PRERECORDED SOURCES IN BROADBAND NETWORKS

Martin Reisslein

A Dissertation

in

Systems Engineering

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

1998

Prof. Keith W. Ross

Supervisor of Dissertation

Prof. Tony Smith

Graduate Group Chairperson

Acknowledgements

First and foremost I would like to thank my advisor Prof. Keith W. Ross for his support and guidance over the last three years. I greatly cherish the hours on end we spent discussing the research presented in this thesis. His thorough reviews of the individual chapters improved the presentation tremendously. His help in publishing the work presented herein has been invaluable. I am also indebted to him for introducing me to a great many fellow researchers in the networking area.

The research efforts summarized in this thesis have greatly benefited from discussions with Srini Rajagopal. Vincent Verillotte helped me with the implementation of the simulation of the decentralized prefetching protocol. I am also grateful to my former office mates Scott Malcolm, Jean McManus and Bill Liang for their support during my early days in the Systems Engineering department.

Finally I would like to thank the friends — to numerous to mention by name — that I made during my time at PENN, they have made my time here a most memorable experience.

Abstract

Prerecorded Sources in Broadband Networks

Martin Reisslein

Supervisor: Prof. Keith W. Ross

We study how the emerging broadband networks can efficiently carry traffic from prerecorded multimedia sources. Toward this end we first develop call admission policies for statistically multiplexing prerecorded sources over a bufferless transmission link. We develop a stochastic model which captures the random phases of the sources. We then apply large deviation theory to our model to develop admission rules. The accuracy of the large deviation approximation is verified with simulation experiments employing importance sampling techniques.

We next present a high–performance prefetching protocol for the delivery of prerecorded VBR video from a server across a packet-switched network to a large number of clients. The protocol requires (1) that each client have a small amount of memory dedicated to the application (2) that there is one bottleneck shared link between the server and the clients. The server chooses prefetched frames according to a join–the–shortest–queue (JSQ) policy. We also develop a variation of the the JSQ prefetching protocol that allows for multiple servers that are distributed deeper into the network.

We finally study call admission control and traffic mangement of regulated multimedia traffic. We develop a refined call admission control procedure for buffered multiplexers. This procedure involves searching over the buffered multiplexers' buffer–bandwidth tradeoff curve and finding the most adversarial traffic pattern of the regulated sources. Importantly, we show that on–off is not the most adversarial traffic pattern of leaky bucket constraint sources. We next propose a traffic management scheme for multimedia traffic which consists of smoothers at the network ingresses and bufferless statistical multiplexing within the network. We explicitly characterize the smoothers that maximize the connection carrying capacity of the network. We develop call admission rules for our smoothing/bufferless multiplexing scheme that are based on the assumption of adversarial traffic at the smoother

outputs.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Traditional queueing theory assumes that the traffic is random and generated on the fly. This traditional model is appropriate for live sources such as a video conference or the broadcast of a sporting event. Broadband networks of the near future, however, will increasingly carry traffic from prerecorded sources. These sources include high–fidelity audio, such as an orchestral symphony, short multimedia clips, such as information on a specific consumer product, and video–on–demand (VoD), such as the transfer of a full–length movie from a video server to a user's television. In fact, it is possible that in the upcoming years the majority of traffic in broadband data networks will emanate from prerecorded sources. It is therefore crucial to manage the transmission and transport of the traffic from these sources so that the network and end–system resources are efficiently utilized and that users receive satisfactory service.

Our research applies to arbitrary prerecorded sources, but in order to fix ideas we shall assume throughout this thesis that the prerecorded sources are video sources. Our research explicitly assumes that the videos are VBR encoded with high peak–to–mean ratios. The motivation for our approach is that, for the same perceived video quality, Constant Bit Rate (CBR) encoding produces an output rate significantly higher than the average rate of the corresponding VBR encoding for action movies [9]. CBR traffic allows for nearly 100% link utilization; the number of connections that can be carried over a link of given capacity is roughly the link capacity divided by the CBR rate (assuming homogeneous connections). The number of VBR connections that can be transmitted simultaneously is

the achievable link utilization multiplied by the link capacity divided by the average rate of the VBR video stream. Therefore schemes for transmitting VBR encoded video that achieve high average link utilizations while keeping losses at a negligible level, can allow for significantly more video connections than does CBR video.

The majority of the work on VBR video assumes that the traffic is generated by a *live source*, such as a video conference or a sporting event. The statistics of these live sources are typically not known in advance. The network is therefore forced to allocate resources in a conservative manner to live sources. This approach leads typically to low utilization of network resources. We contend that prerecorded VBR sources have special properties which are not shared by their live counterparts and which, when fully exploited, can lead to more efficient use of network resources. These properties are: (1) the traffic in each video frame is known before the video session begins; (2) while the video is being played, some of the video can be prefetched into the receiver memory; (3) the users may tolerate a small initial delay from when the video is ordered until when it appears on the screen. Because we believe that prerecorded VBR traffic will constitute much, if not the majority, of traffic in broadband networks, we feel that it is important to learn how to exploit these three properties in order to efficiently utilize network and end–system resources.

The traditional best–effort service of the Internet does not provide any Quality of Service (QoS) assurances to connections. This best–effort concept works well for services that do not rely on reliable real–time delivery such as e–mail, file transfer via FTP or transfer of web pages via HTTP. In order to provide more sophisticated services such as high–quality real–time audio and video connections, however, it is imperative that the Internet provides certain QoS guarantees such as a bound on the loss or maximum delay in the network. An emerging concept that facilitates the provisioning of QoS guarantees is *regulated traffic*. It requires a connection requesting a specific QoS to advertise a set of regulator constraints, typically a set of leaky bucket parameters, to the network. The network bases call admission control and the provisioning of network resources, such as buffer and bandwidth, solely on these leaky bucket parameters and the requested QoS. In order to be able to guarantee the requested QoS the network conservatively assumes that the source sends the worst–case or adversarial traffic pattern, that is, the traffic pattern that maximizes the loss in the network while still satisfying the advertised regulator constraints.

The characterization of the adversarial traffic pattern given a set of leaky bucket parameters is still considered an open issue in the networking community. We address this problem in Chapter 5.

Motivated by the expected dominance of prerecorded traffic in the Internet of the future and the acceptance of regulated traffic concepts in the standards bodies of the networking community [19, 69] we work towards a traffic management scheme that allows for the efficient transport of prerecorded sources within the framework of regulated traffic. We envision a traffic management scheme that provides stringent QoS guarantees and is thus able to deliver high–quality prerecorded multimedia content. At the same time we strive to utilize the network resources efficiently, that is, we strive to achieve high average link utilizations, thus maximizing the connection carrying capacity of the network.

1.1 Literature Review

Work in the area of traffic management of prerecorded video falls into two classes: traffic management issues in the disk subsystem of the video server (e.g., see [21] [22] [55] [73] [76]) and traffic management issues in the network. Although the ideas of this dissertation may be useful for disk subsystem design, our focus is on management of prerecorded VBR traffic in the network. The traffic management schemes for VBR video in the literature fall into four main categories: deterministic; deterministic with smoothing and/or prefetching; probabilistic; and probablistic with collaborative prefetching; see Figure 1.1. The principal performance metrics for all of these schemes are average link utilization, initial delay, delays after interactive actions, and client buffer size. The deterministic schemes send into the network the original VBR traffic, and admission control ensures that the delays never exceed a prespecified limit [75][34][40]. For highly variable VBR traffic, these deterministic schemes typically require large initial delays to achieve moderate link utilizations [45]. The deterministic schemes with prefetching and smoothing do not send the original VBR traffic into the network, but instead send some smoothed version of it. CRTT [46] is the extreme case of such a scheme, whereby the server transmits packets into a reserved CBR connection at the average rate of the video; such CBR connections are available from ATM and are being proposed for the Internet [68]. Although CRTT produces close to 100%

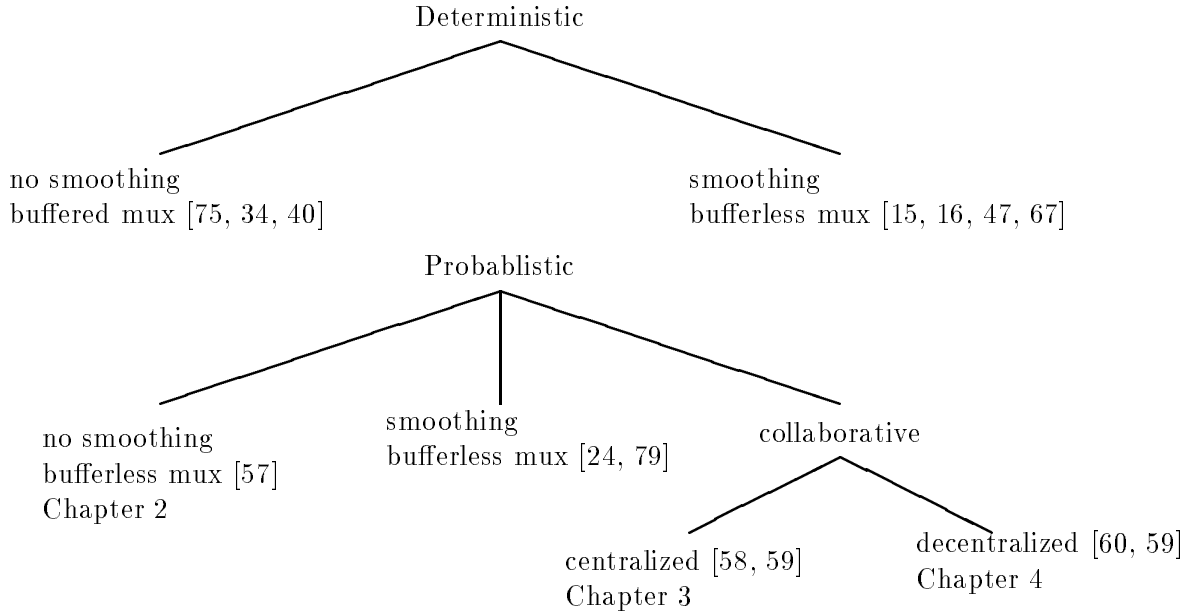


Figure 1.1: Transmission Schemes for VBR Video on Demand.

link utilizations, it has potentially a long initial delay and potentially long delays after temporal jumps. Several independent research teams have proposed schemes whereby the server transmits the video at different constant rates over different intervals; these schemes vary in how the rates and intervals are chosen [24] [67] [47] [15] [16]. As compared with CRTT, these schemes trade off link utilizations and simplicity for lower initial delay and lower interactivity delays. None of the deterministic schemes (with or without prefetching and smoothing) allows for both high link utilizations (>90%) and consistently high responsiveness (less than a second) to interactivity.

In Chapter 2 of this dissertation (see also [57]) we consider sending the original VBR encoded video into an unbuffered multiplexer. This scheme allows for responsive interactivity, but introduces packet loss whenever the aggregate transmission rate exceeds the link rate. In [24] and [79] related ideas are explored whereby the original traffic is first smoothed before it is statistically multiplexed at an unbuffered link; the statistically multiplexing of the smoothed traffic can substantially increase link utilization at the expense of small packet loss probabilities. In particular, in [79] the authors demonstrate that their

prefetching scheme, *Optimal Smoothing*, can give moderately high link utilizations when it is combined with statistical multiplexing.

We next introduce a probablistic transmission scheme with collaborative prefetching, *Join–the–Shortest–Queue (JSQ) prefetching* in Chapter 3 (see also [58]). We show that JSQ prefetching has substantially less packet loss than does Optimal Smoothing for the same link utilization. JSQ prefetching achieves nearly 100% link utilization, immediate commencement of playback and instantaneous response to viewer interactions. JSQ prefetching, however, can only be applied when one centralized server feeds many clients. In Chapter 4 we introduce a decentralized and collaborative prefetching protocol that allows the video streams to emanate from multiple distributed and decentralized servers (see also [60]).

