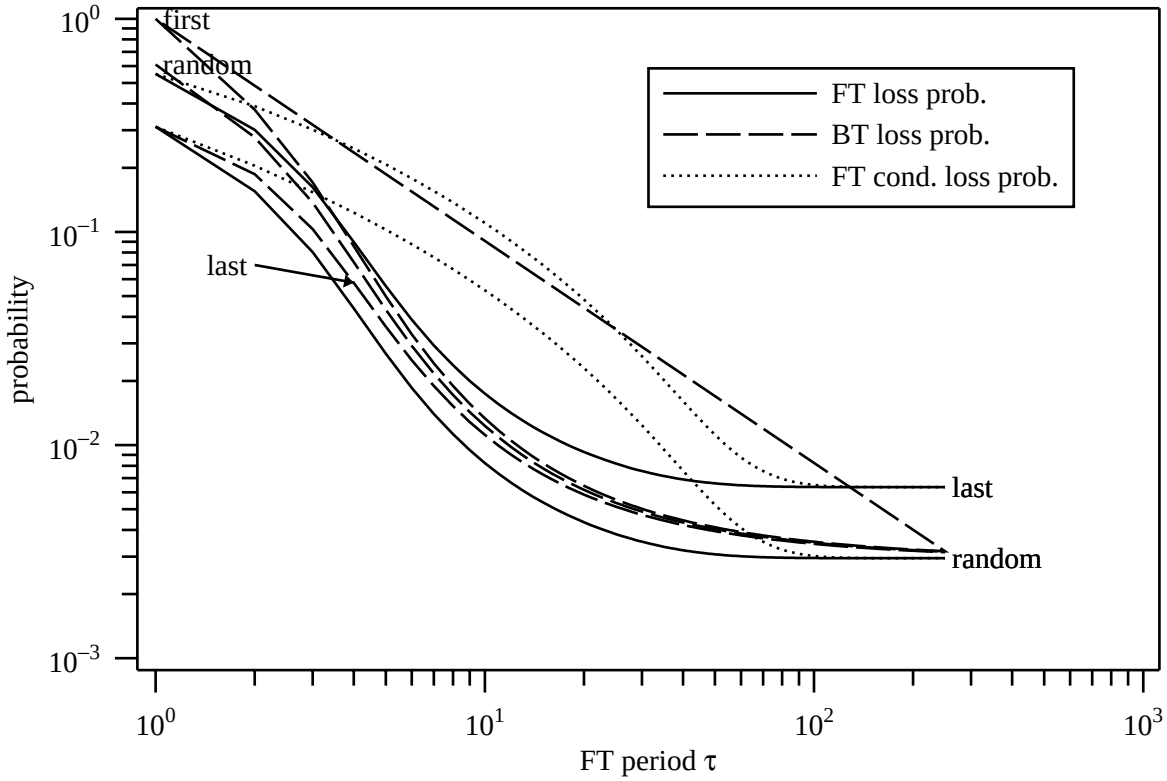

Figure 6.5. Loss probability and conditional loss probability for Poisson distributed background traffic BT with $\lambda_0 = 0.8$ and periodic traffic FT, as a function of τ; system size $K = 10$

conditional loss of FT) converge to the same value for Poisson BT, while FT and BT loss probabilities for geometric BT converge to different values. The convergence of the unconditional and conditional loss probability for all batch distributions was pointed out earlier and is to be expected since the state seen by one FT arrival is less and less correlated to that seen by the next FT arrival as the spacing between FT arrivals increases.

The unconditional loss probabilities for both types of traffic as τ approaches infinity is computed intuitively by ignoring the FT arrivals. The loss probability can easily be derived formally. We define

$$b_j = \sum_{i=j}^{\infty} \frac{a_i}{i+1},$$

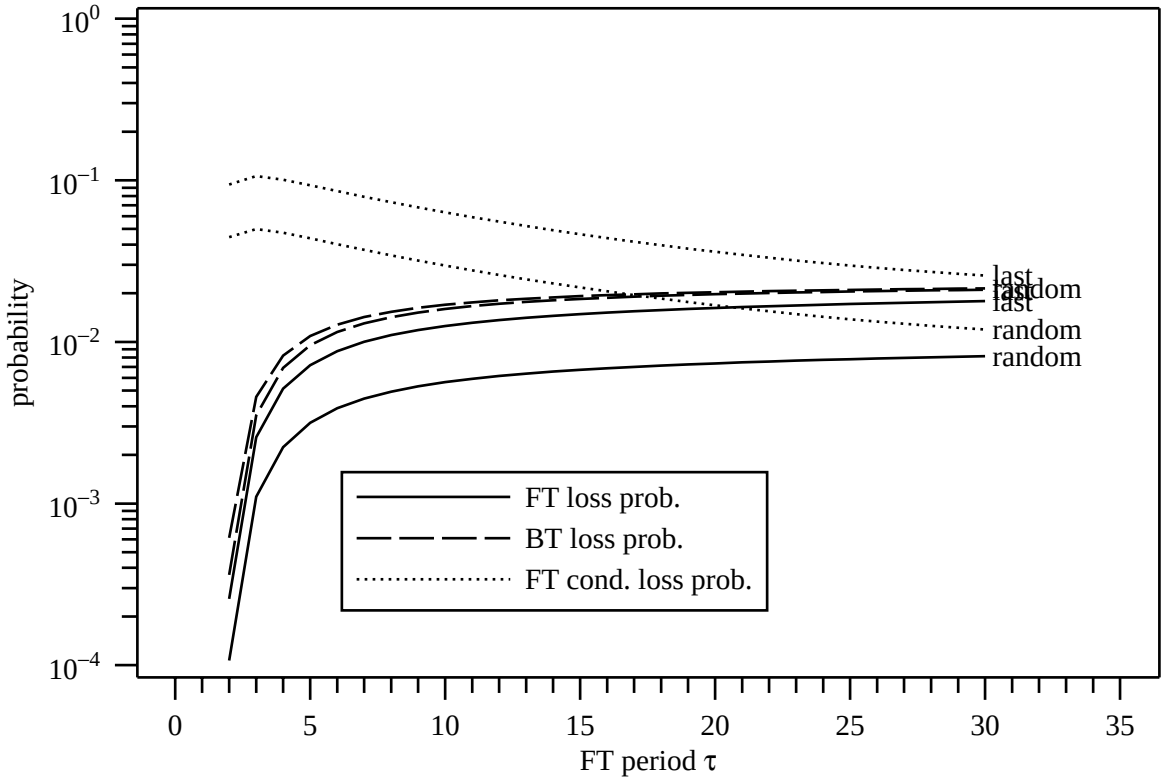

Figure 6.6. Loss probability and conditional loss probability for geometrically distributed background traffic BT with total load of $\lambda = 0.8$ and periodic traffic FT, as a function of τ; system size $K = 10$

$$\beta_j = \sum_{i=j+1}^{\infty} \frac{a_i}{\lambda}.$$

Also, $\tilde{\pi}_k$ is defined as the steady-state distribution computed from $\tilde{P}$. Using standard arguments [230, p. 45], we take the limit of Eq. (6.6):

$$\begin{aligned}
\lim_{\tau\to\infty} P[L_0] &= 1 - \sum_{k=1}^{K} \left\{ \lim_{\tau\to\infty} \frac{1}{\tau} \sum_{j=0}^{\tau-1} \tilde{\pi}_k^{(j)} \sum_{i=0}^{k-1} \beta_i^{(j)} \right\} \\
&= 1 - \sum_{k=1}^{K} \left\{ \lim_{\tau\to\infty} \tilde{\pi}_k^{(\tau)} \sum_{i=0}^{k-1} \beta_i^{(\tau)} \right\} \\
&= 1 - \sum_{k=1}^{K} \left\{ \tilde{\pi}_k \sum_{i=0}^{k-1} \beta_i. \right\}
\end{aligned}$$

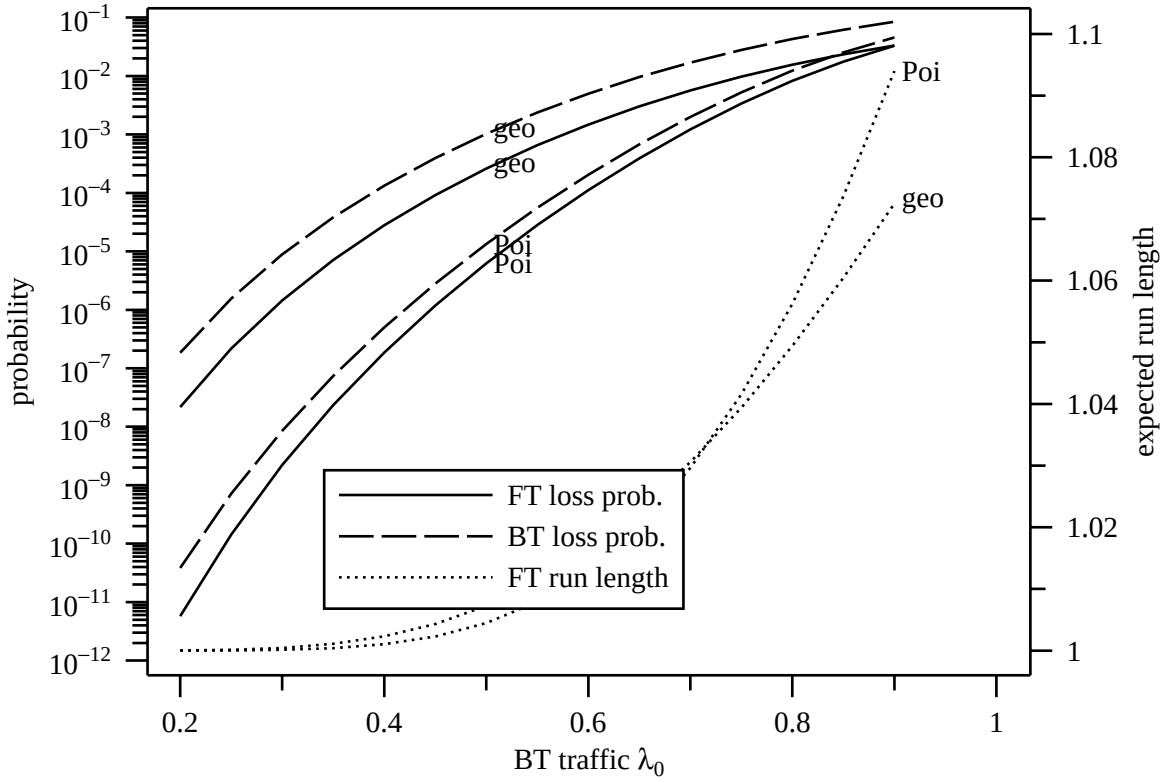

Figure 6.7. Loss probabilities and expected run length for Poisson and geometrically distributed background traffic BT and periodic traffic FT (random arrival), as a function of BT intensity λ_0; system size $K = 10$, period $\tau = 10$

Here, $\tilde{\pi}_k^{(j)}$ is the steady state distribution for the jth slot, which can be computed as one row of $[P^{(j)}]^\infty$. Taking the limit for the state transition matrix used to compute $\tilde{\pi}$, we obtain, $\lim_{j\to\infty} \tilde{P}^{(j)} = P^0 P' P^\infty = P' P^\infty = P^\infty$. The last step follows since P' is stochastic and all rows are equal to the steady-state state distribution in matrix P^∞.

Hence, every slot has the same transition matrix and state distribution. The derivation for the limit of $P[L_1]$ proceeds similarly. Therefore, the loss probabilities for background and foreground traffic are given by, respectively,

$$P[L_0] = 1 - \sum_{k=1}^{K} \tilde{\pi}_{K-k} \sum_{i=0}^{k-1} \beta_i, \tag{6.7}$$

$$P[L_1] = 1 - \sum_{k=1}^{K} \tilde{\pi}_{K-k} \sum_{i=0}^{k-1} b_i. \tag{6.8}$$

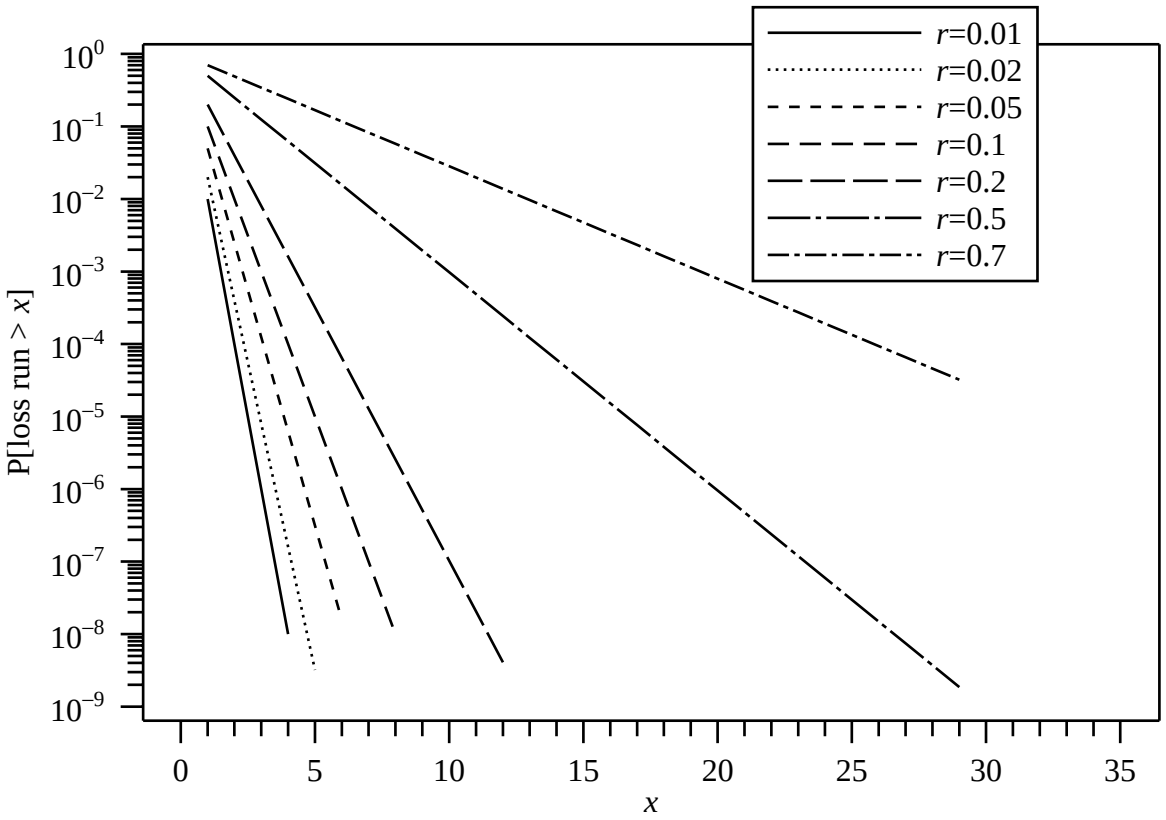

Figure 6.8. Probability that a loss run exceeds length x for given conditional loss probability r

(6.9)

The two loss probabilities are equal if and only if b_k equals β_k. For the geometric distribution with parameters p and q, due to its memorylessness property, β_j equals a_j, so that

$$\sum_{j=0}^{k-1} \beta_j = q^{k-1}.$$

No closed-form expression appears to exist for the corresponding sum of b_j. Clearly, β_j is not equal to b_j for the geometric distribution. For the Poisson distribution, however, we see that β_j and b_j, and therefore the loss probabilities, are indeed the same:

$$b_j = \sum_{i=j}^{\infty} \frac{e^{-\lambda}\lambda^i}{(i+1)i!} = e^{-\lambda} \sum_{i=j}^{\infty} \frac{\lambda^i}{(i+1)!},$$

$$\beta_j = \frac{1}{\lambda}\sum_{i=j+1}^{\infty}\frac{e^{-\lambda}\lambda^i}{i!} = e^{-\lambda}\sum_{i=j}^{\infty}\frac{\lambda^i}{(i+1)!}.$$

6.2.3 Random Period, Random Position

Previously, the foreground source was assumed to transmit at periodic intervals of constant duration τ. We now relax this restriction and allow the interarrival time of the foreground stream, τ, to be a random variable. By simple conditioning, the results for the conditional loss probability in the previous section can be extended to cover this case. If τ can assume only values strictly greater than zero, that is, the foreground cells do not arrive in batches, we can write

$$P[L_1(n)|L_1(n-1)] = \sum_{k=1}^{\infty} P[L_1(n)|L_1(n-1),\tau]P[\tau = k].$$

The evaluation of this sum is made easier by the fact that the conditional loss probability approaches the unconditional loss probability as τ gets large.

If batch arrivals are permitted for the foreground stream[5], renewal-type effects need to be taken into account and the conditional loss probability is written as

$$P[L_1(n)|L_1(n-1)] = P[\tau^* = 0] + \sum_{k=1}^{\infty} P[L_1(n)|L_1(n-1),\tau]P[\tau^* = k]. \qquad (6.10)$$

Here, τ^* denotes the interarrival time random variable as seen by a random foreground arrival, which is related to the interarrival distribution seen by a random observer by [231]

$$P[\tau^* = k] = \frac{kP[\tau = k]}{E[\tau]}.$$

The first term in the sum of Eq. (6.10) is due to the fact that once an arrival is lost within a slot, all subsequent arrivals within the same slot will also be lost.

[5]In most applications, batch arrivals from the foreground stream seem unlikely unless its peak rate exceeds the channel rate or a fast link feeds a switch with a slow outgoing link. The Knockout switch [199], for example, allows at most one packet from each input to reach the concentrator. Depending on the switch hardware to be modeled, the FT cells can either be treated as arriving consecutively or randomly distributed within the total arrivals in a slot.

Computation of the unconditional loss probability is more difficult, requiring a two-dimensional Markov chain, except for the case where the interarrival distribution translates into a distribution of the number of FT packets per slot that is identical and independent from slot to slot. This is satisfied for the geometric interarrival distribution with support $[1, \infty]$, where each slot sees zero or one FT arrivals according to a Bernoulli process.

6.2.4 Future work

A number of extensions to the model treated in this section suggest themselves. First, a stream could receive preferential treatment if it has suffered a loss at its last arrival, reducing the average loss run length to very close to one. However, implementation costs may not make this practical except in the case when the number of streams is very small (which is where the issue of long loss runs is of interest in any event).

In the analysis, it was assumed that the foreground stream contributes at most one packet per slot, which is a reasonable assumption as long as the outgoing link speed is at least as large as the access link speeds. The effect of batch foreground traffic on loss correlation could be investigated within the same analytical framework.

6.3 Bursty Traffic Sources

There appears to be general agreement that traditional, Poissonian traffic models do not capture the bursty arrival process that characterizes many proposed sources of BISDN traffic, for example packet voice, still and compressed full motion video. Since the exact behavior of these sources is ill understood and may depend strongly on the source material, several generic bursty-source models have been proposed and analyzed. These models differ not only in their generality, but also seem to be appropriate for different parts of the network. Note that altogether different

traffic models are appropriate when aggregating sources. (Examples include Markov modulated Poisson processes or batch-arrival processes.) We will emphasize discrete-time models for individual sources.

The first model, commonly referred to as an *on/off source*, features periodic arrivals during a burst and no arrivals during the silence periods separating bursts. For analytical tractability, it is often assumed that the burst and silence duration are exponentially (continuous time) or geometrically (discrete-time) distributed. The model was probably first used for packet voice sources with speech activity detection. Examples can be found in [17,89,181,232–234]. For packet video, Maglaris *et al.* [235] model each source as the superposition of a number of on/off minisources. For their picturephone source, about ten identical minisources provided sufficient accuracy.

Li also investigated the loss behavior for a voice source model [181] related to the $N \cdot \mathrm{IPP}/D/c/K$ queue studied here. However, he considered the temporal properties of the aggregate loss rate for all sources, not the behavior of individual sources. Li also arrives at the conclusion that the buffer size does not materially affect the loss behavior once an overload period sets in.

Multiplexers and other packet switch queues introduce delay jitter, thus, packet arrivals within a burst even of a on/off source will no longer be strictly periodic within the network. The interrupted Poisson process (IPP) source model attempts to capture this. In continuous time [236,237], the on/off time periods are exponentially distributed and independent of each other. A fixed-rate Poisson source emits packets during the on periods. In discrete-time [238,239], the on/off periods are taken to be geometrically distributed. A packet arrives at each slot during the on period with probability λ. The designation as a discrete-time IPP appears misleading since the source does not generate a Poisson-distributed number of cells in a time interval during on periods. Interrupted Bernoulli process (IBP) seems more descriptive.

When the peak rate of sources or input channels exceeds the output rate, a batch arrival model is more appropriate. We can generalize the IPP model by allowing

batch arrivals at each time slot during the on period. No performance evaluations using this model are known to the authors.

For the aggregate stream, a number of simplifying models can be used. The most popular in continuous time is probably the Markov-modulated Poisson process (MMPP), typically with two states. A simpler model aggregates all arrivals into batches, where batch arrivals occur as Bernoulli events with a given probability in a slot [238]. Thus, batches are separated by a geometrically distributed number of slots.

In this section, we limit our discussion to the IPP model. Possible extensions are discussed at the end of the section. In the subsequent discussion we will neglect source discrimination effects and assume random admittance, as defined in Section 6.2.

We begin in section 6.3.1 by reviewing some fundamental results for a single interrupted Poisson process (IPP) and then use these in section 6.3.2 first to describe the queue arrival process as a superposition of IPPs, proceed to find the loss probability and finally, the conditional loss probability. Numerical examples in section 6.3.4 illustrate the influence of buffer size and burstiness on loss correlation.

6.3.1 The Interarrival Time Distribution in an IPP

An interrupted Poisson process (IPP) is defined as follows:

Definition 1 *A discrete-time interrupted Poisson process (IPP) alternates between active (on) and silence (off) periods. At each slot, the state moves from active to silent with probability* $1-\gamma$ *and from silence to active with probability* $1-\omega$*. Thus, silence and activity durations are geometrically distributed. In the active state, a packet is generated independently from slot to slot with probability* λ*.*

We assume that a state transition takes place just prior to the end of a time slot and that a packet is generated at the beginning of a slot with probability λ if the

new state is the active state. Below, the silence and active states will be indicated by subscripts zero and one, where necessary.

For ease of reference, we reproduce the results for the interarrival (distance) random variable D presented by Murata *et al.* [238]. The average dwell times, T_i, for the off and on state, respectively, are given by $T_0 = 1/(1-\omega)$ and $T_1 = 1/(1-\gamma)$, or, conversely, $\omega = 1 - 1/T_0$ and $\gamma = 1 - 1/T_1$. We also define the activity factor α,

$$\alpha = \frac{T_1}{T_0 + T_1} = \frac{1-\omega}{2-\omega-\gamma}.$$

The z-transform of the interarrival times is computed [238, Appendix B] by recursively writing down the probability that no arrival occurred in k slots since the last arrival, given that the system is in burst or silence state after k steps. These probabilities are denoted by $p_1(k)$ and $p_0(k)$, respectively. The recursive equations are

$$\begin{aligned}
p_1(0) &= 1, \\
p_0(1) &= (1-\gamma)p_1(0), \\
p_1(1) &= \gamma p_1(0), \\
p_0(k) &= \omega p_0(k-1) + (1-\lambda)(1-\gamma)p_1(k-1), \ k = 2, 3, \ldots, \\
p_1(k) &= (1-\omega)p_0(k-1) + (1-\lambda)\gamma p_1(k-1), \ k = 2, 3, \ldots
\end{aligned}$$

Arrivals occur with probability λ if the system is in burst state, so that the z-transform of D, denoted by Δ_{IPP} can be derived as

$$\Delta_{\mathrm{IPP}}(z) = E[z^D] = \lambda P_1(z) = \frac{\lambda[(1-\gamma-\omega)z^2 + \gamma z]}{1 - (\gamma(1-\lambda)+\omega)z - (1-\lambda)(1-\gamma-\omega)z^2}.$$

Inversion of A_{IPP} yields the probability density function of the interarrival time, $\Delta_{\mathrm{IPP}}(k)$:

$$\Delta_{\mathrm{IPP}}(k) = d(1-r_1)r_1^{k-1} + (1-d)(1-r_2)r_2^{k-1} \tag{6.11}$$

where

$$r_{1,2} = \frac{1}{2}\left\{\gamma(1-\lambda) + \omega \pm \sqrt{(\gamma(1-\lambda)+\omega)^2 + 4(1-\lambda)(1-\gamma-\omega)}\right\}$$

$$d = \frac{1-\lambda\gamma-r_2}{r_1-r_2}$$

By differentiating the z-transform, or by taking the ratio of the burst over the total cycle time, we obtain the average interarrival time

$$E[\Delta] = \left.\frac{\partial\Delta_{\mathrm{IPP}}(z)}{\partial z}\right|_{z=1} = \frac{1/(1-\gamma)+1/(1-\omega)}{\lambda/(1-\gamma)} = \frac{1}{\lambda}+\frac{1-\gamma}{\lambda(1-\omega)}$$

and its inverse, the arrival rate

$$\rho = \frac{\lambda\cdot 1/(1-\gamma)}{1/(1-\gamma)+1/(1-\omega)} = \lambda\frac{1-\omega}{2-\gamma-\omega}.$$

The first moment can also be expressed in the quantities $r_{1,2}$ and d:

$$E[\Delta] = \frac{d}{1-r_1}+\frac{1-d}{1-r_2}$$

With the aid of the relation [240, p. 72]

$$\sum_{k=1}^{\infty} k^2 x^{k-1} = \frac{x+1}{(1-x)^3},$$

we obtain the second moment from the probability density function,

$$E[\Delta^2] = d\frac{1+r_1}{(1-r_1)^2}+(1-d)\frac{1+r_2}{(1-r_2)^2}.$$

The autocorrelation function of this process was derived by Gihr [241]. Other metrics, and methods of matching this process to an observed random process, are provided by Gilbert [160].

6.3.2 The $N\cdot IPP/D/c/K$ queue

We analyze the $N\cdot$IPP$/D/c/K$ queue operating in discrete time. N IPP sources, as described above, feed into the queue, which has K buffer positions, **not** including the customer being served. There are c identical servers. Each customer requires a service time of one time slot. All customers arrive at the beginning of a slot, while the departure, if any, occurs at the end of the slot (so-called *early arrival system* [200]).

We work towards the goal of computing the conditional loss probability[6] in several steps. First, immediately below, we characterize the aggregate arrival process. Then, we write down the transition matrix, which also yields the (unconditional) loss probability. Through a number of intermediate quantities, we finally arrive at the desired conditional loss probability.

To set the stage, we review the behavior of the aggregate arrival process. The superposition of N i.i.d. IPPs is described in [239]. The transition probability that j sources are active in the current slot, given that i were active in the previous slot can be computed by [239, Eq. (3)]

$$S_{i,j} = \sum_{m=0}^{i} \left[\binom{i}{m} \gamma^m (1-\gamma)^{i-m} \right] \left[\binom{N-i}{j-m} (1-\omega)^{j-m} \omega^{N-i-(j-m)} \right]$$
$$\text{for } i,j \in [0,N], 0 \leq j-m \leq N-i$$

The first bracketed term is the probability that $i-m$ sources go off, while m sources stay on; the second bracketed term is the probability of $j-m$ sources turning on and $N-i-(j-m)$ staying off. The steady-state distribution vector $\mathbf{s}$ of the number of active sources is, as usual, computed as $\mathbf{s} = \mathbf{sS}$, where $\mathbf{S}$ is the transition probability matrix with elements $S_{i,j}$. We can, however, write it down immediately:

$$s_i = \binom{N}{i} \alpha^i (1-\alpha)^{N-i}.$$

The mean and variance of the number of active sources are $N\alpha$ and $N\alpha(1-\alpha)$. The correlation coefficient r is given by [233] $r = \omega + \gamma - 1$.