There exists a rich set of literature on regulated traffic. Cruz in his seminal papers [6, 7, 8] defines regulated traffic and develops a calculus for analyzing networks fed by sources satisfying regulator constraints. He derives bounds on delay and buffering requirements in packet switched networks. He assumes that a sources' traffic stream satisfies specific "burstiness constraints" that constrain the amount of traffic entering the network over any time interval to a value that depends on the length of the interval. His analysis is purely deterministic and assumes worst–case or adversarial source behavior. Chang [3] examines stability conditions for queues fed by regulated sources and compares the stability conditions of deterministic queueing networks with those of stochastic queueing networks.

Several researchers have espoused the problem of finding the worst–case source behavior of leaky bucket constraint sources. It is a common belief that on–off is the worst–case traffic patterns, that is, it is believed that on–off pattern maximize the loss at a buffered multiplexer. Proofs in support of this belief are attempted by Mitra and Morrison [48] and Worster [74]. Recent results by Doshi [12] and Oechslin [50], however, indicate that tristate patterns cause more loss than on–off patterns. We address this problem in Chapter 5 (see also [54]) and find that on–off is not the most adversarial source behavior.

Elwalid *et al.* [14] and LoPresti *et al.* [42] develop call admission procedures for buffered multiplexers fed by regulated sources. Elwalid *et al.* [14] in their seminal paper transform the buffered multiplexer into an unbuffered multiplexer and then estimate the loss probability using Large Deviation techniques. LoPresti *et al.* [42] consider two resources —

buffer and bandwidth — and develop a refined call admission rule. We extend and refine the rule of LoPresti *et al.* in Chapter 5 (see also [54]).

Georgiadis *et al.* [23], Peris [52], Zhang [77] and Zhang and Ferrari [78] study rate–controlled service disciplines that afford a connection deterministic end–to–end delay guarantees in high speed networks These disciplines assume that a connections' traffic entering the network satisfies specific regulator constraints. Each connections' traffic is reshaped by a traffic regulator (also called traffic shaper) at every hop. The traffic shapers smooth out the bursts that have been introduced by the buffered multiplexer at the upstream node.

Knightly *et al.* [31, 34, 35, 75] characterize prerecorded video sources by cascaded leaky bucket regulators and develop deterministic call admission rules. They investigate the fundamental limits in providing deterministic QoS to video characterized by regulator functions. Their deterministic schemes while ensuring lossless transmission and thus excellent video quality allow only for modest network utilizations. We develop a traffic management scheme in Chapter 6 that like the schemes of Knightly *et al.* bases its call admission decision on cascaded leaky bucket characterizations of the prerecorded videos. Unlike the schemes of Knightly *et al.*, however, our scheme exploits the independence of the ongoing video streams. The video streams are first smoothed at the network ingress and then statistically multiplexed with minute loss probabilities within the network. We demonstrate that our scheme can carry many more video connections than the deterministic schemes of Knightly *et al.*

1.2 Organization of the Dissertation

We study how the emerging broadband networks can efficiently carry traffic from prerecorded sources. Toward this end we develop call admission policies for statistically multiplexing prerecorded sources over a bufferless transmission link in Chapter 2. Our model is appropriate for video on demand as well as other on–demand multimedia applications. In particular we allow users to specify when the sources begin transmission; we also allow the user to invoke VCR actions such as pause and temporal jumps. We suppose that the Quality of Service (QoS) requirement allows for a small amount of packet loss. We develop a stochastic model which captures the random phases of the sources. We then

apply large deviation theory to our model to develop global admission rules. The accuracy of the large deviation approximation is verified with simulation experiments employing importance sampling techniques. Numerical results are presented for the Star Wars trace. Finally, we develop efficient schemes for the real–time implementation of our global test. In particular, we demonstrate that the Taylor series expansion of the logarithmic moment generating function of the frame size distribution allows for fast and accurate admission decisions.

In Chapter 3 we present a high–performance prefetching protocol for the delivery of prerecorded VBR video from a server across a packet–switched network to a large number of clients. Not only does the protocol give constant perceptual quality and almost 100 % link utilization, but it also allows for immediate commencement of the video upon user request and near instantaneous response to pause/resume and temporal jumps. The protocol requires (1) that each client have a small amount of memory dedicated to the application (2) that there is one bottleneck shared link between the server and the clients. The protocol is based on the observation that there are frequent periods of time during which the shared link's bandwidth is under utilized. During these periods the server can prefetch frames from any of the ongoing videos and send the frames to the buffers in the appropriate clients. The server chooses prefetched frames according to a join–the–shortest–queue (JSQ) policy. We present simulation results of our prefetch policy that are based on MPEG encoded videos. Our simulations show that JSQ prefetching performs favorably as compared with the smoothing–based protocols in the existing literature.

In Chapter 4 we present a variation of the the JSQ prefetching protocol that allows for multiple servers that are distributed deeper into the network, the decentralized prefetching protocol. The protocol requires that (1) the client has a moderate amount of memory dedicated to the VoD application (2) the client sends a positive acknowledgment back to the server for each received video frame. Our decentralized prefetching protocol employs window flow control. A send window limits the number of frames a server is allowed to send in a frame period. The send window grows larger than one when the network is underutilized, allowing the server to prefetch future frames into the client memory. When the network becomes congested the send window is reduced and the server is throttled. Simulation results based on MPEG encoded videos show that our decentralized prefetching

protocols compare favorably with other prefetching protocols in the existing literature.

In Chapter 5 we consider a finite–buffer multiplexer to which traffic arrives from several independent regulated sources. As in Chapter 2 we develop call admission rules for the multiplexer. The work in this chapter, however, differs from Chapter 2 in two respects. First, the multiplexer in Chapter 2 is unbuffered while the multiplexer studied in this chapter has a finite buffer. This buffering capability complicates the analysis tremendously. Secondly, we assume that the moment generating function of the amount of traffic arriving to the multiplexer is known in Chapter 2. In this Chapter we assume that we have only a sources' cascaded leaky bucket characterization. We compare our call admission control rule for the buffered multiplexer with the call admission control rules in the literature.

In Chapter 6 we develop traffic management schemes for regulated multimedia traffic in networks. We propose a pragmatic scheme for multimedia traffic which consists of (i) cascaded leaky–buckets for traffic regulation, (ii) smoothers at the network edges, and (iii) bufferless statistical multiplexers within the network. For this scheme we show that loss probabilities are minimized with simple one–buffer smoothers which operate at specific minimum rates. We also show that the worst–case input traffic is extremal on–off traffic for all connections. These two results lead to a straightforward scheme for guaranteeing QoS to regulated traffic. Our scheme guarantees that the fraction of a connections' traffic that is delayed by more than a connection–specific limit is less than some minute number. We base admission control on a connections' leaky bucket parameters. We also develop and evaluate a heuristic for finding the optimal leaky bucket characterization of a prerecorded source. By employing smoothers at the network ingress and statistical multiplexing within the network we can carry many more connections than the deterministic schemes in the literature. Using MPEG video traces, we present numerical results which demonstrate the methodology.

Chapter 2

Call Admission for Prerecorded Sources

2.1 Overview

In recent years there has been an explosion of research in packetized VBR video, with the great majority of the work addressing *live video* such as video conference and the broadcast of a sporting event. While this research on live video certainly merits the attention it has received, much (if not the majority) of the video carried on high–speed packet–switched networks will emanate from *prerecorded sources*. These sources include full–length movies, music video clips, and educational material. From the perspective of the transport network, prerecorded VBR video sources are fundamentally different from live video sources: for live video, the exact dynamics of the VBR traffic are unknown; for prerecorded sources, the amount of traffic in each frame is known before the frames are transmitted into the network.

In this chapter we develop call–admission policies for statistically multiplexing prerecorded sources over a single transmission link. Our model is appropriate for video on demand (VoD) as well as other multimedia and on demand applications. However to fix ideas we will assume that all sources are video sources. We suppose that the transmission link is bufferless, so that packet loss occurs whenever the sum of the traffic rates, over the videos in progress, is greater than the link rate. We permit the users to begin the videos at random independent times; we also permit the users to pause and force temporal jumps

(rewind and fast–forward). We refer to pause and temporal jumps as VCR features.

We shall consider two QoS requirements. To define these QoS requirements, let ϵ be a fixed positive number, where a typical value of ϵ is 10^{-6}. The first QoS requirement is that the expected fraction of time during which cell loss occurs must be less than ϵ. The second QoS requirement is the expected fraction of bits lost must be less than ϵ.

An *ideal admission policy* will accept a new video connection if and only if the QoS requirement will continue to hold with the additional connection. For live video, one is typically forced to employ a conservative admission policy since one never knows with certainty the type of traffic the live videos will generate. In this chapter we show how to construct an ideal admission policy for prerecorded video which can be efficiently implemented in real time.

Recently several research groups have proposed schemes for the *lossless* multiplexing of prerecorded traffic. McManus and Ross [46] [47] [45], Kesidis and Hung [28], and Salehi *et al.* [67] have proposed schemes which use receiver memory and preplay delay. The principal feature of these schemes is that they produce high link utilizations, thereby reducing the network transport cost. But these schemes also require a more expensive receiver (due to the additional receiver memory) and are not easily amenable to temporal jumps. Knightly *et al.* [75], Knightly and Zhang [33, 34], and Liebeherr [41][40] also propose schemes for lossless multiplexing for prerecorded sources; their approach is to place a buffer before the transmission link and admit new connections as long as the buffer is guaranteed to never overflow. It is shown in McManus and Ross [45] that unless the link buffer is large (implying a large playback delay), this last set of schemes give low link utilizations when the traffic is highly variable (such as in action movies).

Elwalid *et al.* [14] study admission control policies for statistically multiplexing VBR sources over a single buffered link; they assume the traffic is leaky–bucket controlled and organized in classes. The class structure is not appropriate for VoD systems supporting a large number of videos; moreover, leaky buckets provide a loose bound on video traffic and result in pessimistic admission decisions.

When a video experiences a VCR action, its phase alignment changes with respect to the other videos being transmitted over the link. In Section 2.2 we develop a novel stochastic model for prerecorded traffic with random phases, a model which captures the

random occurrences of VCR actions in the videos. In Section 2.3 we apply the central limit theorem and large deviation theory to our stochastic model to develop connection admission rules. These rules admit new connections only if it is highly likely that the QoS requirements will be satisfied for the duration of the video, even if the videos experience VCR actions. We refer to these admission rules as *global rules* since they account for the long–term expected behavior of the video traffic. In Section 2.4 we present the result of our numerical experiments with an MPEG encoding of *Star Wars*. We first develop an importance sampling heuristic for estimating cell loss with Monte Carlo simulation. We then show that one of our approximations is extremely accurate and therefore leads to an ideal admission policy. We also find the statistical multiplexing gain to be high, especially when each video is smoothed over each of its Group of Pictures (GOPs). A brute force implementation of our ideal admission policy can be computationally prohibitive. In Section 2.5 we describe two modifications which significantly reduce the amount of on–line computation that is needed for admission control. In Section 2.6 we introduce a refinement of our admission control procedure which takes both a global and myopic view of the traffic offered to the link. This policy admits a new connection only if (1) there will be negligible cell loss in (say) the minute following call admission, assuming that no VCR actions occur; and (2) the QoS requirements are likely to be met over a long period of time (say, an hour). We find that an additional myopic test does not significantly reduce link utilization. We summarize our findings in Section 2.7.

2.2 Modeling Prerecorded Traffic with Random Phase Shifts

As mentioned in the Overview of this chapter, we assume that each video can experience VCR actions. We model these interactive features by associating with each video in progress an independent and random phase shift. We begin with some notation.

Consider J video streams multiplexed over a bufferless link with transmission rate C bps; see Figure 2.1. For simplicity assume that each video has N frames and has a frame rate of F frames per second. Let $x_n(j)$ be the number of bits in the nth encoded frame $(1 \leq n \leq N)$ of the jth video. Because we suppose that all videos are prerecorded, the

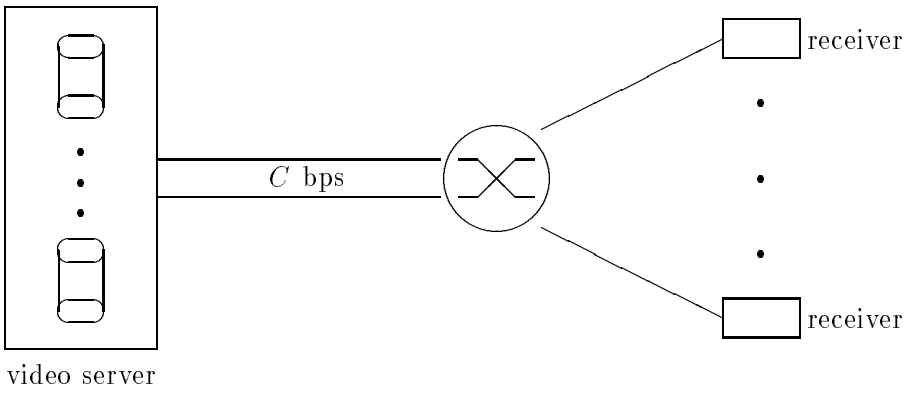


Figure 2.1: Prerecorded videos multiplexed over a link of capacity C bps. Throughout we assume that the switch and the connections to the receivers introduce no delay.

sequence $\{x_n(j), 1 \le n \le N\}$ for the jth video is a known sequence of integers. We shall find it convenient to extend the definition of this sequence for n ranging from $-\infty$ to $+\infty$ as follows:

$$(\ldots, x_{-2}(j), x_{-1}(j), x_0(j), x_1(j), x_2(j), \ldots),$$

where

$$x_{n+N}(j) = x_n(j) \qquad \text{for} \quad n = \ldots, -2, -1, 0, 1, 2, \ldots$$

Thus the infinite sequence is created by repeating the video trace over and over again.

To model the random start times and the VCR actions, we assign to the jth video, $j = 1, \ldots, J$, the random phase θ_j. We suppose that the θ_j's, $j = 1, \ldots, J$, are independent and that each θ_j is uniformly distributed over $\{0, \ldots, N-1\}$. (We choose a uniform distribution to fix ideas; however, the theory can be developed with an arbitrary distribution.) Our model supposes that the amount of traffic generated by the jth video at frame time n is

$$X_n(j, \theta_j) := x_{n+\theta_j}(j).$$

Therefore, for a given phase profile $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_J)$, the total amount of traffic generated by the J videos at frame time n is

$$X_n(\boldsymbol{\theta}) = \sum_{j=1}^{J} X_n(j, \theta_j) = \sum_{j=1}^{J} x_{n+\theta_j}(j).$$

This completes our formal definition of the traffic model. Henceforth we write $X_n(j)$ and X_n for $X_n(j, \theta_j)$ and $X_n(\boldsymbol{\theta})$, respectively.

Having defined the traffic model, we now highlight some of its implications. First note that for each fixed n, $X_n(1), \ldots, X_n(J)$ are independent random variables. Second note that the probability mass function for $X_n(j)$ can be calculated directly from the known trace $\{x_1(j), x_2(j), \ldots, x_N(j)\}$ as follows:

$$P(X_n(j) = l) = \pi_j(l), \tag{2.1}$$

where we define

$$\pi_j(l) := \frac{1}{N} \sum_{n=1}^{N} 1(x_n(j) = l).$$

Observe, in particular, that the distribution of $X_n(j)$ does not depend on n. We tacitly assume here that VCR actions don't change the frame size distribution, that is, we assume that viewers don't invoke VCR actions to see more high action scenes.

2.2.1 Quality of Service Measures

Over the period of time during which a video is in progress, the video will see a variety of different phase profiles $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_J)$ with respect to the other videos in progress. For each phase profile, there will be a fraction of frame periods during which the traffic rate exceeds the link rate and cell loss occurs. By taking the expectation over all possible phase profiles, we obtain the expected fraction of frame periods during which cell loss occurs. We are therefore motivated to define $P_{\text{loss}}^{\text{time}}$ by

$$P_{\text{loss}}^{\text{time}} := E[\lim_{M \to \infty} \frac{1}{M} \sum_{m=1}^{M} 1(X_m > \frac{C}{F})].$$

Note that for all $M \geq 1$,

$$P_{\text{loss}}^{\text{time}} = E[\frac{1}{M} \sum_{m=1}^{M} 1(X_m > \frac{C}{F})]$$

and that for all n

$$P_{\text{loss}}^{\text{time}} = P(X_n > \frac{C}{F}). \tag{2.2}$$

In the subsequent sections we will find it convenient to work with the expression (2.2) for $P_{\text{loss}}^{\text{time}}$.

Because X_n is the sum of J independent random variables, the probability in (2.2) can be calculated by convolution. However, convolution often leads to numerical problems, and its computational complexity may not be suitable for real–time admission control. For these reasons we shall explore approximations and bounds for $P(X_n > \frac{C}{F})$ in the next section.

We shall also study the QoS measure $P_{\text{loss}}^{\text{info}}$, where

$$P_{\text{loss}}^{\text{info}} := \frac{E[(X - \frac{C}{F})^+]}{E[X]}. \tag{2.3}$$

From the fact

$$\sum_{n=1}^{N} X_n = \sum_{n=1}^{N} \sum_{j=1}^{J} x_n(j)$$

it follows that

$$\begin{aligned} P_{\text{loss}}^{\text{info}} &= E\left[\frac{\sum_{n=1}^{N}(X_n - \frac{C}{F})^+}{\sum_{n=1}^{N} X_n}\right] \\ &= E\left[\lim_{M\to\infty} \frac{\sum_{m=1}^{M}(X_m - \frac{C}{F})^+}{\sum_{m=1}^{M} X_m}\right]. \end{aligned}$$

These two last expressions evoke an average — over all possible phase profiles — of the fraction of information (bits) lost.

To simplify notation, we henceforth write X for X_n and write $X(j)$ for $X_n(j)$. Also let $a := C/F$.

2.3 Bounding and Approximating QoS measures

In this section we develop central–limit and large–deviation approximation for $P_{\text{loss}}^{\text{time}} = P(X > a)$ and $P_{\text{loss}}^{\text{info}} = E[(X-a)^+]/E[X]$.

2.3.1 Central Limit Approximation

For the jth video, the average number of bits in a frame is

$$m(j) = \frac{1}{N} \sum_{n=1}^{N} x_n(j)$$

and the sample variance is

$$\sigma^2(j) = \frac{1}{N-1}\sum_{n=1}^{N}[x_n(j) - m(j)]^2.$$

By the central limit theorem [1, p. 310], X is approximately a normal random variable with mean and variance

$$m = \sum_{j=1}^{J} m(j) \qquad \sigma^2 = \sum_{j=1}^{J} \sigma^2(j).$$

Throughout the remainder of this subsection we assume that the approximation is exact, that is, we assume that

$$X \stackrel{d}{=} N(m, \sigma^2).$$

With the established traffic model, the expected fraction of time there is cell loss $P_{\text{loss}}^{\text{time}} = P(X > a)$ may now be easily computed from the tail of the normal distribution. Given a specific quality of service (QoS) requirement $P_{\text{loss}}^{\text{time}} \leq \epsilon$, where ϵ is a small number such as 10^{-7}, the QoS requirement $P_{\text{loss}}^{\text{time}} \leq \epsilon$ is met if and only if

$$\frac{1}{2}\text{erfc}(\frac{a-m}{\sqrt{2\sigma^2}}) \leq \epsilon \tag{2.4}$$

where $\text{erfc}(\cdot)$ denotes the well–known complementary error function defined as

$$\text{erfc}(a) = \frac{2}{\sqrt{\pi}}\int_a^{\infty} e^{-t^2}dt.$$

Using this admission rule (2.4), whenever a new video $J+1$ requests establishment, we update $m \leftarrow m + m(J+1)$ and $\sigma^2 \leftarrow \sigma^2 + \sigma^2(J+1)$. (It is natural to assume that $m(j)$ and $\sigma^2(j)$ have been calculated off line.) We then admit the new video if and only if (2.4) is met.

Now suppose that the QoS requirement $P_{\text{loss}}^{\text{info}} \leq \epsilon$ is imposed. Given the normal distribution of the amount of traffic that arrives during a specific frame period, $P_{\text{loss}}^{\text{info}} = E[(X-a)^+]/E[X]$ can be calculated by noting that

$$E[(X-a)^+] = \int_a^{\infty}(x-a)f_X(x)dx,$$

where

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x-m)^2}{2\sigma^2}},$$

and furthermore $E[X] = m$. Some straightforward calculus shows, that the QoS requirement $P_{\text{loss}}^{\text{info}} \leq \epsilon$ is met if and only if

$$\frac{1}{2}(1 - \frac{a}{m})\text{erfc}(\frac{a-m}{\sqrt{2\sigma^2}}) + \frac{1}{\sqrt{2\pi}}\frac{\sigma}{m}\exp[\frac{-(a-m)^2}{2\sigma^2}] \leq \epsilon.$$

2.3.2 Large Deviation Bound and Approximation

For small tail probabilities, the central limit approximation can under estimate the loss probability (see numerical results in Section 2.4). In this subsection we provide a upper bound for $P_{\text{loss}}^{\text{time}}$ and large deviation approximations for $P_{\text{loss}}^{\text{time}}$ and $P_{\text{loss}}^{\text{info}}$. To this end for any random variable Y let

$$\mu_Y(s) := \ln E[e^{sY}].$$

Note that $\mu_Y(s)$ is the logarithm of the moment generating function for Y.

For any random variable Y and constant c, the Chernoff bound gives an upper bound for $P(Y > c)$ (see Hui [27, 26], Kelly [30], Mitra *et al.* [14] , Roberts [62]). Applying the Chernoff bound to $P(X > a)$ we obtain

$$P_{\text{loss}}^{\text{time}} \leq e^{-s^\star a + \mu_X(s^\star)} \tag{2.5}$$

where $s^\star$ is the unique solution to

$$\mu'_X(s^\star) = a. \tag{2.6}$$

Note that

$$\mu_X(s) = \sum_{j=1}^{J} \mu_{X(j)}(s).$$

Given a specific QoS requirement $P_{\text{loss}}^{\text{time}} \leq \epsilon$, the Chernoff bound gives the admission control condition

$$e^{-s^\star a + \mu_X(s^\star)} \leq \epsilon. \tag{2.7}$$

Using (2.7), the admission control mechanism operates as follows. The logarithmic generating functions $\mu_{X(j)}(s)$ are first calculated off line for all videos. Now suppose a new video $J+1$ requests establishment. We update $\mu_X(s) \leftarrow \mu_X(s) + \mu_{X(J+1)}(s)$ and then find the

$s^\star$ that satisfies (2.6). Finally, we admit the video if and only if (2.7) is satisfied. We remark that Grossglauser *et al.* [24] have recently proposed a large-deviation approximation for $P_{\text{loss}}^{\text{time}}$. In their scheme, the prerecorded traffic is first smoothed into piecewise constant rates. For each change in the constant rate, they propose a renegotiation of network bandwidth. Their "probability of renegotiation failure" is similar to $P_{\text{loss}}^{\text{time}}$.