Because of batch-effects, the activity distribution vector seen by a random arrival, $\mathbf{s}'$, is a scaled version of the vector $\mathbf{s}$ [185, p. 69]:

$$s'_i = \frac{i s_i}{E[i]}$$

[6] As usual, we mean the conditional probability that customer n is lost given that customer $n-1$ from the same source was also lost.

Given that i sources are active, the number of packets generated during a slot is binomially distributed with pmf

$$a_{i,k} = \binom{i}{k}\lambda^k(1-\lambda)^{i-k} \text{ for } k = 0, 1, \ldots, i$$

and zero otherwise. The distribution of batch sizes seen by a random observer is given by

$$a_k = \sum_{i=1}^{N} a_{k,i}s_i,$$

while the distribution of batch sizes seen by a random *arrival* is given by $a'_k = ka_k/E[A]$, where $E[A]$ is the expected batch size as seen by a random observer. Naturally, it equals ρ.

6.3.2.1 The Loss Probability

As a first performance metric, we derive the customer loss probability of this queueing system through the state transition matrix. The state of the system is described by the tuple (k, i), i.e., k cells in the system (waiting and in service), with i sources active and $N - i$ silent. For brevity, k and i will be referred to as the occupancy and activity, respectively. The state is tracked on slot boundaries, with state transitions occurring on the boundary and batch arrivals immediately afterwards. As in all finite-capacity, early-arrival systems with deterministic service time and batch arrivals, the first cell in a batch always enters the system. At most a single cell departs immediately prior to the slot boundary.

The $(N, K) \times (N, K)$ transition matrix is given by

$$\begin{array}{lcll} P_{(i,q);(j,r)} & = & S_{i,j}a_{i,r-q+1} & \text{for } 0 \le q \le K, 0 < r < K \\ P_{(i,q);(j,K)} & = & S_{i,j}\sum_{m=K-q+c}^{\infty} a_{i,m} & \text{for } 0 < q \le K \\ P_{(i,q);(j,0)} & = & S_{i,j}\sum_{m=0}^{c-q} a_{i,m} & \\ P_{(i,q);(j,r)} & = & 0 & \text{otherwise} \end{array}$$

where we define $a_{i,k} = 0$ for $k < 0$ and $k > i$. We denote the corresponding steady-state distribution as $\mathbf{p}$ with elements $p_{i,k}$.

The probability that a customer is lost, $P[L]$, is most readily computed by conservation-of-flow arguments as the ratio of channel occupancy, called ν here, to arrival rate:

$$1 - P[L] = \frac{\nu}{\rho} = \frac{1 - \sum_{i=0}^{N} p_{i,0} a_{i,0}}{\rho} \text{ for } c = 1.$$

For general values of c, the mean channel occupancy is given by

$$\nu = \sum_{i=1}^{N} \left[\sum_{k=1}^{c-1} \sum_{j=0}^{i} p_{i,k-j} a_{i,j} + c \sum_{j=0}^{i} p_{i,k-j} \sum_{m=j}^{i} a_{i,m} \right]$$

6.3.3 The Conditional Loss Probability

The conditional loss probability is computed by tracing the state of the system from the time some random customer, called the test customer, is lost to the time when the next customer from that same source, the so-called test source, arrives. The computation is only possible because arrivals from a *single* source are i.i.d., even though the aggregate arrival process is correlated.

The computation proceeds in four major steps. First, we compute the activity seen by the test customer. Then, we compute the activity without the test source at the next slot boundary immediately following the loss. At both instants, the occupancy is known to be $K + 1$ or K, respectively, since at least one loss occurred. Then, we condition on the interarrival time A of the test source. For each value of the interarrival time, we compute the state, i.e., occupancy and activity, at the slot beginning of the slot containing the next arrival from the test source. Finally, we arrive at the loss probability of the next arrival from the test source by computing the number of customers from other sources that arrive in that slot.

Even though not directly needed for the final result, the conditional probability that one or more customers are lost during a slot given that $M = i$ sources are active is given by

$$P[\Omega | M = i] == \sum_{k=0}^{K} \frac{p_{i,k}}{s_i} \sum_{j=K-k+2}^{i} a_{i,j}.$$

Here, we designate the overflow event, i.e., one or more customers are lost, by Ω.

Through Bayes' rule, the conditional probability that i sources are active given that one or more losses occurred during a slot can be computed given that the converse probability is available:

$$P[M = i|\Omega] = \frac{s_i P[\Omega|M = i]}{\sum_{i=1}^{N} P[\Omega|M = i]s_i}.$$

Recall that L defines the loss event for a customer. For the first step, we need the probability $P[M = i|L]$ that a lost customer sees i active sources (including itself), which is most readily derived by Bayes' rule,

$$P[M = i - 1|L] = \frac{s_i' P[L|M = i]}{\sum_{i=1}^{N} P[L|M = i]s_i'},$$

where

$$P[L|M = i] = \sum_{k=0}^{K} \frac{p_{i,k}}{s_i} \sum_{j=K-k+1}^{i-1} b_{i,j},$$

and s_i' is the activity seen by a random arrival, with $s_i' = is_i/E[s]$.

$P[L|M = i]$ depends in turn on the number of arrivals that precede a randomly chosen customer in the same slot, conditioned on the number of active sources i in a slot. Since a customer is equally likely to occupy every position within the batch, we have that

$$b_{i,k} = \begin{cases} \sum_{j=k}^{i} \frac{a_{i-1,j}}{j+1}, & \text{for } k < i, \\ 0, & \text{otherwise.} \end{cases}$$

For convenience, we define $b_{0,0} = b_{1,0} = 1$.

Naturally, $P[L|M = i]$ is related to the loss probability:

$$P[L] = \sum_{i=1}^{N} P[L|M = i]s_i',$$

The initial state vector $\mathbf{q_0}$ seen by the test customer is written as

$$q_0(i,k) = \begin{cases} P[M = i - 1|L], & \text{for } k = K \\ 0, & \text{otherwise.} \end{cases}$$

Note that here and in the following steps, we condition on the interarrival time so that the test source is known to be inactive up to the last slot of that interarrival time.

The transition matrix $\tilde{P}_0$ from the loss of the test customer to the next slot has to reflect the fact that the test source is inactive. Any arrivals that occur after the test customer are lost and do not affect the system state. Thus,

$$\tilde{P}_{0(i,k);(j,l)} = \begin{cases} \tilde{S}_{i,j}, & \text{for } k = l \\ 0, & \text{otherwise.} \end{cases}$$

Here, $\tilde{S}$ and $\tilde{P}$ are the source activity and queue state transition matrices for $N-1$ sources, respectively.

In the slot where the test source emits the next customer, the activity increases deterministically by one. Also, the arrivals before the customer from the test source increase the queue occupancy. Note that the occupancy ranges now from 0 to $K+1$. This $(N, K+1) \times (N, K+1)$ transition matrix is therefore written down as

$$\tilde{P}_\infty(i,k);(j,l) = \begin{cases} \sum_{j=l-k}^{N} b_{i+1,j} & \text{for } k \le l \le K+1, 0 \le i < N, j = i+1 \\ 1 & \text{for } 0 \le k \le K, i = j = N \\ 1 & \text{for } 0 \le i < N, j = i+1, k = l = K+1 \\ 0 & \text{otherwise.} \end{cases}$$

Given all these intermediate quantities, we then uncondition on the interarrival distribution $P[D = n]$ and obtain the state distribution vector seen by the next arrival from the test source as

$$\mathbf{q} = \sum_{n=1}^{\infty} \mathbf{q_0} \tilde{P}_0 \tilde{P}^{n-1} \tilde{P}_\infty P[A = n]. \tag{6.12}$$

The conditional loss probability is then $\sum_{i=0}^{N} q_{i,k}$. Note that the interarrival time distribution, Eq. (6.11), is monotonically decreasing in k, simplifying the truncation of the sum when numerically evaluating Eq. (6.12).

The computation cost is dominated by the matrix multiplication to compute $\tilde{P}^n$ and the matrix product. Since the computation time is proportional to $(N \cdot K)^3$, and, to make matters worse, larger N tend to increase the number of terms that need to be evaluated in Eq. (6.12), a direct computation is only feasible for relatively small systems. On the other hand, as pointed out in more detail below, the loss correlation depends very little on K, so that $K = 1$ or even $K = 0$ can be used.

Also, non-random behavior is most pronounced when a small number of sources is superimposed. It remains to be determined whether bounding the conditional loss probability is possible, for example by assuming that the number of active sources remains constant in the interarrival interval. Choosing the distribution as seen by a lost customer should provide an upper bound, while choosing the distribution seen by an average arrival should bound the conditional loss probability from below. As a closer approximation, we could treat the number of active sources as a birth-death process, as is commonly done for voice multiplexers [232,233], and then use numerical methods for sparse and block matrices. This approximation would probably improve in accuracy with larger N and longer state holding times.

While the discussion has focused on the $N \cdot \mathrm{IPP}/D/c/K$ system, a number of extensions are immediately apparent. We discuss them briefly in their rough order of complexity. As for the example in Section 6.2, different ordering of sources in competing for buffer space (priority discarding) can be modeled by appropriately modifying $b_{i,j}$. Sources with more than two states are also readily handled at the expense of a larger state space. Note that as long as the loss probability for the individual source can be determined, only the aggregate batch arrival process of the remaining background sources enters into the calculation.

The case of an on/off source with *periodic* arrivals during the active periods can be approximately treated within this framework as well by making some simplifying assumptions. First, the interarrival time of the test source can be modeled as a constant since most of the consecutive losses will take place within the active period of the source. Then, the superposition of the remaining sources should have roughly the same impact on the test source as the superposition of the sources with Bernoulli arrivals, at least for large N. These assumptions remain to be validated by analysis. Also, the case of several heterogeneous sources and of sources with two different Bernoulli arrival processes (MMBP) rather than on/off sources deserve closer scrutiny (see [242,243] for a loss probability analysis).

In the Gilbert channel model [160], the system alternates between two states in a manner identical to the IPP source. In one state, every packet is lost, in the other, no losses occur. While the loss runs are indeed geometrically distributed, simulations indicate that the success runs (the sequences of customers from a single source without loss) are not. It remains to be investigated whether the model can be applied nevertheless. Naturally, the mean of the success runs can be computed from the loss probability and the loss run length (see [212]).

6.3.4 Numerical Examples

Compared to earlier systems, the large number of queueing system parameters (λ, γ, ω, N, K, c) covers a wide range of systems, but also makes it difficult to generalize from numerical examples.

The systems studied earlier had the property that the conditional loss probability depended only weakly, if at all, on the system size K. This property carries over into this queueing system, as indicated by Fig. 6.9, and other examples not shown here. The effect of K will be small, by the argument of Theorem 6.1, as long as most of the probability mass of the interarrival time is concentrated in the first K slots.

The influence of the channel bandwidth is indicated by Fig. 6.10, where the ratio of the source on/off durations and the system load are kept constant, but the number of sources, N, is raised, with λ, γ and ω scaled correspondingly. Thus, the source correlation coefficient r increases with N. This leads to a somewhat unexpected result: The conditional loss probability decreases with increasing N, indicating that the effect of burstiness is more than compensated for by the "thinning" during active periods.

As a final parameter study, we investigate the effect of different source behavior, again maintaining the same average load by adjusting λ and α. We also keep the sum of active and silent time constant. Thus, for high values of λ, a source

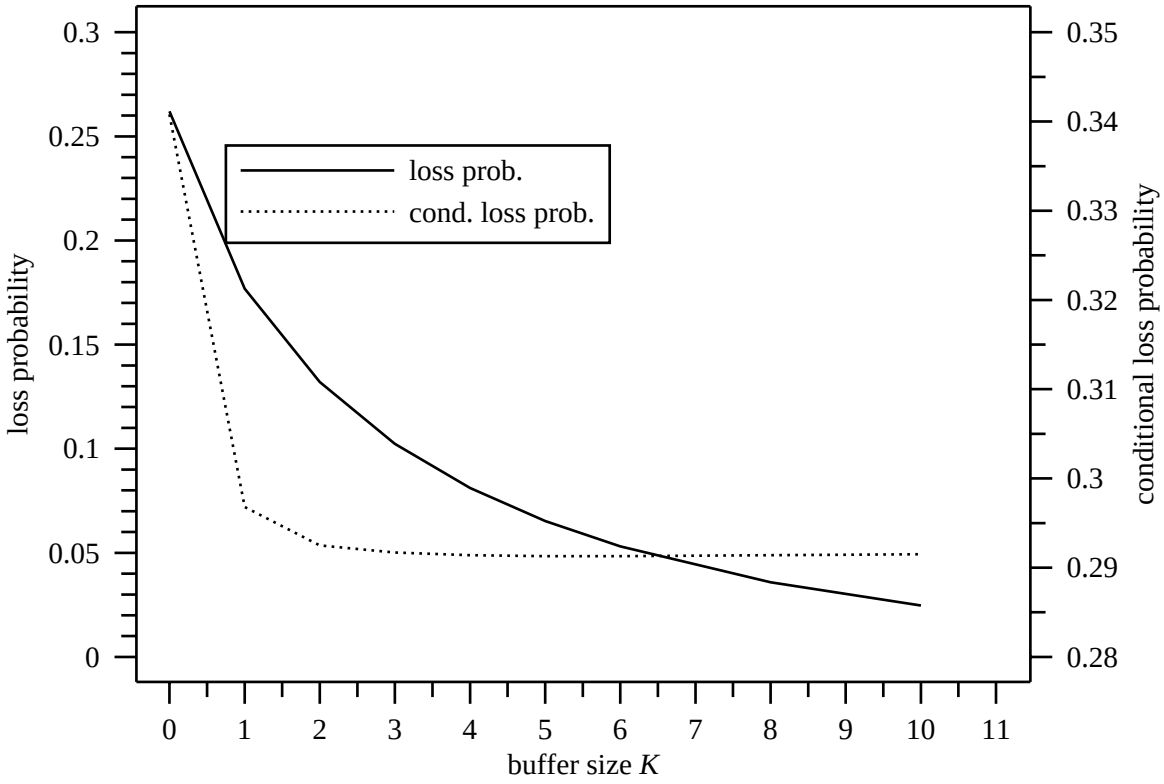

Figure 6.9. Conditional and unconditional loss probability for $N \cdot \mathrm{IPP}/D/c/K$ system, as a function of the buffer size, K; $N = 4$, $\lambda = 0.8$, $\gamma = 0.7$, $\omega = 0.9$

emits a short burst of densely packed packets, followed by a long silence. A lower value of λ, conversely leads to a smoother traffic process. Fig. 6.11 shows that the loss probability is far more affected by the change in traffic burstiness than is the conditional loss probability.