The bound given by (2.5) can be converted to an accurate approximation by applying the theory of large deviations (see Hui [27, p. 202], Hsu and Walrand [25], Roberts Roberts96):

$$P_{\text{loss}}^{\text{time}} = P(X \geq a) \approx \frac{1}{s^\star \sqrt{2\pi \mu_X''(s^\star)}} e^{-s^\star a + \mu_X(s^\star)}.$$

The large deviation (LD) approximation gives the following admission control condition:

$$\frac{1}{s^\star \sqrt{2\pi \mu_X''(s^\star)}} e^{-s^\star a + \mu_X(s^\star)} \leq \epsilon. \tag{2.8}$$

There is also a large deviation approximation for $P_{\text{loss}}^{\text{info}}$ (see Roberts [62]):

$$P_{\text{loss}}^{\text{info}} = \frac{E[(X-a)^+]}{E[X]} \approx \frac{1}{m s^{\star 2} \sqrt{2\pi \mu_X''(s^\star)}} e^{-s^\star a + \mu_X(s^\star)}.$$

Thus the large deviation approximation gives the following admission control criterion:

$$\frac{1}{m s^{\star 2} \sqrt{2\pi \mu_X''(s^\star)}} e^{-s^\star a + \mu_X(s^\star)} \leq \epsilon. \tag{2.9}$$

2.4 Numerical Results for the "Star Wars" Trace

In order to evaluate the admission control conditions introduced in the previous section, we conducted some numerical experiments with the MPEG–I *Star Wars* bandwidth trace, available via anonymous FTP from Bellcore [20]. The trace gives the number of bits in each video frame. In our experiments, all of the J videos use the *Star Wars* trace, but each video has its own random phase. The movie was compressed with the Group of Pictures (GOP) pattern IBBPBBPBBPBB (12 frames) at a frame rate $F = 24$ frames/second. The trace has a total number of $N = 174{,}136$ frames which corresponds to a run time of approximately 2 hours. Some salient statistical properties of the *Star Wars* trace are given in Table 2.1. We consider a single bufferless ATM node with transmission rate $C = 155$

Frames			GOPs			Bitrate	
Mean bits	St. Dev. bits	Peak/ Mean	Mean bits	St. Dev. bits	Peak/ Mean	Mean Mbps	Peak Mbps
15,598	18,165	11.9	187,178	72,869	5.05	0.37	4.45

Table 2.1: Statistics of *Star Wars* trace.

Mbps. (At the end of this section we also consider $C = 45$ Mbps.) We furthermore assume that all 48 bytes of the ATM cell payload are used to transport the video frames.

In order to validate the approximation described in Section 2.3, we ran simulation experiments using the *Star Wars* trace. In the simulation algorithms described below we focus on the $P_{\text{loss}}^{\text{time}}$ criterion; the algorithms for $P_{\text{loss}}^{\text{info}}$ are similar. Based on our assumption of uniformly distributed phases (see Section 2.2) there are two different simulation approaches possible.

One approach works as follows: For a fixed number of connections, J, draw the phases θ_j, $j = 1, \ldots, J$, from a discrete uniform distribution over $[0, N-1]$ (denote this distribution by DU$[0,N-1]$) and check whether loss occurs for this phase profile, that is, check whether $\sum_{j=1}^{J} x_{\theta_j}(j) > a$. Repeat this procedure many times in order to obtain an estimate for the fraction of phase profiles that have loss, that is, for $P_{\text{loss}}^{\text{time}}$.

An alternative approach proceeds as follows: Draw the start phases $\theta_j \sim$ DU$[0, N-1]$, $j = 1, \ldots, J$, and then simulate the transmission of the entire *Star Wars* trace and count the number of frames with loss (see Figure 2.2 for the details of this algorithm). In this algorithm we wrap the trace around when the index extends beyond the end of the trace, that is, for $n + \theta_j > N$ we replace $n + \theta_j$ by $n + \theta_j - N$.

We used the second approach in our simulation experiments since it gives tighter confidence intervals than the first approach for the same amount of CPU time. The first approach is computationally more expensive because it requires a new set of random phases for each simulated frame time, while the second approach allows us to use the same set of random phases for N frame times. Note that there is a statistical difference between the two approaches: In the first approach every simulated frame period constitutes an independent trial; in the second approach, on the other hand, the simulated transmission of the entire trace is an independent trial since the frame sizes are correlated within the

1. Fix J;
2. $o = 0$; $p = 0$;
3. **For** $l = 1$ **to** L **do**
4. Draw $\theta_j \sim \mathrm{DU}[0, N-1]$, $j = 1, \ldots, J$;
5. q = 0;
6. **For** $n = 1$ **to** N **do**
7. $X_n = \sum_{j=1}^{J} x_{n+\theta_j}(j)$;
8 $q = q + 1(X_n > C/F)$;
9. $o = o + q$;
10. $p = p + q^2$;
11. $\hat{P}_{\mathrm{loss}}^{\mathrm{time}} = \frac{o}{NL}$; (sample mean)
12. $\hat{S}^2 = \frac{1}{N^2(L-1)}(p - o^2/L)$; (sample variance)

Figure 2.2: Simulation Algorithm for $P_{\mathrm{loss}}^{\mathrm{time}}$.

	no IS		IS	
J	$\hat{P}_{\mathrm{loss}}^{\mathrm{time}}$	90% CI	$\hat{P}_{\mathrm{loss}}^{\mathrm{time}}$	90% CI
268	$4.59\ 10^{-8}$	$[0,\ 4.54\ 10^{-7}]$	$1.80\ 10^{-7}$	$[1.37\ 10^{-7},\ 2.23\ 10^{-7}]$
276	$8.43\ 10^{-7}$	$[0,\ 4.61\ 10^{-6}]$	$1.35\ 10^{-6}$	$[9.83\ 10^{-7},\ 1.72\ 10^{-6}]$
284	$8.50\ 10^{-6}$	$[0,\ 5.12\ 10^{-5}]$	$5.88\ 10^{-6}$	$[3.76\ 10^{-6},\ 7.99\ 10^{-6}]$
292	$8.28\ 10^{-5}$	$[0,\ 4.46\ 10^{-4}]$	$7.27\ 10^{-5}$	$[1.40\ 10^{-5},\ 1.31\ 10^{-4}]$

Table 2.2: Simulation Results for J unsmoothed *Star Wars* connections without and with Importance Sampling.

video trace. This fact has to be accounted for when computing the sample variance (see Figure 2.2, line 12).

In Table 2.2 (column no IS) we give the sample mean and 90% confidence intervals for $P_{\mathrm{loss}}^{\mathrm{time}}$ for the unsmoothed *Star Wars* trace. In this experiment we ran $L = 500$ replications for a total of $500 \times 174,136 \approx 87 \times 10^6$ simulated frame periods. We note that the half lengths of the confidence intervals are larger than the sample means; this implies that the lower end of the confidence interval for $P_{\mathrm{loss}}^{\mathrm{time}}$ is zero, as subtracting the half length from the sample mean would result in negative probabilities. This phenomenon is a consequence of the simulation of rare loss events; tightening the confidence intervals further without employing variance reduction techniques, such as importance sampling (discussed below), would require immense computational resources. Each of the 4 simulation results in the

J	no IS		IS	
	$\hat{P}_{\text{loss}}^{\text{time}}$	90% CI	$\hat{P}_{\text{loss}}^{\text{time}}$	90% CI
338	$1.10\ 10^{-6}$	$[3.90\ 10^{-7}, 1.82\ 10^{-6}]$	$5.37\ 10^{-7}$	$[4.89\ 10^{-7}, 5.86\ 10^{-7}]$
342	$7.17\ 10^{-6}$	$[5.29\ 10^{-6}, 9.05\ 10^{-6}]$	$6.37\ 10^{-6}$	$[5.54\ 10^{-6}, 7.19\ 10^{-6}]$
346	$6.15\ 10^{-5}$	$[5.77\ 10^{-5}, 6.52\ 10^{-5}]$	$5.04\ 10^{-6}$	$[4.27\ 10^{-5}, 5.81\ 10^{-5}]$

Table 2.3: Simulation Results for J GOP–smoothed *Star Wars* connections without and with Importance Sampling.

no IS column in Table 2.2 took about 2 days on a SPARCstation 2.

Table 2.3 shows the results obtained after smoothing the *Star Wars* trace over each Group of Pictures (GOPs). In this experiment we ran $L = 2,000$ replications for a total of $2,000 \times 14,511 \approx 29 \times 10^6$ simulated GOPs. We observe that the simulation of 29×10^6 GOPs gives confidence intervals that are significantly tighter than those obtained for the unsmoothed trace, even though the simulation for the unsmoothed trace required three times the computational effort. This seems to indicate that the GOP smoothed trace is more amenable to simulation experiments.

Importance Sampling

In order to obtain better estimates for $P_{\text{loss}}^{\text{time}}$, particularly for the unsmoothed trace, and reduce the simulation time, we apply Importance Sampling (IS) techniques to our problem. The basic idea of IS is to draw the random phases $\theta_j,\ j = 1, \ldots, J$, from a distribution $g_j(m_j) = P(\theta_j = m_j)$ that leads to more frequent losses than the discrete unform distribution used in the experiments described above. Let $f_j(m_j) = 1/N$ denote this discrete uniform distribution. Since we use the *Star Wars* trace for all J video streams, we use the same distribution for all videos and write henceforth $g(m_j)$ and $f(m_j)$ for $g_j(m_j)$ and $f_j(m_j)$. The fraction of frame periods for which there is cell loss is given by

$$\begin{aligned} P(X > a) &= E[1(X > a)] \\ &= \sum_{m_1=1}^{N} \cdots \sum_{m_J=1}^{N} h(m_1, \ldots, m_J) \prod_{j=1}^{J} g(m_j) \end{aligned}$$

where

$$h(m_1, \ldots, m_J) = 1(x_{m_1} + \cdots + x_{m_J} > a)(\prod_{j=1}^{J} \frac{f(m_j)}{g(m_j)}).$$

The Monte Carlo estimate for $P_{\text{loss}}^{\text{time}}$ is given by

$$\hat{P}_{\text{loss}}^{\text{time}} = \frac{1}{L}\sum_{l=1}^{L} h(m_1^{(l)}, \ldots, m_J^{(l)}),$$

where the samples $m_j^{(l)}, j = 1, \ldots, J$, are all drawn from $g(\cdot)$ for $l = 1, \ldots, L$. With IS for each l we use a new set of random phases (the θ_j's). Recall that without IS we use the same set of phases for any entire simulated trace.

The challenge of the IS approach lies in finding the distribution $g(\cdot)$ that gives small variances for $P_{\text{loss}}^{\text{time}}$. For the GOP smoothed trace we obtained the IS results in Table 2.3 for L = 100,000 with

$$g(m) = \frac{y_m}{\sum_{n=1}^{N} x_n}.$$

Here y_m is the size of the mth GOP, that is $y_m = \sum_{n=(m-1)G+1}^{mG} x_n$ and G denotes the number of frames per GOP. Each simulation result took about one hour, roughly twenty times faster than the corresponding simulation without IS. Note this choice of $g(\cdot)$ favors larger frames (as compared with $f(\cdot)$). This will cause $h(m_1, \ldots, m_J)$ to be strictly positive more frequently; but when $h(m_1, \ldots, m_J)$ is strictly positive, it will not be excessively large since its denominator is likely large. Thus the sampling function $g(\cdot)$ causes the output stream of $h(\cdot)$ to be less variable, thereby giving a tighter confidence interval.

We found that the discrete empirical distribution

$$g(m) = \frac{\sqrt{x_m}}{\sum_{n=1}^{N} \sqrt{x_n}}$$

works well for the unsmoothed *Star Wars* trace. The square root in $g(\cdot)$ dampens the tendency to draw only large frames. Given the large ratio of peak–to–mean frame size (see Figure 2.1) the distribution $g(m) = x_m / \sum_{n=1}^{N} x_n$ would misrepresent the trace. A small number of excessively large frames would dominate the simulation while the large number of small frames would be ignored. The chosen distribution however still encourages large values for the x_n's and leads to more frequent losses; the $g(\cdot)$ in the denominator of the expression for $h(\cdot)$ compensates for this effect. The IS results displayed in Table 2.2 were obtained for $L = 2 \times 10^6$; note that IS has significantly reduced the width of the confidence intervals. Each of the four simulation results took approximately 2 hours on a SPARCstation 2.

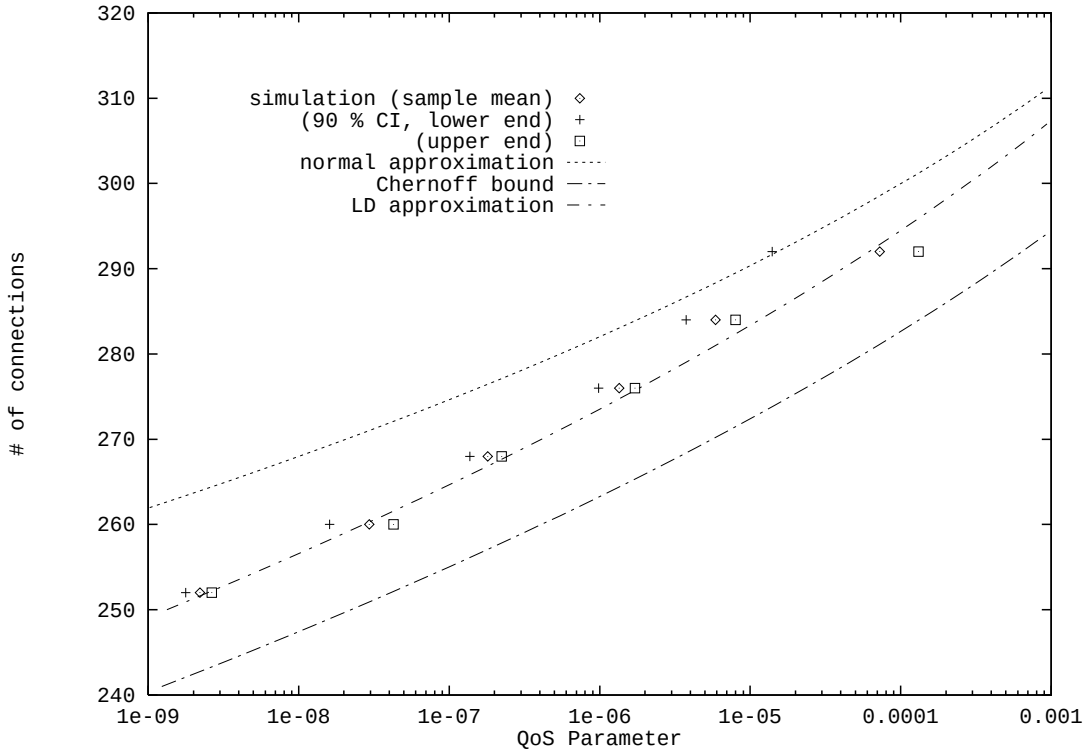


Figure 2.3: A comparison of approximations and simulation (unsmoothed, $P_{\text{loss}}^{\text{time}}$ criterion).

Approximations and Bounds

In Figure 2.3 we compare the results from IS, the normal approximation, the LD approximation and the Chernoff bound. The figure shows the number of *Star Wars* connections allowed for by the condition $P_{\text{loss}}^{\text{time}} \leq \epsilon$ as the QoS parameter ϵ varies. The calculations are based on the unsmoothed *Star Wars* trace, that is, each frame is transmitted in its assigned interval of length $1/F$. The diamonds represent the sample mean, $\hat{P}_{\text{loss}}^{\text{time}}$, resulting from IS. We ran $L = 2 \times 10^6$ replications for each $J = 252, 260, \ldots, 292$. For a fixed number of connections, J, the crosses and boxes indicate the lower and upper ends of the 90% confidence interval for $P_{\text{loss}}^{\text{time}}$. The curve labeled "normal approximation" is calculated using the admission control condition (2.4). Given that the simulation represents the actual loss probabilities, the normal approximation is an optimistic admission policy. The curves labeled "Chernoff bound" and "LD approximation" are computed using conditions (2.7) and (2.8), respectively. The Chernoff bound appears to be a conservative admission control policy as it lies about 10 connections below the boxes representing the upper end

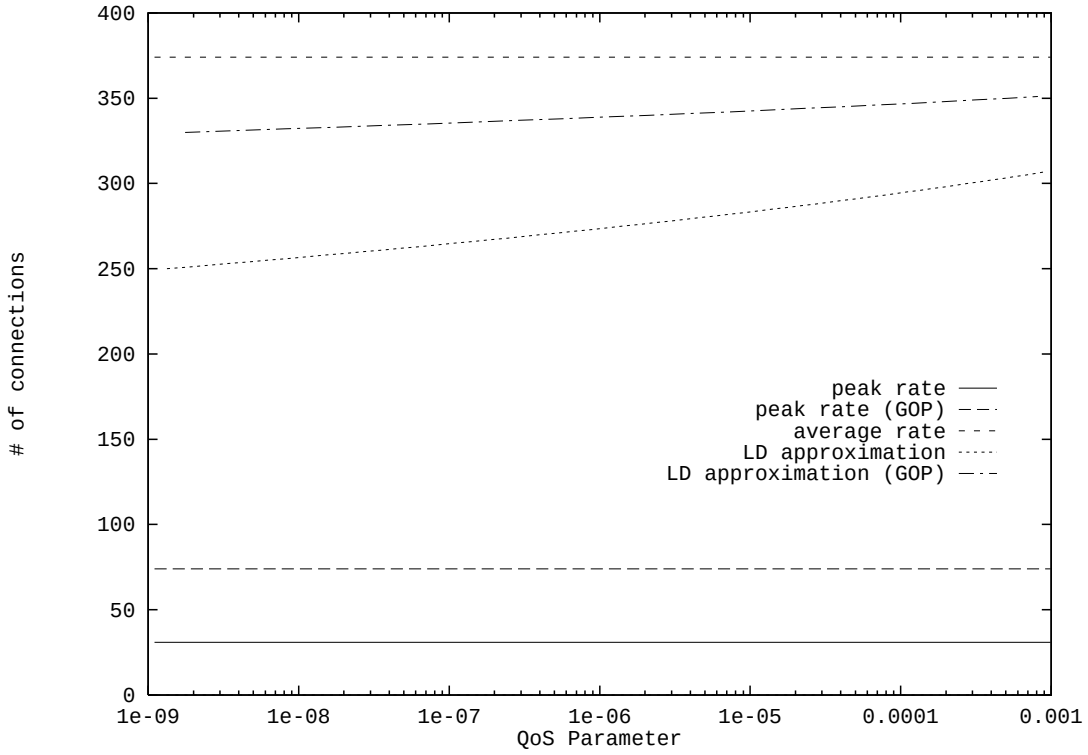


Figure 2.4: The effect of GOP smoothing on the number of admissible connections as a function of $P_{\text{loss}}^{\text{time}}$.

of the 90% confidence interval for $P_{\text{loss}}^{\text{time}}$. The LD approximation appears to be the appropriate tool for admission control for prerecorded VBR video as it almost coincides with the sample means from the simulation. We will henceforth focus on the LD approximation in our analysis.

In Figure 2.4 we investigate the effect of smoothing the trace over a GOP. The curves were obtained by using the $P_{\text{loss}}^{\text{time}}$ criterion (2.8). The "unsmoothed curves" already appeared in Figure 2.3 and are replicated here for comparison purposes. The GOP curves were calculated by first smoothing the *Star Wars* trace over each GOP (12 frames). The figure reveals that smoothing over the GOP increases the admission region substantially, resulting in higher network utilization for a given QoS requirement. For a QoS parameter $\epsilon = 10^{-6}$, for example, smoothing over the GOP increases the number of admissible connections by approximately 24%. This corresponds to an increase in the average link

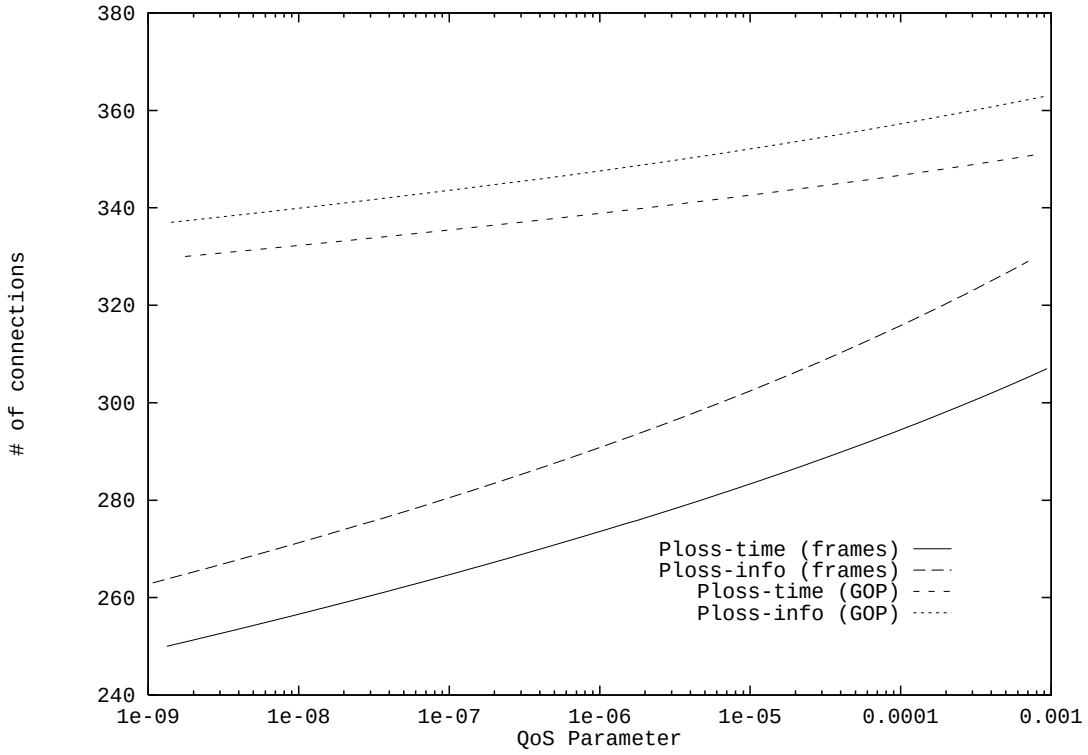


Figure 2.5: Comparison of $P_{\text{loss}}^{\text{time}}$ and $P_{\text{loss}}^{\text{info}}$ criteria.

utilization (defined as the number of admissible connections $\times m/a$) from 73% for the unsmoothed trace to 90% for the GOP smoothed trace. We also observe from Figure 2.4 that the number of allowed connections is nearly constant with respect to the QoS requirement for GOP smoothing. Peak rate admission allows for 31 unsmoothed *Star Wars* connections, giving an average link utilization of 8.3%. After smoothing the *Star Wars* trace over the GOP, a peak rate admission policy would allow for 74 connections, which results in an average link utilization of 20%. Average–rate admission permits 374 connections ($= C/Fm$).