To provide a real-life example, the loss correlation in a DS1 voice multiplexer was determined. The DS-1/T1 channel offers a user capacity of 1.536 Mb/s. Following Heffes and Lucantoni [17], each source encodes speech in 64 kb/s and generates a 128-byte packet every 16 ms. Talk spurts and silence periods are geometrically distributed with means 352 ms and 650 ms, respectively, for an activity factor of 35%. When active, this source produces a packet every 24 slots. The channel processes 1324.2 packets per second. These system characteristics translate, at a load of 0.8,

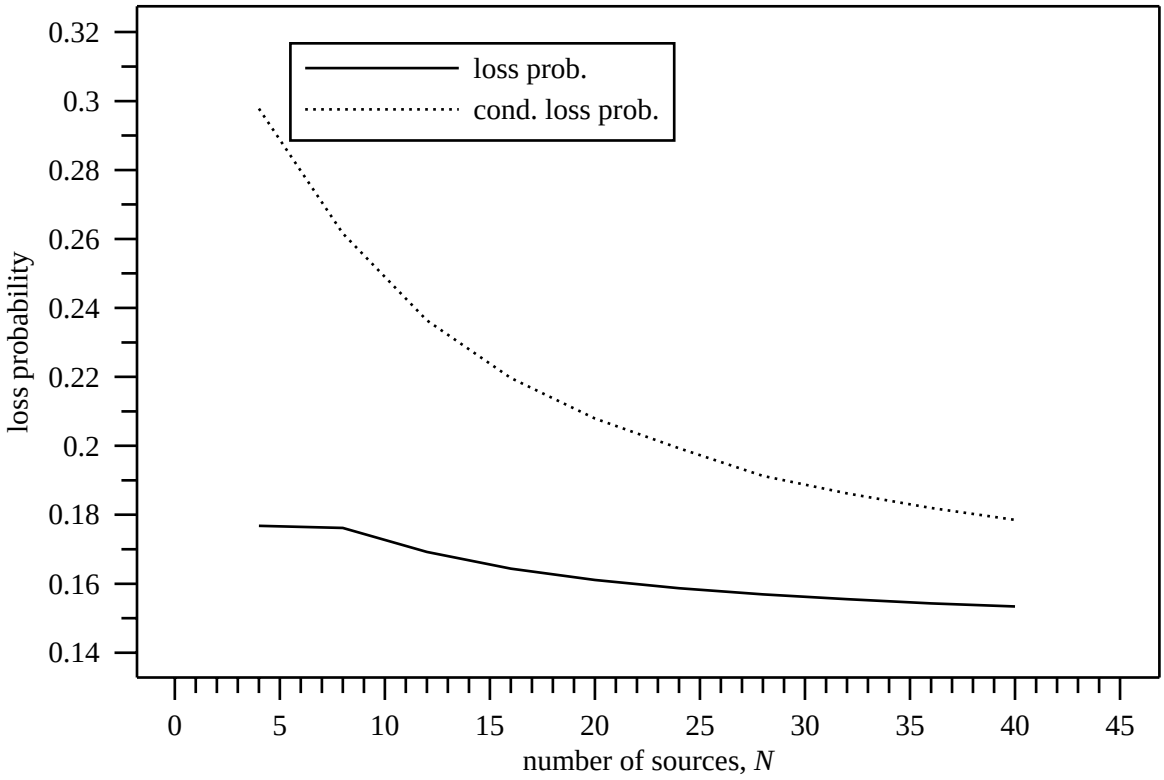

Figure 6.10. Conditional and unconditional loss probability for $N \cdot \mathrm{IPP}/D/c/K$ system, for constant activity, $\alpha = 0.25$, and load, $\rho = 0.8$, as a function of the number of sources, N; $K = 1$, $\lambda = 3.2/N$

to the model parameters $1 - \lambda = 0.04167$, $1 - \gamma = 2.145 \cdot 10^{-3}$, $\omega = 1.162 \cdot 10^{-3}$ and $N = 55$. At a buffer size of $K = 3$, the conditional loss probability evaluates to 10.83%, or a mean loss run length of 1.12. This computation required about 20 hours of DECstation 5000 CPU time, evaluating 1483 terms of the total probability summation. A simulation confirmed these results (90% confidence on intervals 5.93 to 6.06% for loss and 10.78 to 10.99% for conditional loss probability) and showed that the numerical accuracy is satisfactory.

For a smaller multiplexing factor, consider video sources on an STS-3 channel. The STS3 channel has a usable capacity of 150.336 Mb/s or 354566 ATM cells/s. Unlike voice sources, video sources have not been well characterized in terms of standard source models. We approximately model the CATV video coder of Verbiest

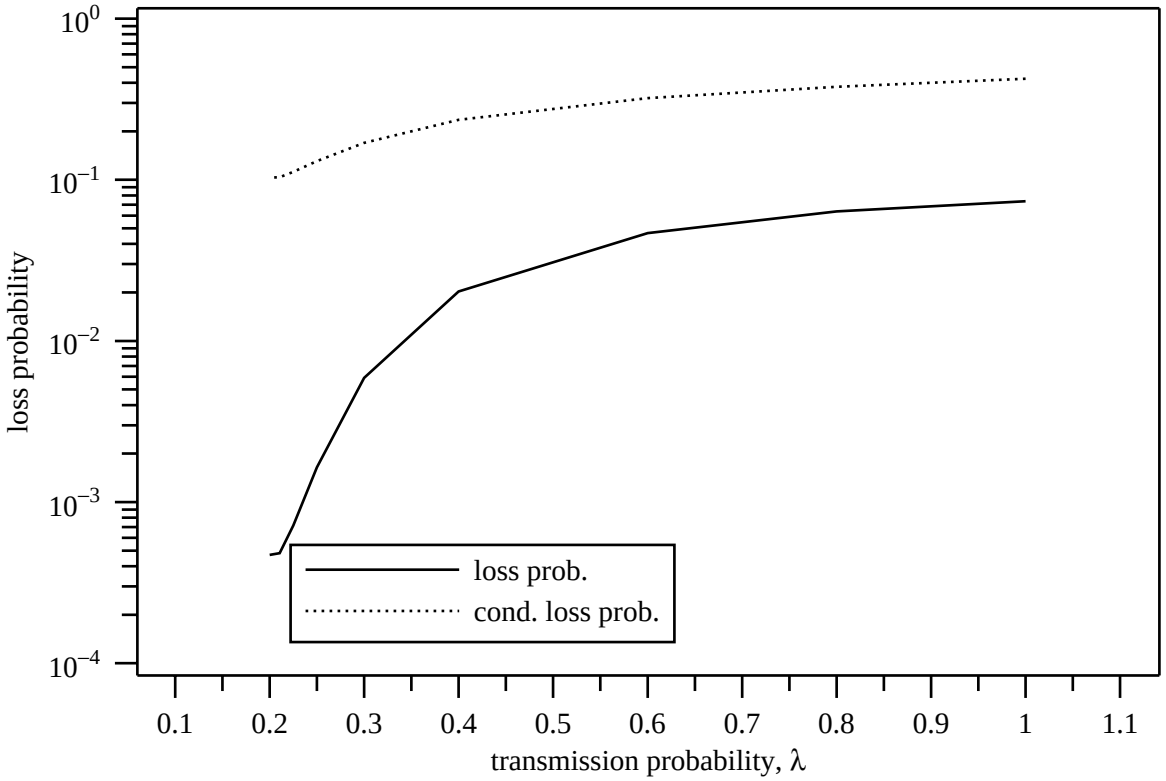

Figure 6.11. Conditional and unconditional loss probability for $N \cdot \mathrm{IPP}/D/c/K$ system, for constant load ($\rho = 0.8$), cycle time ($T_0 + T_1 = 25$) and buffer size ($K = 1$), as a function of λ

and Pinnoo [244]. It has an average rate of 26.5 Mb/s and a standard deviation of 3.4 Mb/s. We assume that the coder transmits at channel capacity and that the active and silent period add up to the frame duration of 1/25 s. We choose model parameters $\lambda = 1$, $1 - \gamma = 3.623 \cdot 10^{-4}$, $1 - \omega = 8.755 \cdot 10^{-5}$, and, again for a load of approximately 0.8, $N = 5$. With these parameters and $K = 1$, the conditional loss probability and mean loss run evaluate to the significantly higher values (compared to the voice example) of 0.5658 and 2.303, respectively. Thus, appropriate packet interleaving or priority mechanisms must be used to reduce the incidence of noticeable display artifacts.

6.4 Summary and Conclusions

In this chapter, we have attempted to provide both methodologies and traffic engineering insight. The tools provided should allow the computation of the loss correlation for a wide variety of sources of practical interest in high-speed networking. Also, for a range of systems, it was shown qualitatively that the buffer size or waiting time limit has virtually no influence on the loss correlation. For the periodic arrival case, we saw that as long as individual sources do not contribute more than, say, 10% of the total bandwidth, the average loss run lengths are not significantly larger than one, even though the conditional loss probability can be orders of magnitude higher than the time-averaged loss probability. As expected, the traffic burstiness and the number of multiplexed sources play significant roles in the loss correlation for the IPP system.

Throughout this chapter, we have used the conditional loss probability, or, equivalently, the average loss run length as an indicator of loss correlation. Another useful indicator could be the number of lost packets among a random sequence of packets with a given length. Also, the distribution of success runs deserves closer investigation. Particularly for queues with correlated inputs, Approximation methods need to be investigated that allow systems with large state spaces to be treated.

Chapter 7

Conclusion

We conclude this dissertation by summarizing the main points again, while outlining areas for future research. We highlight some of the dualities of real-time and data traffic, reflect on the methods of research used and stress the need to do end-to-end analysis.

High-bandwidth wide area network provide the foundation for a range of new services, many of which are characterized by end-to-end real-time constraints. Multimedia conferencing, for example, offers the promise of collaboration and social interaction unhindered by physical distance. Current packet networks, even if they have sufficient physical-layer capacity to carry these services, lack the control mechanisms to provide adequate quality of service to make them truly feasible.

Through a combination of queueing theoretical analysis, simulations and actual implementations, this dissertation has endeavored to fill in some of the missing pieces in an architecture for networks integrating real-time and non-real-time services. Our guiding concern was to provide *end-to-end* results rather than results characterizing individual system components.

Our research was motivated by the observation that many interesting applications envisioned for future high-speed integrated services networks would be characterized by rather different properties than the data transfer applications in common use today, namely loss tolerance and delay sensitivity. Our selective discarding algorithm makes use of that property by trading direct packet loss for reduced delay distribution tails.

Despite their differences, it has been realized in recent years that there exist interesting dualisms between real-time services and bulk data services. For example,

it could be argued that window-based flow control for data transfer applications makes a similar tradeoff as our controlled-loss algorithm, but in terms of throughput. By limiting throughput voluntarily and in a controlled manner, they prevent the uncontrolled and presumably larger loss of throughput caused by queue overflows at intermediate and end systems.

Another dual between real-time and data applications can be noted in relation to propagation delays. For data transfer applications, window-based flow control and appropriate selective retransmission mechanisms aim to make the throughput largely independent of the propagation delay within the network. Similarly, the proposed hop-laxity scheduling algorithm can help to compensate to some extent for longer propagation delays and larger hop counts in a real-time connection.

More recently, the dichotomy of traffic characteristics between controllable data traffic and basically unchangeable real-time sources has been reexamined. Originally, the perception was that, for example, audio and video traffic characteristic are determined by the content, with few possibilities for controlling its traffic characteristics. On the other hand, data traffic could be slowed down as necessary to relieve network congestion. Rate control appears to be a promising approach for data and real-time sources, in particular video [51]. Rate control for multimedia, however, takes the form of actually influencing the data that is carried in the network, rather than just pacing its entry into the network. The interest in rate control is an example of increased interaction between the network and the data source. Adaptive applications, made possible by sophisticated end systems such as workstations replacing "dumb" terminals and telephones, that adjust to the current network congestion are an example of fruitful interaction between the network and the receiver. This approach contrasts somewhat with the more adversarial approach implied by such terms as traffic contracts and policing[1]. These interactions between

[1]This does not imply that either are unnecessary.

the end systems and the network provide a new facet of meaning to the term "intelligent network" and appear to be a promising area of further research.