In Figure 2.5 we compare the criteria $P_{\text{loss}}^{\text{time}}$ and $P_{\text{loss}}^{\text{info}}$. We notice that the $P_{\text{loss}}^{\text{info}}$ criterion allows for slightly more connections. For $\epsilon = 10^{-7}$, for example, the number of admissible unsmoothed *Star Wars* connections is increased by approximately 6% while the number of GOP smoothed connections is 2.4% higher. This observation can be intuitively explained by noting that only the fraction $(X - a)/a$ of cells is lost during periods of overload. While the criterion $P_{\text{loss}}^{\text{time}}$ is based on the long run fraction of time there is cell loss, the criterion

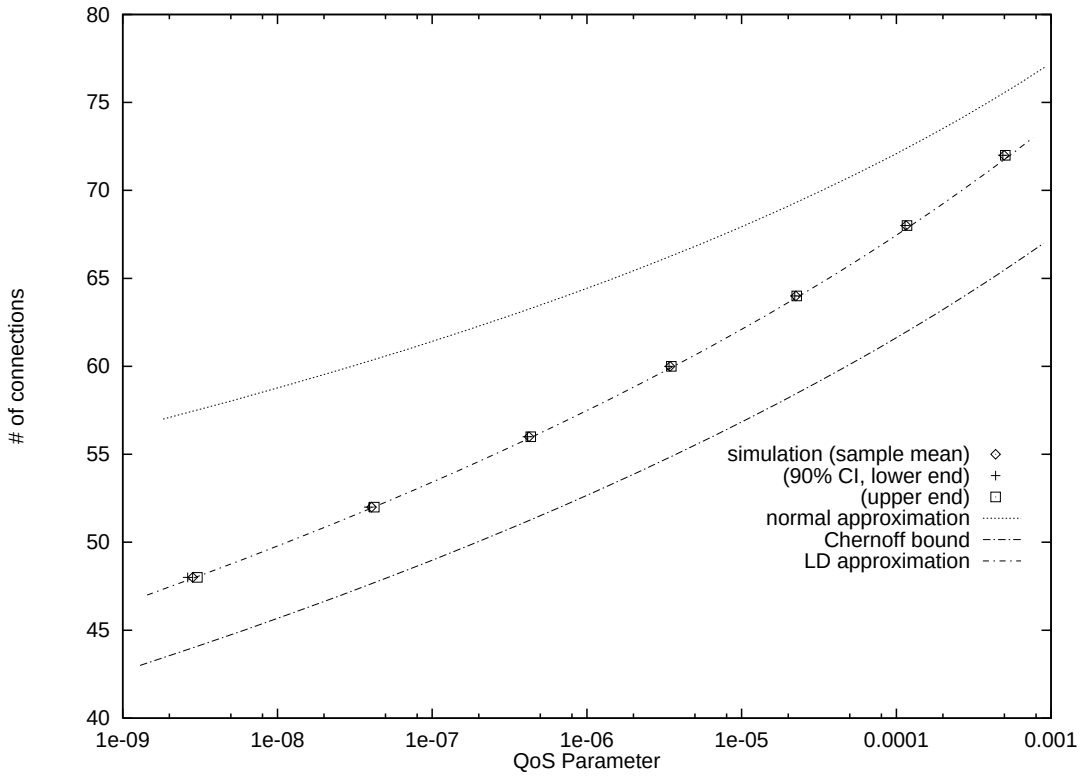


Figure 2.6: A comparison of approximations and simulation for $C = 45$ Mbps as a function of $P_{\text{loss}}^{\text{time}}$.

$P_{\text{loss}}^{\text{info}}$ accounts for the fact that not all cells are lost during overload.

2.4.1 Numerical Experiments for 45 Mbps Link

In order to investigate the effect of network bandwidth on our results, in particular on the accuracy of the LD approximation, we chose to replicate the experiments described in the previous section for an ATM link with bandwidth $C = 45$ Mbps. As in the previous experiments, we use the MPEG 1 encoded *Star Wars* trace and assume that all 48 bytes of the ATM cell payload are used to transport the frames.

In Figure 2.6 we depict the results from IS simulation, normal approximation, Chernoff bound and LD approximation for the unsmoothed *Star Wars* trace for $C = 45$ Mbps. As in the experiment for $C = 155$ Mbps (see Figure 2.3) we set the parameter L of the IS simulation to 2×10^6. Notice that the IS heuristic introduced in the previous section gives even tight confidence intervals for $C = 45$ Mbps. The simulation also verifies that the LD

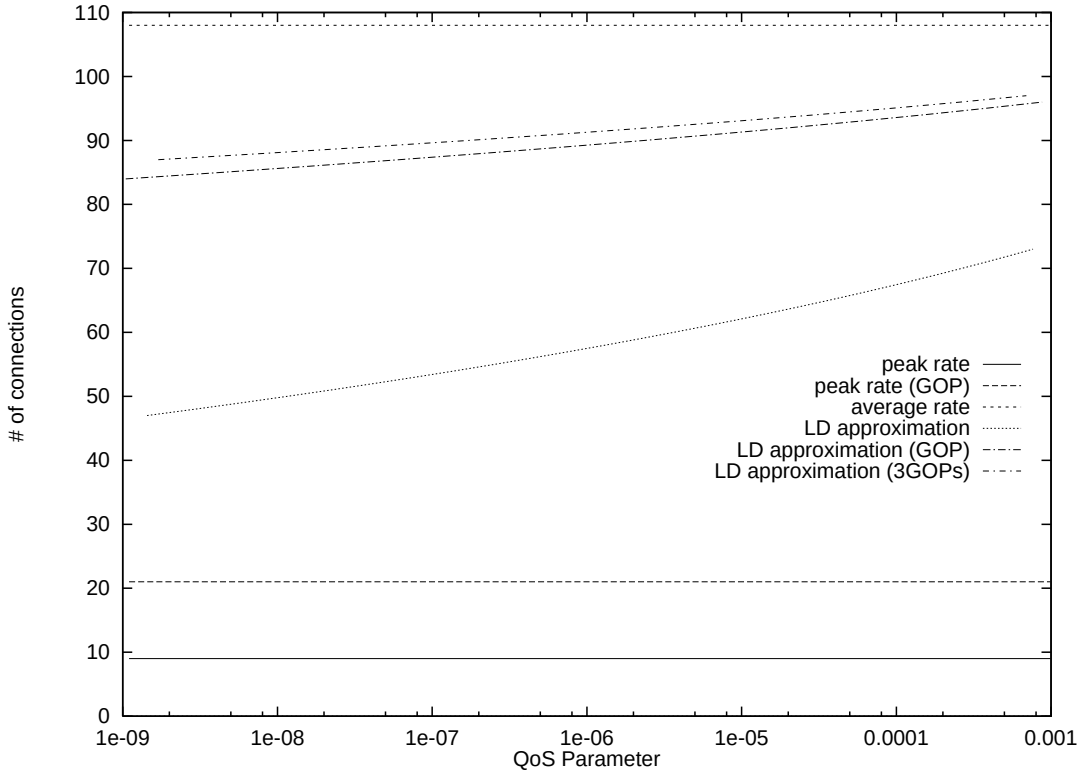


Figure 2.7: The effect of smoothing over one GOP or three GOPs on the number of admissible connections as a function of $P_{\text{loss}}^{\text{time}}$.

approximation remains accurate when fewer connections are multiplexed.

Figure 2.7 shows the effect of smoothing the trace over each GOP. We also plot the number of admissible connections when the *Star Wars* trace is smoothed over three GOPs. Note that the additional smoothing hardly increases the admission region. We also see, as in the case of $C = 155$ Mbps (see Figure 2.4), that smoothing over one GOP increases the number of admissible connections substantially. For the QoS parameter $\epsilon = 10^{-6}$, for instance, the number of admissible connections is increased by about 54 %.

Note that the link utilizations are not as high as for the 155 Mbps link. The QoS requirement $P_{\text{loss}}^{\text{time}} \leq 10^{-6}$ permits 57 unsmoothed *Star Wars* connections which corresponds to a link utilization of approximately 53 % (we had 73 % for $C = 155$ Mbps). For the same QoS requirement 88 GOP smoothed connections are allowed, resulting in a link utilization of about 81 % (90 % for $C = 155$ Mbps). As in the case of $C = 155$ Mbps we see that the number of admissible GOP smoothed connections is almost insensitive to the

QoS parameter.

2.5 Efficient Calculation of Admission Control Decisions

In this section we address the real time implementation of the proposed admission control tests. The criterion derived for the normal approximation (2.4) can be readily tested in real time (assuming that $m(j)$'s and $\sigma(j)$'s have been calculated off–line for all videos).

The conditions resulting from Chernoff bound and LD approximation, however, require more computational effort. Recall from Section 2.3.2 that these conditions are based on $\mu_X(s)$, the logarithmic generating function of the total amount of traffic arriving from all active video streams in one frame period. With (2.1), this function can be explicitly written as

$$\mu_X(s) = \sum_{j=1}^{J} \ln\Big(\sum_{l=x_{\min}}^{x_{\max}} \pi_j(l) e^{sl} \Big), \tag{2.10}$$

where $x_{\min}$ and $x_{\max}$ denote the smallest and largest frame size, respectively. We assume for simplicity that $x_{\min}$ and $x_{\max}$ are identical for all videos. We furthermore assume that the histogram of the frame sizes, $\pi_j(l)$, has been computed off–line for all videos.

Now suppose that a new video $J+1$ requests connection establishment. The admission control will proceed as follows: First, find the $s^\star$ that satisfies $\mu'_X(s^\star) + \mu'_{X(J+1)}(s^\star) = a$. This can best be done with Newtons method [53] starting from the $s^\star$ of the last admission test. The new connection is accepted if and only if (2.9) is satisfied. This procedure can require an excessive amount of CPU time for real-time implementation. For this reason we now investigate computational procedures for accelerating the run time. (We note that such procedures are not needed for the scheme of Grossglauser *et al.* [24] because their scheme employs a relatively small number of rates.)

Considerable speedup of the described computation can be achieved by reducing the resolution of the histogram $\pi_j(l)$. So far, our calculations are based on $\pi_j(l)$ computed according to (2.1) for $l = x_{\min}, \ldots, x_{\max}$, where the step size for l is one bit. In order to reduce the resolution, we may compute the histogram as

$$\pi_j(l) = \frac{1}{N} \sum_{n=1}^{N} 1(l - \text{binsize} < x_n(j) \le l) \quad \text{for } l = x_{\min}, x_{\min} + \text{binsize}, \ldots, x_{\max}.$$

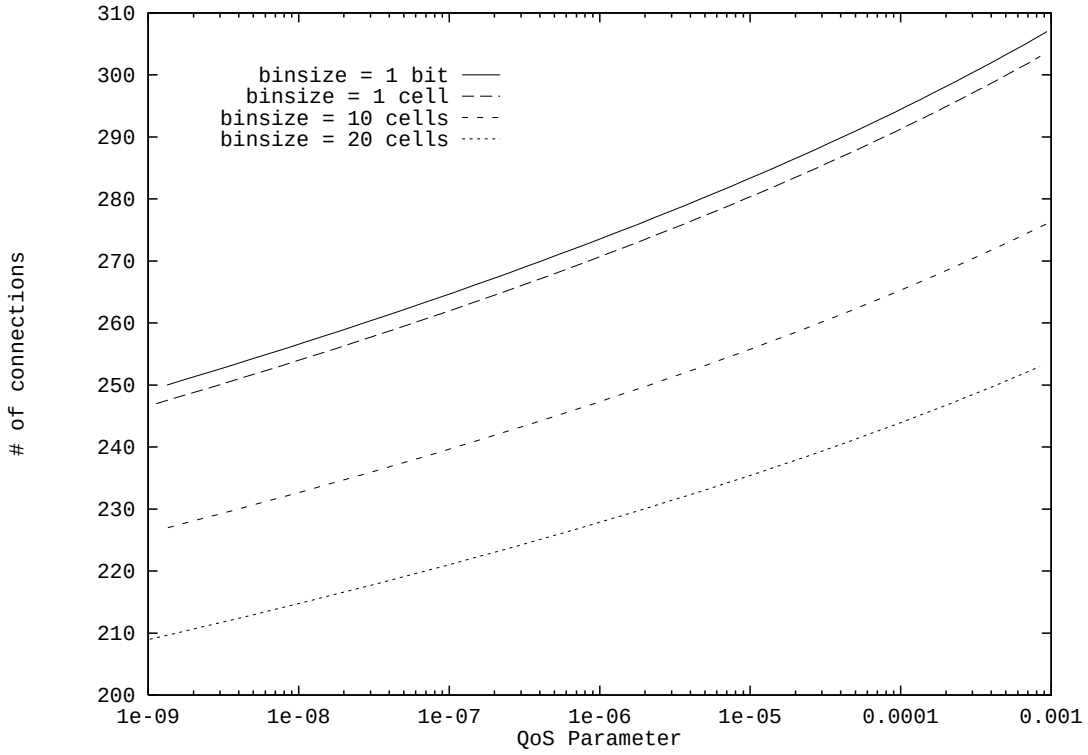


Figure 2.8: Number of connections with varying resolution of the histogram as a function of $P_{\text{loss}}^{\text{time}}$.

binsize	1 bit	1 cell	10 cells	20 cells
CPU time	80 min	13 sec	4.5 sec	0.75 sec

Table 2.4: CPU times for admission tests based on histogram with varying resolution.