Data applications and real-time applications such as audio and video also share the property that packet losses can severely impact their performance. For real-time applications, propagation delays make retransmission often infeasible so that lost packets can translate directly into audible and visible impairments. For that reasons, many of these applications can recover from isolated packet losses, using forward error correction or data interpolation. However, the performance of these concealment techniques depend to a large measure on the temporal structure of packet loss, not just its rate. This dissertation has made contributions to more accurately predict the performance as seen by an application by allowing a detailed description of the packet loss behavior in time. The evaluation of scheduling policies should be extended to provide performance results after delay adaptation has taken place. Unfortunately, these results will likely involve a complex tradeoff between average delay seen by the application and loss due to deadline misses. Developing simple models that accurately predict end-to-end performance promises to be challenging. The large number of states needed to describe realistic traffic models for a single queue implies that more qualitative or approximate analyses are called for to understand loss correlation in multi-queue systems.

The use of multi-process workstations also challenges the assumptions underlying many traffic models for voice and video, as evidenced by the traffic data gathered in the course of experiments. Clearly, additional efforts must be made in modeling even supposedly well-understood traffic classes such as packet voice, in particular when source and network interact.

As our understanding of networks grow, we have learned that looking at single nodes in isolation provides at best an incomplete picture of network performance. In recent years, characterizing end-to-end network behavior has begun to attract

research interest, but much remains unknown particularly when sources, network nodes and receivers interact as described above.

A more complete understanding of emerging packet networks requires a range of approaches. Through the use of analysis, simulation and actual network implementation and measurements, the relative benefits and predictive power of all three methods and their combinations (such as trace driven simulation and statistical packet generation in real networks) could be evaluated in this dissertation. The high cost of running controlled experiments on real networks, both in terms of providing physical lines spanning a continent as well as the difficulty debugging "live" operating systems and of observing performance without changing the measurement results, suggests that guidelines that allow to relate simulation experiments to actual network performance are urgently needed. Also, configurable network emulators might provide a way station between simulation and integration into testbeds and operational networks.

Bibliography

[1] Jayant, Nuggehally S., "Effects of packet losses on waveform-coded speech," in *Proceedings of the Fifth International Conference on Computer Communications*, (Atlanta, Georgia), pp. 275–280, IEEE, Oct. 1980.

[2] Verbiest, Willem, Pinnoo, Luc, and Voeten, Bart, "The impact of the ATM concept on video coding," *IEEE Journal on Selected Areas in Communications*, vol. JSAC-6, pp. 1623–1632, Dec. 1988.

[3] Williamson, Carey L. and Cheriton, David R., "Loss-load curves: support for rate-based congestion control in high-speed datagram networks," in *Sigcomm '91 Conference: Communications Architectures and Protocols*, (Zürich, Switzerland), pp. 17–28, ACM, Sept. 1991.

[4] Mankin, Allison, "Random drop congestion control," in *SIGCOMM Symposium on Communications Architectures and Protocols*, (Philadelphia, Pennsylvania), pp. 1–7, ACM, Sept. 1990. in *Computer Communication Review*, Vol. 20, No. 4.

[5] Cassandras, Christos G., Kallmes, Michelle Hruby, and Towsley, Don, "Optimal routing and flow control in networks with real-time traffic," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Ottawa, Canada), pp. 784–791, IEEE, Apr. 1989.

[6] Hashem, Eman Salaheddin, "Analysis of random drop for gateway congestion control," Technical report MIT/LCS/TR-465, MIT Laboratory for Computer Science, Cambridge, Massachusetts, Nov. 1989.

[7] Ramakrishnan, K. K. and Jain, Raj, "A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer," in *Proceedings of the 1988 SIGCOMM Symposium on Communications Architectures and Protocols*, (Stanford, California), pp. 303–313, ACM, Aug. 1988.

[8] Mitra, Debasis and Mitrani, Isi, "Asymptotic optimality of the go-back-n protocol in high speed data networks with small buffers," in *Proceedings of the Fourth International Conference on Data Communication Systems and Their Performance*, (Barcelona), June 1990.

[9] Mitra, Debasis, "Optimal design of windows for high speed data networks," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (San Francisco, California), pp. 1156–1163, IEEE, June 1990.

[10] Mitra, Debasis, Mitrani, Isi, Ramakrishnan, K. G., Seery, J. B., and Weiss, Alan, "A unified set of proposals for control and design of high speed data networks," in *International Teletraffic Congress, Seventh Specialist Seminar*, (Morristown, New Jersey), pp. 12.4.1–12.4.8, ITC, Oct. 1990.

[11] Mitra, Debasis and Seery, Judith B., "Dynamic adaptive windows for high speed data networks with multiple paths and propagation delays," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Bal Harbour, Florida), pp. 39–48 (2B.1), IEEE, Apr. 1991.

[12] Fendick, Kerry W., Mitra, Debasis, Mitrani, Isi, Rodrigues, Manoel A., Seery, Judith B., and Weiss, Alan, "An approach to high-performance, high-speed data networks," *IEEE Communications Magazine*, vol. 29, pp. 74–82, Oct. 1991.

[13] Elwalid, Anwar I. and Mitra, Debasis, "Analysis and design of rate-based congestion control of high speed networks, I: stochastic fluid models, access regulation," *Queueing Systems*, vol. 9, pp. 29–64, Oct. 1991.

[14] Mitra, Debasis, Mitrani, I., Ramakrishnan, K. G., Seery, J. B., and Weiss, Alan, "A unified set of proposals for control and design of high-speed data networks," *Queueing Systems*, vol. 9, pp. 215–234, Oct. 1991.

[15] Zhang, Hui and Keshav, Srinivasan, "Comparison of rate-based service disciplines," in *Sigcomm '91 Symposium – Communications Architectures and Protocols*, (Zürich, Switzerland), pp. 113–121, ACM, Sept. 1991.

[16] Sriram, Kotikalapudi and Whitt, Ward, "Characterizing superposition arrival processes in packet multiplexers for voice and data," in *Proceedings of the Conference on Global Communications (GLOBECOM)*, (New Orleans, Louisiana), pp. 778–784 (25.4), IEEE, Dec. 1985.

[17] Heffes, Harry and Lucantoni, David M., "A Markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance," *IEEE Journal on Selected Areas in Communications*, vol. SAC-4, pp. 856–867, Sept. 1986.

[18] Sriram, K. and Lucantoni, David M., "Traffic smoothing effects of bit dropping in a packet voice multiplexer," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (New Orleans), pp. 759–769, IEEE, Mar. 1988.

[19] Dravida, S. and Sriram, K., "End-to-end performance models for variable bit rate voice over tandem links in packet networks," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Ottawa, Canada), pp. 1089–1097, IEEE, Apr. 1989.

[20] Li, San-qi and Mark, Jon W., "Simulation study of a network of voice/data integrated TDMs," *IEEE Transactions on Communications*, vol. COM-36, pp. 126–132, Jan. 1988.

[21] Bhargava, Amit, Kurose, James F., Towsley, Don, and van Lamport, Guy, "Performance comparison of error control schemes in high speed computer

communication networks," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (New Orleans), IEEE, Apr. 1988.

[22] Takahashi, Kenzo, Yokoi, Tadahiro, and Yamamoto, Yutaka, "Communications quality analysis for ATM networks," in *Conference Record of the International Conference on Communications (ICC)*, (Boston), pp. 423–427 (13.6), IEEE, June 1989.

[23] De Prycker, Martin, "Impact of data communication on ATM," in *Conference Record of the International Conference on Communications (ICC)*, (Boston), pp. 705–712 (22.4), IEEE, June 1989.

[24] Yin, Nanying, Li, San-qi, and Stern, Thomas E., "Congestion control for packet voice by selective packet discarding," in *Proceedings of the Conference on Global Communications (GLOBECOM)*, (Tokyo), pp. 1782–1786, IEEE, Nov. 1987.

[25] Yuan, Chin and Silvester, John A., "Queueing analysis of delay constrained voice traffic in a packet switching system," *IEEE Journal on Selected Areas in Communications*, vol. SAC-7, pp. 729–738, June 1989.

[26] Kim, Byung K. and Towsley, Don, "Dynamic flow control protocols for packet-switching multiplexers serving real-time multipacket messages," *IEEE Transactions on Communications*, vol. COM-34, pp. 348–356, Apr. 1986.

[27] Luan, Daniel T. and Lucantoni, David M., "Throughput analysis of a window-based flow control subject to bandwidth management," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (New Orleans), pp. 411–417, IEEE, Mar. 1988.

[28] Cohen, J. W., "Single server queues with restricted accessibility," *Journal of Engineering Mathematics*, vol. 3, pp. 265–284, Oct. 1969.

[29] Gavish, Bezalel and Schweitzer, Paul J., "The Markovian queue with bounded waiting time," *Management Science*, vol. 23, pp. 1349–1357, Aug. 1977.

[30] Schulzrinne, Henning, Kurose, James F., and Towsley, Don, "Congestion control for real-time traffic in high-speed networks," Technical Report TR 89-92, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, 1989.

[31] Doshi, Bharat T. and Heffes, Harry, "Overload performance of several processor queueing disciplines for the M/M/1 queue," *IEEE Transactions on Communications*, vol. COM-34, pp. 538–546, June 1986.

[32] Papoulis, Athanasios, *Probability, Random Variables, and Stochastic Processes*. New York, New York: McGraw-Hill Book Company, 2nd ed., 1984.

[33] Heidelberger, Philip and Welch, Peter D., "A spectral method for confidence interval generation and run length control in simulations," *Communications ACM*, vol. 24, pp. 233–245, Apr. 1981.

[34] Kobayashi, Hisashi, "Application of the diffusion approximation to queueing networks II: Nonequilibrium distributions and applications to computer modeling," *J. ACM*, vol. 21, pp. 459–469, July 1974.

[35] Kobayashi, Hisashi, *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology*. Reading, Massachusetts: Addison-Wesley, 1981.

[36] Clark, David D., Shenker, Scott, and Zhang, Lixia, "Supporting real-time applications in an integrated services packet network: architecture and mechanism," in *SIGCOMM Symposium on Communications Architectures and Protocols*, (Baltimore, Maryland), pp. 14–26, ACM, Aug. 1992.

[37] Kalmanek, C. R., Kanakia, H., and Keshav, Srinivasan, "Rate controlled servers for very high-speed networks," in *Proceedings of the Conference on Global Communications (GLOBECOM)*, (San Diego, California), pp. 12–20 (300.3), IEEE, Dec. 1990.

[38] Ferrari, Domenico and Verma, Dinesh C., "Scheme for real-time channel establishment in wide-area networks," *IEEE Journal on Selected Areas in Communications*, vol. 8, pp. 368–379, Apr. 1990.

[39] Ferrari, Domenico, Verma, Dinesh Chandra, and Zhang, Hui, "Guaranteeing delay jitter bounds in packet-switching networks," tech. rep., TENET Group, CS Division of the University of California and the International Computer Sciences Institute, Berkeley, California, 1991.

[40] Golestani, S. Jamaloddin, "A stop-and-go queueing framework for congestion management," in *Sigcomm '90: Communication Architectures and Protocols*, (Philadelphia, Pennsylvania), pp. 8–18, ACM, Sept. 1990.

[41] Golestani, S. Jamaloddin, "Congestion-free communication in broadband packet networks," in *Conference Record of the International Conference on Communications (ICC)*, (Atlanta, Georgia), pp. 489–494, IEEE, Apr. 1990.

[42] Golestani, S. Jamaloddin, "Congestion-free transmission of real-time traffic in packet networks," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (San Francisco, California), pp. 527–536, IEEE, June 1990.

[43] Golestani, S. Jamaloddin, "Congestion-free communication in high-speed packet networks," *IEEE Transactions on Communications*, vol. 39, pp. 1802–1812, Dec. 1991.

[44] Golestani, S. Jamaloddin, "Duration-limited statistical multiplexing of delay sensitive traffic in packet networks," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Bal Harbour, Florida), pp. 323–332 (4A.4), IEEE, Apr. 1991.

[45] Golestani, S. Jamaloddin, "A framing strategy for congestion management," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 1064–1077, Sept. 1991.

[46] Demers, Alan, Keshav, Srinivasan, and Shenker, Scott, "Analysis and simulation of a fair queueing algorithm," in *Proceedings of Sigcomm'89: Communication Architectures and Protocols*, (Austin, Texas), pp. 1–12, ACM, Sept. 1989.

[47] Parekh, Abhay K. and Gallager, Robert G., "A generalized processor sharing approach to flow control in integrated services networks: the multiple node code," manuscript, Massachusetts Institute of Technology, Cambridge, Massachusetts, Summer 1992. submitted to ACM/IEEE Transactions on Networking.

[48] Parekh, Abhay K. and Gallager, Robert G., "A generalized processor sharing approach to flow control in integrated services networks – the multiple node case," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, vol. 2, (San Francisco, California), pp. 521–530 (5a.1), IEEE, March/April 1993.

[49] Zhang, Lixia, "VirtualClock: A new traffic control algorithm for packet switching networks," in *SIGCOMM Symposium on Communications Architectures and Protocols*, (Philadelphia, Pennsylvania), pp. 19–29, ACM, Sept. 1990.

[50] Kurose, Jim, "On computing per-session performance bounds in high-speed multi-hop computer networks," in *Sigmetrics 1992*, (New Port, Rhode Island), pp. 128–139, ACM, June 1992.

[51] Reibman, Amy R. and Berger, Arthur W., "On VBR video teleconferencing over ATM networks," in *Proceedings of the Conference on Global Communications (GLOBECOM)*, (Orlando, Florida), pp. 314–319 (10.03), IEEE, Dec. 1992.

[52] Saltzer, J. H., Reed, D. P., and Clark, D. D., "End-to-end arguments in system design," *ACM Trans. Computer Systems*, vol. 2, pp. 277–288, Nov. 1984.

[53] Partridge, Craig, "Isochronous applications do not require jitter-controlled networks," Network Working Group Request for Comment RFC 1257, Swedish Institute of Computer Science (SICS), Kista, Sweden, Sept. 1991.

[54] Bala, Krishna, Cidon, Israel, and Sohraby, Khosrow, "Congestion control for high speed packet switched networks," in *Proceedings of the Conference*

on Computer Communications (IEEE Infocom), (San Francisco, California), pp. 520–526, IEEE, June 1990.

[55] Cohen, Danny, "A network voice protocol: NVP-II," technical report, University of Southern California/ISI, Marina del Ray, California, Apr. 1981.

[56] Cole, Randy, "PVP - a packet video protocol," W-Note 28, Information Sciences Institute, University of Southern California, Los Angeles, California, Aug. 1981.

[57] Kobza, John and Liu, Steve, "A head-of-line approximation to delay-dependent scheduling in integrated packet-switched networks," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Ottawa, Canada), pp. 1106–1113, IEEE, Apr. 1989.

[58] Kallmes, Michelle Hruby, Towsley, Don, and Cassandras, Christos G., "Optimality of the last-in-first-out (LIFO) service discipline in queueing systems with real-time constraints," in *Proceedings of the 28th Conference on Decision and Control (CDC)*, (Tampa, Florida), pp. 1073–1074, IEEE, 1989.

[59] van As, Harmen R., "Congestion control in packet switching networks by a dynamic foreground-background storage strategy," in *Performance of Computer-Communication Systems (Proceedings of the IFIP WG 7.3/TC 6 Second International Symposium on the Performance of Computer-Communication Systems)* (Bux, Werner and Rudin, Harry, eds.), (Zürich, Switzerland), pp. 433–448, IFIP, North-Holland, Mar. 1984.

[60] Kamoun, Farouk, "A drop and throttle control policy for computer networks," *IEEE Transactions on Communications*, vol. COM-29, pp. 444–452, Apr. 1981.

[61] Chen, Thomas M., Walrand, Jean, and Messerschmitt, David G., "Dynamic priority protocols for packet voice," *IEEE Journal on Selected Areas in Communications*, vol. SAC-7, pp. 632–643, June 1989.

[62] Lim, Youngho and Kobza, John, "Analysis of a delay-dependent priority discipline in a multi-class traffic switching node," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (New Orleans, Louisiana), pp. 889–898, IEEE, Mar. 1988.

[63] Peha, Jon M. and Tobagi, Fouad A., "Evaluating scheduling algorithms for traffic with heterogeneous performance objectives," in *Proceedings of the Conference on Global Communications (GLOBECOM)*, (San Diego, California), pp. 21–27, IEEE, Dec. 1990.

[64] Doshi, Bharat T. and Lipper, E. H., "Comparison of service disciplines in a queueing system with delay dependent customer behavior," in *Applied Probability – Computer Science: The Interface* (Disney, R. L. and Ott, T. J., eds.), vol. 2, Cambridge, Massachusetts: Birkhauser, 1982.

[65] Brady, Paul T., "A technique for investigating on-off patterns of speech," *Bell System Technical Journal*, vol. 44, pp. 1 – 22, Jan. 1965.

[66] Brady, Paul T., "A model for generating on-off speech patterns in two-way conversation," *Bell System Technical Journal*, vol. 48, pp. 2445–2472, Sept. 1969.

[67] Brady, Paul T., "A statistical analysis of on-off patterns in 16 conversations," *Bell System Technical Journal*, vol. 47, pp. 73–91, Jan. 1968.

[68] Karanam, V. R., Sriram, K., and Bowker, Duane O., "Performance evaluation of variable-bit-rate voice in packet-switched networks," *AT&T Technical Journal*, pp. 57–71, September/October 1988.

[69] Nakada, Hiroshi and Sato, Ken-Ichi, "Variable rate speech coding for asynchronous transfer mode," *IEEE Transactions on Communications*, vol. 38, pp. 277–284, Mar. 1990.

[70] Huggins, A. W. F., Viswanathan, R., and Makhoul, J., "Speech-quality testing of some variable-frame-rate (VFR) linear-predictive (LPC) vocoders," *Journal of the Acoustical Society of America*, vol. 62, pp. 430–434, Aug. 1977.

[71] Steele, Raymond and Benjamin, Frank, "Variable-length packetization of μ-law PCM speech," *AT&T Technical Journal*, vol. 64, pp. 1271–1292, July–August 1985.

[72] Steele, Raymond and Fortune, Peter, "An adaptive packetization strategy for A-law PCM speech," in *Conference Record of the International Conference on Communications (ICC)*, (Chicago, Illinois), pp. 941–945 (29.6), IEEE, June 1985.

[73] Kondo, Kazuhiro and Ohno, Masashi, "Variable rate embedded ADPCM coding scheme for packet speech on ATM networks," in *Proceedings of the Conference on Global Communications (GLOBECOM)*, (San Diego, California), pp. 523–527 (405.3), IEEE, Dec. 1990.

[74] Magill, D. T., "Adaptive speech compression for packet communication systems," in *Conference record of the IEEE National Telecommunications Conference*, (??), pp. 29D–1 – 29D–5, ?? 1973.

[75] Sherman, D. N., "Storage and delay estimates for asynchronous multiplexing of data in speech," *IEEE Transactions on Communications*, vol. COM-19, pp. 551–555, Aug. 1971.

[76] Yatsuzuka, Yohtaro, "Highly sensitive speech detector and high-speed voiceband data discriminator in DSI-ADPCM systems," *IEEE Transactions on Communications*, vol. COM-30, pp. 739–750, Apr. 1982.

[77] Mahmoud, Samy A., Chan, Wai-Yip, Riordon, J. Spruce, and Aidarous, Salah E., "An integrated voice/data system for VHF/UHF mobile radio," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, pp. 1098–1111, Dec. 1983.

[78] Haccoun, David, Cohen, Paul, and Hai-Hoc, Hoang, "An experimental investigation of the active-idle pattern of speech over land mobile radio telephone channels," *IEEE Transactions on Vehicular Technology*, vol. VT-32, pp. 260–268, Nov. 1983.

[79] Gruber, John G., "A comparison of measured and calculated speech temporal parameters relevant to speech activity detection," *IEEE Transactions on Communications*, vol. COM-30, pp. 728–738, Apr. 1982.

[80] Gruber, John G. and Strawczynski, Leo, "Subjective effects of variable delay and speech clipping in dynamically managed voice systems," *IEEE Transactions on Communications*, vol. COM-33, pp. 801–808, Aug. 1985.

[81] Campanella, S. Joseph, "Digital speech interpolation techniques," in *Conference record of the IEEE National Telecommunications Conference*, vol. 1, (Birmingham, Alabama), pp. 14.1.1 – 14.1.5, IEEE, Dec. 1978.

[82] Rieser, J. H., Suyderhood, H. G., and Yatsuzuka, Y., "Design considerations for digital speech interpolation," in *Conference Record of the International Conference on Communications (ICC)*, (Denver, Colorado), pp. 49.4.1 – 49.4.7, IEEE, June 1981.

[83] Ashmend, Rais and Fatehchand, Richard, "Effect of sample duration on the articulation of sounds in normal and clipped speech," *The Journal of the Acoustical Society of America*, vol. 31, pp. 1022–1029, July 1959.

[84] Nagarajan, Ramesh, Kurose, James F., and Towsley, Don, "Approximation techniques for computing packet loss in finite-buffered voice multiplexers," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (San Francisco, California), pp. 947–955, IEEE, June 1990.

[85] Nagarajan, Ramesh, Kurose, James F., and Towsley, Don, "Approximation techniques for computing packet loss in finite-buffered voice multiplexers," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 368–377, Apr. 1991.

[86] Brochin, Frank M. and Thomas, John B., "Voice transmission in packet switching networks: a model for queueing analysis," in *26th Annual Allerton Conference on Communication, Control and Computing*, (Monticello, Illinois), pp. 1001–1004, Sept. 1988.

[87] Berger, Arthur W., "Overload control using rate control throttle: Selecting token bank capacity for robustness to arrival rates," in *Proceedings of the*

IEEE Conference on Decision and Control, (Tampa, Florida), pp. 2527–2529, IEEE, Dec. 1989.

[88] Ahmadi, Hamid, Guérin, Roch, and Sohraby, Khoshrow, "Analysis of leaky bucket access control mechanism with batch arrival process," in *Proceedings of the Conference on Global Communications (GLOBECOM)*, (San Diego, California), pp. 344–349, IEEE, Dec. 1990.

[89] Butto´, Milena, Cavallero, Elisa, and Tonietti, Alberto, "Effectiveness of the "leaky bucket" policing mechanism in ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 335–342, Apr. 1991.

[90] Cooper, C. Anthony and Park, Kun I., "Toward a broadband congestion control strategy," *IEEE Network*, vol. 4, pp. 18–23, May 1990.

[91] Rathgeb, Erwin P., "Modeling and performance comparison of policing mechanisms for ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 325–334, Apr. 1991.

[92] Escobar, Julio, Deutsch, Debra, and Partridge, Craig, "Flow synchronization protocol," in *Proceedings of the Conference on Global Communications (GLOBECOM)*, (Orlando, Florida), pp. 1381–1387 (40.04), IEEE, Dec. 1992.

[93] Little, T. D. C., Ghafoor, A., Chen, C. Y. R., Chang, C.S., and Berra, P. B., "Multimedia synchronization," *The Quarterly Bulletin of the IEEE Computer Society Technical Committe on Data Engineering*, vol. 14, pp. 26–35, Sept. 1991.

[94] Anderson, David P. and Homsy, George, "A continuous media I/O server and its synchronization mechanism," *IEEE Computer*, vol. 24, pp. 51–57, Oct. 1991.

[95] Auzimoor, P., Hazard, L., Horn, F., Lacroix, D., and Stefani, J. B., "An analysis of multimedia restitution and its architectural impact," in *First International Workshop on Network and Operating System Support for Digital Audio and Video*, (Berkeley, California), 1990. TR-90-062.

[96] Yavatkar, Raj, "Issues of coordination and temporal synchronization in multimedia communication (extended abstract)," *ACM Computer Communication Review*, vol. 22, pp. 77–78, Mar. 1992.

[97] Li, Li, Karmouch, A., and Georganas, N. D., "Real-time synchronization control in multimedia distributed systems," *ACM Computer Communication Review*, vol. 22, pp. 79–87, Mar. 1992.

[98] Little, T. D. C., "Protocols for bandwidth-constrained multimedia traffic," *ACM Computer Communication Review*, vol. 22, pp. 47–48, Mar. 1992.

[99] Steinmetz, Ralf and Meyer, Thomas, "Multimedia synchronization techniques: experiences based on different system structures," *ACM Computer Communication Review*, vol. 22, pp. 90–91, Mar. 1992.

[100] Campbell, Andrew, Coulson, Geoff, García, Francisco, and Hutchison, David, "A continuous media transport and orchestration service," in *SIGCOMM Symposium on Communications Architectures and Protocols* (Sidhu, Deepinder P., ed.), (Baltimore, Maryland), pp. 99–110, ACM, Aug. 1992. in *Computer Communication Review* 22 (4), Oct. 1992.

[101] DARPA/ISI, *DARTnet planning and review workshop*, (Marina del Ray, California), Dec. 1991.

[102] Deering, Stephen E. and Cheriton, David R., "Multicast routing in datagram internetworks and extended LANs," *ACM Trans. Computer Systems*, vol. 8, pp. 85–110, May 1990.

[103] Deering, Steve, "Host extensions for IP multicasting," Network Working Group Request for Comments RFC 1054, Stanford University, May 1988.

[104] Deering, Steve, "Host extensions for IP multicasting," Network Working Group Request for Comments RFC 1112, Stanford University, Aug. 1989.

[105] Comer, Douglas E., *Internetworking with TCP/IP*, vol. 1. Englewood Cliffs, New Jersey: Prentice Hall, 1991.

[106] Postel, Jon, "Internet protocol," Network Working Group Request for Comments RFC 791, Information Sciences Institute, Sept. 1981.

[107] Topolcic, Claudio, Casner, Stephen, Lynn, Charles, Jr., Park, Philippe, and Schroder, Kenneth, "Experimental internet stream protocol, version 2 (ST-II)," Network Working Group Request for Comments RFC 1190, BBN Systems and Technologies, Oct. 1990.

[108] Postel, John, "Internet control message protocol," Network Working Group Request for Comments RFC 792, ISI, Sept. 1981.

[109] Zhang, Lixia, Deering, Steve, Estrin, Deborah, Shenker, Scott, and Zappala, Daniel, "Rsvp: a new Resource ReSerVation Protocol." preliminary draft (anon. ftp), Mar. 1993.

[110] Mills, David L., "Network time protocol (version 3) – specification, implementation and analysis," Network Working Group Request for Comments RFC 1305, University of Delaware, Mar. 1992.

[111] Mills, David L., "Network time protocol (version 2) — specification and implementation," Network Working Group Request for Comments RFC 1119, University of Delaware, Sept. 1989.

[112] Mills, David L., "Internet time synchronization: the network time protocol," *IEEE Transactions on Communications*, vol. 39, pp. 1482–1493, Oct. 1991.

[113] Leffler, Samuel J., McKusick, Marshall Kirk, Karels, Michael J., and Quarterman, John S., *The Design and Implementation of the 4.3BSD UNIX Operating System*. Reading, Massachusetts: Addison-Wesley, 1988.

[114] Zhang, Lixia, *A New Architecture for Packet Switched Network Protocols*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, July 1989.