Collecting the frames into bins in this fashion ensures that the probability of cell loss is an increasing function of the binsize and leads hence to conservative admission decisions. Figure 2.8 shows the number of admissible *Star Wars* connections computed with varying resolutions (binsizes) for the *Star Wars* histogram. Table 2.4 gives the typical CPU times required for an admission test. All computations are performed on a Sun SPARCstation 2. The graph shows that increasing the binsize to 1 ATM cell (384 bit) reduces the number of admissible connections by approximately 1.1% while reducing the CPU time by a factor of about 370.

We mention that a quick approximate test can be performed in the following fashion:

Store $s^\star$ and $\mu_X(s^\star)$ of the previous admission test, compute only $\mu_{X(J+1)}(s^\star)$ on–line without changing $s^\star$ (takes approximately 1.4 sec for a histogram with resolution of 1 bit, 3.7 msec for a resolution of 1 cell), and check if (2.9) holds. If the new connection is accepted, update $s^\star$ and $\mu_X(s^\star)$ while the new video is being transmitted. Note that updates must also be done when a connection terminates. This procedure is motivated by the fact that $s^\star$ typically varies only slightly when adding a new connection. Furthermore, using a sub–optimal $s^\star$ leads to conservative acceptance decisions, since the expression in the exponent of the Chernoff bound (2.5), which dominates the LD approximation, is strictly convex [14, p. 1119]. We refer to this procedure as the *on–line procedure*.

The admission control algorithms discussed so far compute the logarithmic generating function $\mu_X(s)$ directly from the histogram of the frame sizes. As an alternative, we apply an idea of Hui [27, p. 206] to our problem at hand: We expand the logarithmic generating function in a Taylor series and base the acceptance decision on pre–computed coefficients representing the videos. To this end, we first note that the moment generating function of the jth video may easily be developed into a Taylor series:

$$M_{X(j)}(s) = \sum_{l=x_{\min}}^{x_{\max}} \pi_j(l) e^{sl} = \sum_{l=x_{\min}}^{x_{\max}} [\pi_j(l) \sum_{u=0}^{\infty} \frac{(sl)^u}{u!}] = \sum_{i=0}^{\infty} a_i(j) s^i$$

where

$$a_i(j) \equiv \frac{1}{i!} \sum_{l=x_{\min}}^{x_{\max}} \pi_j(l) l^i.$$

We note that the frame sizes, l, (as well as the link capacity a in the conditions of Section 2.3.2) should be scaled to improve the convergence of the admission region obtained from the series approximation to the admission region computed directly from the histogram. We achieved rapid convergence (see Figure 2.10) with a scale factor q of 60 cells for the unsmoothed *Star Wars* trace. This factor is somewhat higher than m, the average frame size of the *Star Wars* video, and ensures that $E[X/q] < 1$. Without this scaling the l^i in the expression for $a_i(j)$ can easily lead to huge values; we conjecture that the common scale factor used for admission control should be larger than the largest $m(j)$ of all videos available on the video server.

In a second step we may compute the series expansion of the logarithmic moment

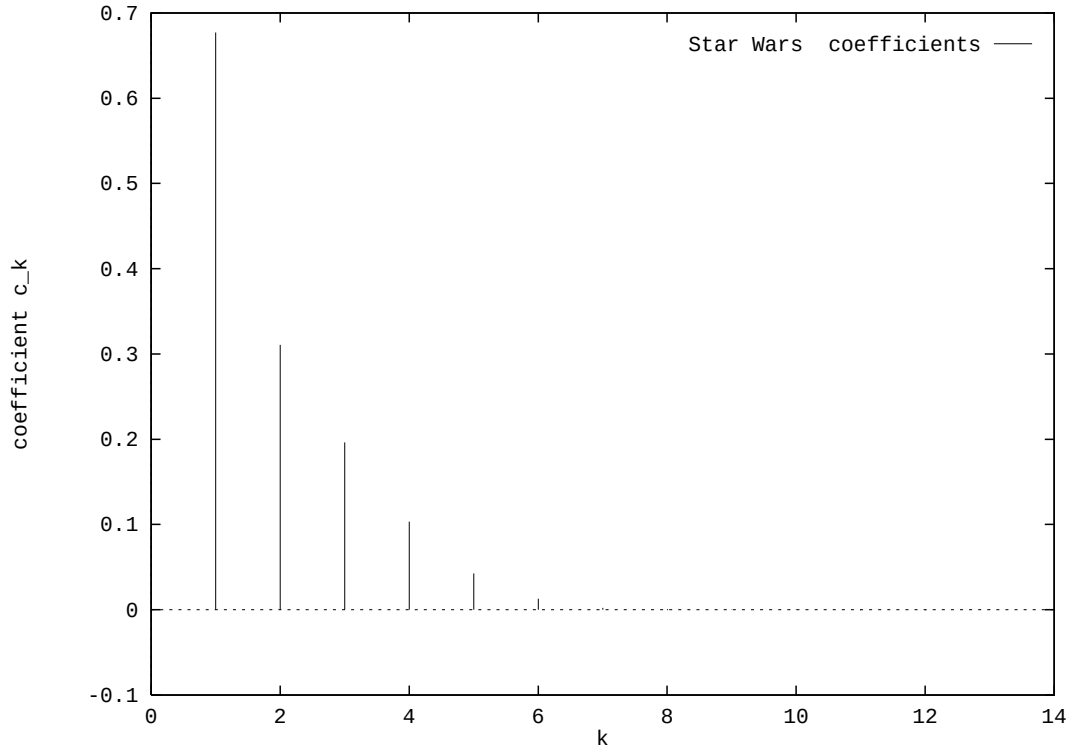


Figure 2.9: c_k coefficients for *Star Wars* video.

generating function:

$$\mu_{X(j)}(s) = \ln M_{X(j)}(s) = \ln \sum_{i=0}^{\infty} a_i(j) s^i = \sum_{k=1}^{\infty} c_k(j) s^k. \tag{2.11}$$

where the coefficients $c_k(j)$ are given by (see Hui [27, p. 206])

$$c_k(j) = a_k(j) - \frac{1}{k} \sum_{i=1}^{k-1} i a_{k-i}(j) c_i(j).$$

The coefficients for the *Star Wars* video, computed with a scale factor of 60 cells, are depicted in Figure 2.9. Our experiments seem to indicate that a relative small number, K, of coefficients in (2.11) approximates the logarithmic moment generating function with sufficient accuracy for the purpose of admission control. Figure 2.10 shows the number of admissible *Star Wars* connections computed from the series expansion of μ_X with different K. The figure shows that the admission region obtained from the series approach converges rather quickly to the admission region calculated with the direct approach. For $K \geq 4$ both curves are almost indistinguishable. This seems to indicate that videos can be characterized

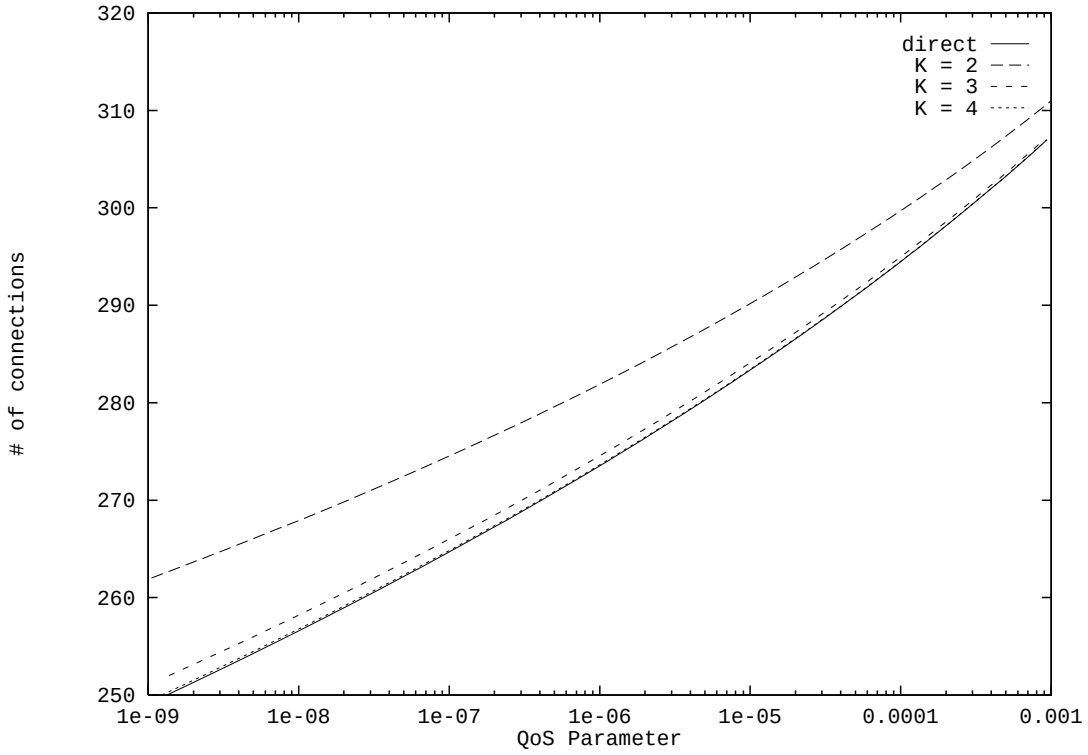


Figure 2.10: The effect of the number of coefficients in Taylor series approximation as a function of $P_{\text{loss}}^{\text{time}}$.

by a set of coefficients $\{c_k(j), 1 \leq k \leq K\}$. These coefficients can be computed off–line and stored with the actual video. Given these video descriptors, the admission control tests of Section 2.3.2 can now be conducted very efficiently by noting that

$$\mu_X(s) = \sum_{k=1}^{K} c_k s^k$$

where

$$c_k = \sum_{j=1}^{J} c_k(j), \qquad 1 \leq k \leq K.$$

This method avoids the expensive computation of the sum of exponentials in the direct approach (2.10) and is furthermore independent of the number of videos already in progress.

If a new video $J+1$ requests connection establishment, we first update $c_k \leftarrow c_k + c_k(J+1),\ 1 \leq k \leq K$, and find the $s^\star$ that satisfies $\mu'_X(s^\star) = a$, starting Newtons method with the $s^\star$ found in the last admission test. Finally, we admit the new connection if and

only if (2.9) is satisfied. This admission test takes approximately 21 msec for $K = 6$, and is hence viable for real time admission control.