[115] Weinrib, Abel and Wu, L. T., "Virtual clocks and leaky buckets: flow control protocols for high-speed networks," in *Proceedings of the IFIP Working Group WG 6.1 and WG 6.4 Second International Workshop on Protocols for High-Speed Networks* (Johnson, Marjory J., ed.), pp. 3–15, IFIP, 1991.

[116] Goli, Praveen, Kurose, James F., and Towsley, Donald F., "Approximate minimum laxity scheduling algorithms for real-time systems," Technical Report TR 90-88, Department of Computer and Information Sciences, University of Massachusetts, Amherst, Massachusetts, Jan. 1990.

[117] Partridge, Craig and Pink, Stephen, "An implementation of the revised Internet Stream Protocol (ST-2)," in *Second International Workshop on Network and Operating System Support for Digital Audio and Video*, (Heidelberg, Germany), ACM Sigcomm, Nov. 1991.

[118] Merritt, Ian H., "Providing telephone line access to a packet voice network," Research Report ISI/RR-83-107, Information Sciences Institute (ISI), Marina del Ray, California, Feb. 1983.

[119] Schulzrinne, Henning, "Voice communication across the internet: A network voice terminal," Technical Report TR 92-50, Dept. of Computer Science, University of Massachusetts, Amherst, Massachusetts, July 1992.

[120] Garrett, Mark W. and Vetterli, Martin, "Congestion control strategies for packet video," in *Fourth International Workshop on Packet Video*, (Kyoto, Japan), Aug. 1991.

[121] Cohen, D., "On packet speech communication," in *Proceedings of the Fifth International Conference on Computer Communications*, (Atlanta, Georgia), pp. 271–274, IEEE, Oct. 1980.

[122] Forgie, James W., "Voice conferencing in packet networks," in *Conference Record of the International Conference on Communications (ICC)*, (Seatle, Washington), pp. 21.3.1–21.3.4, IEEE, June 1980.

[123] Shacham, Nachum, Craighill, Earl J., and Poggio, Andrew A., "Speech transport in packet-radio networks with mobile nodes," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, pp. 1084–1097, Dec. 1983.

[124] Chan, Wai Yip, "A speech detector for mobile radio," Master's thesis, Carleton University, Ottawa, Ontario, Canada, Sept. 1982.

[125] Falk, Gilbert, Groff, Stephen J., Milliken, Walter C., Nodine, Marian, Blumenthal, Steven, and Edmond, Winston, "Integration of voice and data in the wideband packet satellite network," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, pp. 1076–1083, Dec. 1983.

[126] Cohen, Danny, "Specification for the network voice protocol (NVP)," Network Working Group Request for Comment RFC 741, ISI, Jan. 1976.

[127] CCITT, "Draft recommendation G.PVNP: Packetized voice networking protocol," 1989. Appendix 2 to Annex 1 of Question 24/XV (COM XV-1-E).

[128] Schulzrinne, Henning, "Issues in designing a transport protocol for audio and video conferences and other multiparticipant real-time applications." Internet draft (work-in-progress) *draft-ietf-avt-issues-*.txt*, Dec. 1992.

[129] Schulzrinne, Henning, "A transport protocol for real-time applications." Internet draft (work-in-progress) *draft-ietf-avt-rtp-*.txt*, Dec. 1992.

[130] Donofrio, Edward J., "The experimental communication system for voice/data on the Ethernet," in *Proceedings of the Conference on Global Communications (GLOBECOM)*, vol. 2, (New Orleans, Louisiana), pp. 848–851 (28.2), IEEE, Dec. 1985.

[131] Friedman, Eluzor and Ziegler, Chaim, "Real-time voice communications over a token-passing ring local area network," in *SIGCOMM Symposium on Communications Architectures and Protocols*, (Stowe, Vermont), pp. 52–57, ACM, Aug. 1986.

[132] Friedman, Eluzor and Ziegler, Chaim, "Packet voice communications over PC-based local area networks," *IEEE Journal on Selected Areas in Communications*, vol. 7, pp. 211–218, Feb. 1989.

[133] Ades, Stephen, Want, Roy, and Calnan, Roger, "Protocols for real time voice communication on a packet local network," in *Conference Record of the International Conference on Communications (ICC)*, (Toronto, Canada), pp. 525–530 (17.1), IEEE, June 1986.

[134] Brandsma, J. R., Bruekers, A. A. M. L., and Kessels, J. L. W., "PHILAN: a fiber-optic ring for voice and data," *IEEE Communications Magazine*, vol. 24, pp. 16–22, Dec. 1986.

[135] Casey, L. M., Dittburner, R. C., and Gamage, N. D., "FXNET: a backbone ring for voice and data," *IEEE Communications Magazine*, vol. 24, pp. 23–28, Dec. 1986.

[136] Limb, John O. and Flamm, Lois E., "A distributed local area network packet protocol for combined voice and data transmission," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, pp. 926–934, Nov. 1983.

[137] Soares, L. F. G., Martins, S. L., Bastos, T. L. P., Ribeiro, N. R., and Cordeiro, R. C. S., "LAN based real time audio-data system," in *Conference on Office Automation Systems*, vol. 11, (Cambridge, Massachusetts), pp. 152–157, ACM, Apr. 1990.

[138] Schooler, Eve M. and Casner, Stephen L., "A packet-switched multimedia conferencing system," *SIGOIS (ACM Special Interest Group on Office Information Systems) Bulletin*, vol. 10, pp. 12–22, Jan. 1989.

[139] Corley, L. T., "Bellsouth trial of wideband packet technology," in *Conference Record of the International Conference on Communications (ICC)*, vol. 3, (Atlanta, Georgia), pp. 1000–1002 (324.2), IEEE, Apr. 1990.

[140] Barberis, Guilio, Calabrese, Mario, Lambarelli, Livio, and Roffinella, Daniele, "Coded speech in packet-switched networks: Models and experiments," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, pp. 1028–1038, Dec. 1983.

[141] Kapaun, A. A., Leung, W.-H. F., Luderer, G. W. R., Morgan, M. J., and Vaidya, A. K., "Wideband packet access for workstations: integrated voice/data/image services on the Unix PC," in *Proceedings of the Conference on Global Communications (GLOBECOM)*, vol. 3, (Houston, Texas), pp. 1439–1441 (40.6), IEEE, Dec. 1986.

[142] Bowker, D. O. and Dvorak, C. A., "Speech transmission quality of wideband packet technology," in *Proceedings of the Conference on Global Communications (GLOBECOM)*, vol. 3, (Tokyo, Japan), pp. 1887 – 1889 (47.7), IEEE, Nov. 1987.

[143] Spilling, Paal and Craighill, Earl, "Digital voice communications in the packet radio network," in *Conference Record of the International Conference on Communications (ICC)*, (Seattle, Washington), pp. 21.4.1–21.4.7, IEEE, June 1980.

[144] Muise, R. W., Schonfeld, T. J., and Zimmerman III, G. H., "Experiments in wideband packet technology," in *Proceedings of the International Seminar*, (Zürich), pp. 135–139, North-Holland/IEEE, Mar. 1986.

[145] Steinberg, Daniel and Rua, Monica, "Desktop audio at Sun Microsystems," in *American Voice Input/Output Society Conference*, (Minneapolis, Minnesota), AVIOS, Sept. 1992.

[146] Casner, Stephen and Deering, Stephen, "First IETF Internet audiocast," *ACM Computer Communication Review*, vol. 22, pp. 92–97, July 1992.

[147] Montgomery, Warren A., "Techniques for packet voice synchronization," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, pp. 1022–1028, Dec. 1983.

[148] Suda, Tatsuya, Miyahara, Hideo, and Hasegawa, Toshiharu, "Performance evaluation of a packetized voice system — simulation study," *IEEE Transactions on Communications*, vol. COM-34, pp. 97–102, Jan. 1984.

[149] Gopal, Prabandham M., Wong, J. W., and Majithia, J. C., "Analysis of playout strategies for voice transmission using packet switching techniques," *Performance Evaluation*, vol. 4, pp. 11–18, Feb. 1984.

[150] Ma, Joong and Gopal, Inder, "A blind voice packet synchronization strategy," Research Report RC 13893, IBM, T. J. Watson Research Center, Yorktown Heights, New York, July 1988.

[151] Jayant, Nuggehally S. and Noll, Peter, *Digital Coding of Waveforms*. Englewood Cliffs, New Jersey: Prentice Hall, 1984.

[152] Schulzrinne, Henning, Kurose, James F., and Towsley, Don, "Congestion control for real-time traffic in high-speed networks," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (San Francisco, California), pp. 543–550, June 1990.

[153] Berger, Arthur W., "Overload control in star networks: Comparison of percent blocking throttle and LIFO queue discipline." Working paper from AT&T Bell Laboratories, 1989.

[154] Forys, L. J., "Performance analysis of a new overload strategy," in *Proceedings of the Tenth International Teletraffic Congress (ITC-10)*, (Montreal), p. 5.24, IAC, North Holland, June 1983.

[155] Gruber, John G. and Le, Nguyen H., "Performance requirements for integrated voice/data networks," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, pp. 981–1005, Dec. 1983.

[156] Jayant, Nuggehally S., "Effects of packet losses in waveform-coded speech," in *International Conference on Computers and Communications*, (Atlanta, Georgia), pp. 275–280, IEEE, Oct. 1980.

[157] DaSilva, Luiz A., Petr, David W., and Frost, Victor S., "A class-oriented replacement technique for lost speech packets," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Ottawa, Canada), pp. 1098–1105, IEEE, Apr. 1989.

[158] Shacham, Nachum, "Packet resequencing under reliable transport protocols," in *Hawaii International Conference on System Sciences*, vol. 3, (Kailua-Kona, Hawaii), pp. 716–723, Jan. 1989.

[159] Shacham, Nachum and McKenney, Paul, "Packet recovery in high-speed networks using coding and buffer management," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (San Francisco, California), pp. 124–131, IEEE, June 1990.

[160] Gilbert, Edgar N., "Capacity of a burst-noise channel," *Bell System Technical Journal*, vol. 39, pp. 1253–1265, Sept. 1960.

[161] Kaul, A. K., "Performance of high-level data link control in satellite communications," *COMSAT Technical Review*, vol. 8, pp. 41–88, Spring 1978.

[162] Fujiwara, F. C., Kasahara, M., Yamashita, K., and Namekawa, T., "Evaluations of error-control techniques in both independent and dependent-error channels," *IEEE Transactions on Communications*, vol. COM-26, pp. 785–793, June 1978.

[163] Towsley, Don, "A statistical analysis of ARQ protocols operating in a non-independent error environment," *IEEE Transactions on Communications*, vol. COM-29, pp. 971–981, July 1981.

[164] Leung, C. H. C, Kikumoto, Y., and Sorensen, S. A., "The throughput efficiency of the go-back-*n* ARQ scheme under Markov and related error structures," *IEEE Transactions on Communications*, vol. 36, pp. 231–234, Feb. 1988.

[165] Pieris, Gerard R. and Sasaki, Galen H., "The performance of simple error control protocols under correlated packet losses," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Bal Harbour, Florida), pp. 764–772 (7C.1), IEEE, Apr. 1991.

[166] Ohta, Hiroshi and Kitami, Tokuhiro, "Simulation study of the cell discard process and the effect of cell loss compensation in ATM networks," *The Transactions of the IEICE*, vol. E73, pp. 1704–1711, Oct. 1990.

[167] Lazar, Aurel A., Pacifici, Giovanni, and White, John S., "Real-time traffic measurements on MAGNET II," *IEEE Journal on Selected Areas in Communications*, vol. 8, pp. 467–483, Apr. 1990.

[168] Kitami, Tokuhiro and Tokizawa, Ikuo, "Cell loss compensation schemes in an asynchronous broadband ISDN," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (San Francisco, California), pp. 116–123, IEEE, June 1990.

[169] Biersack, Ernst W., "Performance evaluation of forward error correction in ATM networks," in *SIGCOMM Symposium on Communications Architectures and Protocols* (Sidhu, Deepinder P., ed.), (Baltimore, Maryland), pp. 248–257, ACM, Aug. 1992. in *Computer Communication Review* 22 (4), Oct. 1992.

[170] Kamitake, Takashi and Suda, Tatsuya, "Evaluation of an admission control scheme for an ATM network considering fluctuations in cell loss rate," in *Proceedings of the Conference on Global Communications (GLOBECOM)*, (Dallas, Texas), pp. 1774–1780, IEEE, Nov. 1989.

[171] Leland, Will E., "Window-based congestion management in broadband ATM networks: the performance of three access-control policies," in *Proceedings of the Conference on Global Communications (GLOBECOM)*, (Dallas, Texas), pp. 1794–1800, IEEE, Nov. 1989.

[172] Woodruff, Gillian M. and Kositpaiboon, Rungroj, "Multimedia traffic management principles for guaranteed ATM network performance," *IEEE Journal on Selected Areas in Communications*, vol. 8, pp. 437–446, Apr. 1990.

[173] Ferrandiz, Josep M. and Lazar, Aurel A., "Consecutive packet loss in real-time packet traffic," in *Proceedings of the Fourth International Conference on Data Communication Systems*, (Barcelona), pp. 306–324, IFIP TC6, June 1990.

[174] Ferrandiz, Josep M. and Lazar, Aurel A., "A study of loss in $N/GI/1$ queueing systems." Department of Electrical Engineering and Center for Telecommunications Research, Columbia University, New York, New York, Jan. 1990.

[175] Ferrandiz, Josep M. and Lazar, Aurel A., "Rate conservation for stationary processes," *Journal of Applied Probability*, vol. 28, pp. 146–158, Mar. 1991.

[176] van Doorn, Erik A., "On the overflow process from a finite Markovian queue," *Performance Evaluation*, vol. 4, pp. 233–240, Nov. 1984.

[177] Meier-Hellstern, Kathleen S., "Parcel overflows in queues with multiple inputs," in *Proceedings of the 12th International Teletraffic Congress (ITC)* (Bonatti, Mario, ed.), (Torino, Italy), pp. 1359–1366, North-Holland, June 1988.

[178] Meier-Hellstern, Kathleen S., "The analysis of a queue arising in overflow models," *IEEE Transactions on Communications*, vol. 37, pp. 367–372, Apr. 1989.

[179] Norros, I. and Virtamo, J. T., "Who loses cells in the case of burst scale congestion?," in *Teletraffic and Datatraffic in a Period of Change* (Jensen, Arne and Iversen, V. B., eds.), (Copenhagen, Denmark), pp. 829–834, North-Holland, June 1991.

[180] Schulzrinne, Henning, Kurose, James F., and Towsley, Don, "Distribution of the loss period for some queues in continuous and discrete time," Technical Report TR 91-03, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, 1991.

[181] Li, San-qi, "Study of information loss in packet voice systems," *IEEE Transactions on Communications*, vol. 37, pp. 1192–1202, Nov. 1989.

[182] Kleijnen, Jack P. C., *Statistical Tools for Simulation Practitioners.* New York, New York: Marcel Dekker, 1987.

[183] Cidon, Israel and Gopal, Inder S., "PARIS: An approach to integrated high-speed private networks," *International Journal of Digital and Analog Cabled Networks*, vol. 1, pp. 77–85, April–June 1988.

[184] Eng, K. Y., Hluchyj, M. G., and Yeh, Y. S., "A knockout switch for variable-length packets," in *Conference Record of the International Conference on Communications (ICC)*, (Seattle, Washington), pp. 794–799, IEEE, June 1987.

[185] Wolff, Ronald W., *Stochastic Modeling and the Theory of Queues.* Englewood Cliffs, New Jersey: Prentice Hall, 1989.

[186] Kleinrock, Leonard, *Queueing Systems — Theory*, vol. 1. New York, New York: Wiley-Interscience, 1975.

[187] Ross, Sheldon M., *Stochastic Processes.* New York, New York: John Wiley and Sons, 1983.

[188] Kleinrock, Leonard, *Queueing Systems — Computer Applications*, vol. 2. New York, New York: Wiley-Interscience, 1976.

[189] Prabhu, Narahari Umanath, *Stochastic Storage Processes — Queues, Insurance Risk, and Dams*, vol. 15 of *Applications of Mathematics.* New York, New York: Springer-Verlag, 1980.

[190] Kemperman, J. H. B., *The Passage Problem for a Stationary Markov Chain.* Chicago: University of Chicago Press, 1961.

[191] Moran, P. A. P., "A probability theory of a dam with a continuous release," *The Quarterly Journal of Mathematics – Oxford Second Series*, vol. 7, pp. 130–137, June 1956.

[192] Moran, P. A. P., *The Theory of Storage.* Methuen's Monographs on Applied Probability and Statistics, London/New York, New York: Methuen/Wiley, 1959.

[193] Saaty, Thomas L., *Elements of Queueing Theory.* New York, New York: Dover Publications (originally published by McGraw-Hill), 1983/1961.

[194] Prabhu, Narahari Umanath, *Queues and Inventories — A Study of Their Basic Stochastic Processes.* New York, New York: John Wiley, 1965.

[195] Takács, Lajos, *Stochastic Processes — Problems and Solutions.* London/New York, New York: Methuen/Wiley, 1960.

[196] Karlin, Samuel and Taylor, Howard M., *A First Course in Stochastic Processes.* San Diego, California: Academic Press, 2nd ed., 1975.

[197] Bhat, U. Narayan, *Elements of Applied Stochastic Processes.* New York, New York: John Wiley, 2nd ed., 1984.

[198] Karol, Mark J., Hluchyj, Michael G., and Morgan, Samuel P., "Input versus output queueing on a space-division packet switch," *IEEE Transactions on Communications*, vol. COM-35, pp. 1347–1356, Dec. 1987.

[199] Yeh, Yu-Shuan, Hluchyj, Michael G., and Acampora, Anthony S., "The knockout switch: A simple, modular architecture for high-performance packet switching," *IEEE Journal on Selected Areas in Communications*, vol. SAC-5, pp. 1274–1282, Oct. 1987.

[200] Hunter, Jeffrey J., *Mathematical Techniques of Applied Probability – Discrete Time Models: Techniques and Applications*, vol. 2. New York, New York: Academic Press, 1983.

[201] Takács, Lajos, *Combinatorial Methods in the Theory of Stochastic Processes.* New York, New York: John Wiley, 1967.

[202] Moran, P. A. P., *An Introduction to Probability Theory.* Oxford, Great Britain: Clarendon Press, 1968.

[203] Feller, William, *An Introduction to Probability Theory and Its Applications*, vol. 1. New York, New York: John Wiley and Sons, third ed., 1968.

[204] Dor, Neville M., "Guide to the length of buffer storage required for random (Poisson) input and constant output rates," *IEEE Transactions on Electronic Computers*, vol. EC-16, pp. 683–684, Oct. 1967.

[205] Hluchyj, Michael G. and Karol, Mark J., "Queueing in high-performance packet switching," *IEEE Journal on Selected Areas in Communications*, vol. SAC-6, pp. 1587–1597, Dec. 1988.

[206] Birdsall, T. G., Ristenbatt, M. P., and Weinstein, S. B., "Analysis of asynchronous time multiplexing of speech sources," *IRE Transactions on Communication Systems*, vol. CS-10, pp. 390–397, Dec. 1962.

[207] Tran-Gia, Phuoc and Ahmadi, Hamid, "Analysis of a discrete-time $G^{[X]}/D/1-S$ queueing system with applications in packet-switching systems," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (New Orleans), pp. 861–870 (9A.1), IEEE, Mar. 1988.

[208] Lin, Arthur Y. M. and Silvester, John A., "Queueing analysis of an ATM switch with multichannel transmission groups," Tech. Rep. CRI 89-25, Computer Engineering Division, Electrical Engineering-Systems Department, University of Southern California, Los Angeles, California, 1989.

[209] Clare, Loren P. and Rubin, Izhak, "On the design of prioritized multiplexing systems," in *Conference Record of the International Conference on Communications (ICC)*, (Boston, Massachusetts), pp. 1344–1348 (E5.3), IEEE, June 1983.

[210] Clare, Loren P. and Rubin, Izhak, "Preemptive buffering disciplines for time-critical sensor communications," in *Conference Record of the International Conference on Communications (ICC)*, (Toronto, Canada), pp. 904–909, IEEE, June 1986.

[211] Yin, Nanying and Hluchyj, Michael G., "Implication of dropping packets from the front of a queue," in *International Teletraffic Congress, Seventh Specialist Seminar*, (Morristown, New Jersey), p. 10.4, ITC, Oct. 1990.

[212] Schulzrinne, Henning and Kurose, James F., "Distribution of the loss period for some queues in continuous and discrete time," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Bal Harbour, Florida), pp. 1446–1455 (12C.1), Apr. 1991.

[213] Study Group XVIII, CCITT (International Telegraph and Telephone Consultative Committee), "Study group XVIII - report R 34," June 1990.

[214] Joos, P. and Verbiest, W., "A statistical bandwidth allocation and usage monitoring algorithm for ATM networks," in *Conference Record of the International Conference on Communications (ICC)*, (Boston), pp. 415–422 (13.5), IEEE, June 1989.

[215] Bhargava, Amit, Humblet, Pierre, and Hluchyj, Michael G., "Queueing analysis of continuous bit-stream transport in packet networks," in *Proceedings of the Conference on Global Communications (GLOBECOM)*, (Dallas, Texax), pp. 903–907, IEEE, Nov. 1989.

[216] Kaplan, Michael, "The queue $D/D/1$ with a Poisson background," in *Conference Record of the International Conference on Communications (ICC)*, (Toronto, Canada), pp. 36.5.1–36.5.4, IEEE, June 1978.

[217] Sahin, Izzet and Bhat, U. Narayan, "A stochastic system with scheduled secondary inputs," *Operations Research*, vol. 19, pp. 436–446, March-April 1971.

[218] Sahin, Izzet, "Equilibrium behavior of a stochastic system with secondary input," *Journal of Applied Probability*, vol. 8, pp. 252–260, June 1971.

[219] Eckberg, Adrian E., Jr., "The single server queue with periodic arrival process and deterministic service times," *IEEE Transactions on Communications*, vol. COM-27, pp. 556–562, Mar. 1979.

[220] Karol, Mark J. and Hluchyj, Michael G., "Using a packet switch for circuit-switched traffic: A queueing system with periodic input," in *Conference Record of the International Conference on Communications (ICC)*, (Seattle, Washington), pp. 1677–1682 (48.3), IEEE, June 1987.

[221] Virtamo, J. T. and Roberts, J. W., "Evaluating buffer requirements in an ATM multiplexer," in *Proceedings of the Conference on Global Communications (GLOBECOM)*, (Dallas, Texas), pp. 1473–1477, IEEE, Nov. 1989.

[222] Fredericks, A. A., "Congestion in blocking systems — a simple approximation technique," *Bell System Technical Journal*, vol. 59, pp. 805–827, July–August 1980.

[223] Kuczura, Anatol, "Loss systems with mixed renewal and Poisson inputs," *Operations Research*, vol. 21, pp. 787–795, May-June 1973.

[224] Matsumoto, Jun and Watanabe, Yu, "Individual traffic characteristics of queueing systems with multiple Poisson and overflow inputs," *IEEE Transactions on Communications*, vol. COM-33, pp. 1–9, Jan. 1985.

[225] Machihara, Fumiaki, "On the overflow processes from the $PH_1 + PH_2/M/S/K$ queue with two independent ph-renewal inputs," *Performance Evaluation*, vol. 8, pp. 243–253, Aug. 1988.

[226] Kekre, Hemant B., Saxena, C. L., and Khalid, Mohd, "Buffer behavior for mixed arrivals and single server with random interruptions," *IEEE Transactions on Communications*, vol. COM-28, pp. 59–64, Jan. 1980.

[227] Kuczura, Anatol, "Queues with mixed renewal and Poisson inputs," *Bell System Technical Journal*, vol. 51, pp. 1305–1326, July–August 1972.

[228] Gopinath, B. and Morrison, J. A., "Discrete-time single server queues with correlated inputs," *Bell System Technical Journal*, vol. 56, pp. 1743–1768, Nov. 1977.

[229] Chang, Cheng-Shang, Chao, XiuLi, and Pinedo, Michael, "Integration of discrete-time correlated Markov processes in a TDM system," *Probability in the Engineering and Informational Sciences*, vol. 4, pp. 29–56, Jan. 1990.

[230] Parzynski, William R. and Zipse, Philip W., *Introduction to Mathematical Analysis*. New York, New York: McGraw-Hill, 1982.

[231] Burke, P. J., "Delays in single-server queues with batch input," *Operations Research*, vol. 23, pp. 830–833, July–August 1975.

[232] Sriram, Kotikalapudi, Varshney, Pramod K., and Shanthikumar, J. George, "Discrete-time analysis of integrated voice/data multiplexers with and without speech activity detection," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, pp. 1124–1132, Dec. 1983.

[233] Li, San-qi and Mark, Jon W., "Performance of voice/data integration on a TDM system," *IEEE Transactions on Communications*, vol. COM-33, pp. 1265–1273, Dec. 1985.

[234] Li, San-qi and El Zarki, Magda, "Dynamic bandwidth allocation on a slotted ring with integrated services," *IEEE Transactions on Communications*, vol. 36, pp. 826–833, July 1988.

[235] Maglaris, Basil, Anastassiou, Dimitris, Sen, Prodip, Karlsson, Gunnar, and Robbins, John D., "Performance models of statistical multiplexing in packet video communications," *IEEE Transactions on Communications*, vol. 36, pp. 834–844, July 1988.

[236] Kuczura, Anatol, "The interrupted Poisson process as an overflow process," *Bell System Technical Journal*, vol. 52, pp. 437–448, Mar. 1973.

[237] Rath, John H. and Sheng, Diane, "Approximations for overflow from queues with a finite waiting room," *Operations Research*, vol. 27, pp. 1208–1216, November–December 1979.

[238] Murata, Masayaki, Oie, Yuji, Suda, Tatsuya, and Miyahara, Hideo, "Analysis of a discrete-time single-server queue with bursty inputs for traffic control in ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. 8, pp. 447–458, Apr. 1990.

[239] Ohba, Yoshihiro, Murata, Masayuki, and Miyahara, Hideo, "Analysis of interdeparture processes for bursty traffic in ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 468–476, Apr. 1991.

[240] Mangulis, V., *Handbook of Series for Scientists and Engineers.* New York: Academic Press, 1965.

[241] Gihr, O. and Tran-Gia, Phuoc, "A layered description of ATM cell traffic streams and correlation analysis," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Bal Harbour, Florida), pp. 137–144 (2D.4), IEEE, Apr. 1991.

[242] Bae, Jaime Jungok, Suda, Tatsuya, and Simha, Rahul, "Analysis of individual packet loss in a finite buffer queue with heterogeneous Markov modulated arrival processes: a study of traffic burstiness and priority packet discarding.," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, vol. 1, (Florence, Italy), pp. 219–230 (2C.1), IEEE, May 1992.

[243] Bae, Jaime Jungok, Suda, Tatsuya, and Simha, Rahul, "Heterogeneous arrival streams, burstiness and packet discarding: A study of individual packet loss," Technical Report 91-58, Dept. of Computer and Information Science at the University of California, Irvine, Irvine, California, July 1991.

[244] Verbiest, Willem and Pinnoo, Luc, "A variable bit rate codec for asynchronous transfer mode networks," *IEEE Journal on Selected Areas in Communications*, vol. 7, pp. 761–770, June 1989.