Owing to the results in this section, we believe there are 2 options for real time admission control employing the LD approximation: (1) the on–line procedure discussed earlier, (2) the test based on the precomputed $c_k(j)$ coefficients as we just described. We acknowledge that it would be desirable to support these conclusions with additional experiments using other video traces as well as heterogeneous mixes of traces. Unfortunately, there is currently a lack of publicly available traces that give a correct representation of the bandwidth requirements of MPEG compressed videos.

2.6 A Refined Admission Control Procedure

We begin this section by supposing that VCR control (pause, rewind, fast forward) is no longer permitted. We do suppose, however, that the various videos begin playback at different times. For $n = 1, \ldots, N$, define $X_n(j)$ as in Section 2.2. But now define $X_n(j) = 0$ for all $n > N$.

Suppose that during frame time l there are J videos in progress with the jth video having trace $\{x_1(j), x_2(j), \ldots, x_N(j)\}$. Suppose during frame time l, frame θ_j of video j is transmitted. Now consider admitting a new video $J+1$ (with trace $\{x_1(J+1), x_2(J+1), \ldots, x_N(J+1)\}$) which is to begin transmission in frame time $l+1$. With this new video, the amount of offered traffic at frame time $n+l$ is

$$x_{n+\theta_1}(1) + \cdots + x_{n+\theta_J}(J) + x_n(J+1).$$

An admission rule which guarantees no loss for the duration of the new video is

$$\sum_{j=1}^{J+1} x_{\theta_j+n}(j) \le a, \qquad n = 1, \ldots, N, \tag{2.12}$$

where $\theta_{J+1} := 0$. An admission rule which permits loss for a fraction of frame periods is

$$\frac{\sum_{n=1}^{N} 1[\sum_{j=1}^{J+1} x_{\theta_j+n}(j) > a]}{N} \le \epsilon. \tag{2.13}$$

Similarly one can easily define admission rules which permit a fraction of bit loss (analogous to $P_{\text{loss}}^{\text{info}}$).

```
1.  Fix J, M;
2.  o = 0; p = 0;
3.  For l = 1 to L do
4.     q = 0; I = P;
5.     Draw θ_i ~ DU[0, N/G − 1], i = 1, ..., I;
6.     While (q < 1) and (I ≤ J) do
7.        I = I + 1;
8.        Draw θ_I ~ DU[0, N/G − 1];
9.        For m = 1 to M do
10.          X_m = Σ_{i=1}^{I} x_{m+θ_i}(i);
11.          q = q + 1(X_m > Ga);
12.    o = o + I − 1;
13.    p = p + (I − 1)^2;
14. Î = o/L; (sample mean)
15. Ŝ^2 = 1/(L−1) (p − o^2/L); (sample variance)
```

Figure 2.11: Simulation Algorithm for combined admission test.

Now let us once again suppose that VCR features are permitted. The theory developed in Section 2.2 takes a global view on the phase profiles: It essentially assumes that over the course of a video there will be many phase profiles. An admission control procedure that takes a more myopic view to call admission is one akin to rule (2.12) but over fewer frame periods, e.g.,

$$\sum_{j=1}^{J+1} x_{\theta_j+n}(j) \le a, \qquad n = 1, \ldots, M, \tag{2.14}$$

where the θ_j's are the current phases at the call admission time and $M \ll N$.

We can define an appealing admission rule by combining one of the global tests in Section 2.3 with the test (2.14) (or with a test similar to (2.14), such as a test permitting some cell loss). For a given QoS parameter ϵ, such a combined test admits fewer connections than the isolated global test. However, this combined test guards against the possibility of excessive cell loss due to unusual phase profiles at call admission.

We conducted simulation experiments with the GOP smoothed *Star Wars* trace to evaluate the combination of global and myopic admission test. Figure 2.6 presents the employed simulation algorithm for the combination of any of the global tests of Section 2.3 and the myopic test (2.14). (An admission rule involving a test that is akin to (2.14)

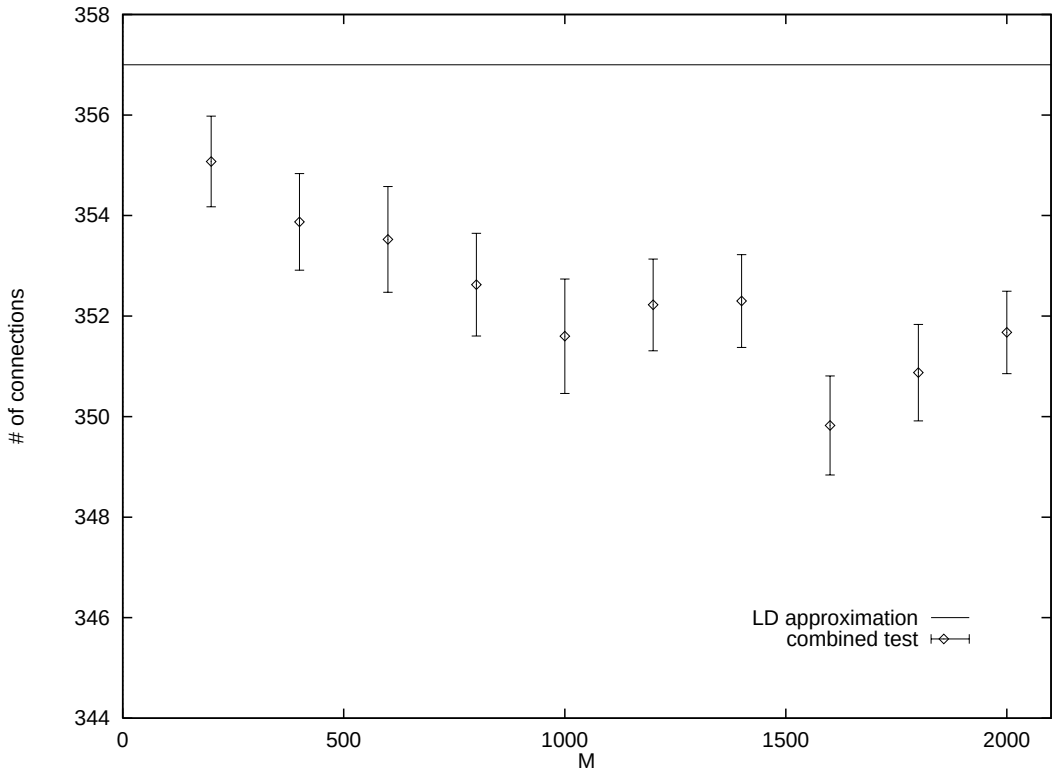


Figure 2.12: Combined admission test (LD approximation for $P_{\text{loss}}^{\text{info}}$ and (14)) for varying M.

but allows for some loss can be simulated in a similar fashion.) We first fix the number of connections, J, allowed by the global admission test for a specific QoS parameter ϵ. We then simulate the transmission of M consecutive GOPs of I videos, where each video has an independent starting phase that is drawn from a discrete uniform distribution over $[0, N/G-1]$, where G denotes the number of frames per Group of Pictures (GOP). Starting from P, the number of connections allowed by a peak rate admission scheme, we increase the number of video streams, I, until we experience loss during the transmission of the M GOPs or hit the limit given by the global admission test. We thus find the maximum number of connections allowed by the combined admission test. We run L replications to find a confidence interval for the expected maximum number of connections.

Figure 2.12 shows the number of *Star Wars* connections admitted by the combination of the LD approximation for $P_{\text{loss}}^{\text{info}}$ (2.9) and the myopic test (2.14) as the parameter M varies. The latter criterion ensures that there is no loss during the transmission of M GOPs following the admission of a new video connection, provided none of the videos experiences

VCR actions. For this simulation run we fixed $\epsilon = 10^{-4}$ for the condition (2.9), this gives $J = 357$ GOP smoothed *Star Wars* connections for the global test (see Figure 2.4). We ran $L = 40$ replications for $M = 200, 400, \ldots, 2000$. Note that M GOPs correspond to $M/2$ real–time seconds. *Star Wars* has a total number of 14,511 GOPs. We see from the figure that the confidence intervals for the expected number of connections admitted by the combined test are centered only slightly below the $P_{\text{loss}}^{\text{time}}$ limit of 357 connections. This shows that, even for large values of M, the myopic test (2.14) has little impact on admission control, implying that the global admission test will typically not lead to large losses.

2.7 Conclusion

In this chapter we have developed rules for admitting prerecorded sources to an unbuffered link. We have considered two QoS requirements: the first requires that the fraction of frame periods during which loss occurs be less than a given ϵ; the second requires that the fraction of cells lost be less than ϵ. We have presented several approximations for loss for prerecorded traffic, and for the *Star Wars* trace we have found that the large–deviations approximation is quite accurate. Our numerical results have also shown, for multiple copies of *Star Wars*, that it is possible to get a high–degree of statistical multiplexing, particularly when each trace is smoothed over its GOPs. We also observed that the number of allowed connections is often insensitive to the cell–loss requirement ϵ. We then explored efficient on–line calculation of admission control decisions and indicated two procedures which appear promising. Finally, we refined the global admission test in order to guard against the possibility of excessive cell loss due to unusual phase profiles at call admission.

Chapter 3

Join–the–Shortest–Queue Prefetching

3.1 Overview

In this chapter we present a high–performance prefetching protocol for the delivery of video on demand (VoD) from a server across a packet–switched network to a large number of clients. The protocol assumes that the videos resident on the video server are variable–bit–rate (VBR) encoded. Not only does this protocol give constant perceptual quality and almost 100% link utilization, but it also allows for immediate commencement of the video upon user request and near instantaneous response to interactive actions (pause/resume and temporal jumps).

To achieve this high performance our protocol has two requirements. First we require all of the clients to have a small amount of memory dedicated to the VoD application. Second, as shown in Figure 3.1, we require that there be at most one bottleneck shared link between the video server and the clients. If the clients are connected to an ADSL residential access network, then this second requirement can be achieved by attaching the video server directly to the ADSL central office; in this case the shared link is the link from the server to the ADSL central office. If the clients are connected to cable, then the second requirement can be achieved by attaching the video server directly to the cable headend.

The user's client could be a television with a set–top box capable of performing buffering

and decoding, or it could be a household PC. The video server could be a cache containing the week's most popular videos. A central repository could multicast the videos to this and other caches over the Internet using the traditional best–effort service [10].

Our protocols explicitly assume that the videos are VBR encoded with high peak–to–mean ratios. The motivation for our approach is that, for the same perceived video quality, Constant Bit Rate (CBR) encoding produces an output rate significantly higher than the average rate of the corresponding VBR encoding for action movies [9]. CBR traffic allows for nearly 100% link utilization; the number of connections that can be carried over a link of given capacity is roughly the link capacity divided by the CBR rate (assuming homogeneous connections). The number of VBR connections that can be transmitted simultaneously is the achievable link utilization multiplied by the link capacity divided by the average rate of the VBR video stream. Therefore schemes for transmitting VBR encoded video that achieve high average link utilizations while keeping losses at a negligible level, can allow for significantly more video connections than does CBR video.

Our protocol achieves the constant perceptual quality, responsiveness to user interactivity, and high link utilizations by exploiting two special properties of the prerecorded video: (1) for each video, the traffic in each video frame is known before the video session begins; (2) while the video is being played, some of the video can be prefetched into the client memory. It is this second property — the ability to prefetch a portion of any video — that is particularly central to our high–performance protocol.

Our protocol is based on the observation that, due to the VBR nature of the multiplexed traffic, there will be frequent periods of time during which the shared link's bandwidth is under utilized. During these periods the server can prefetch video frames from any of the ongoing videos and send the prefetched frames to the buffers in the appropriate clients. With this prefetching, many of the clients will typically have some prefetched reserve in their buffers. Our protocol also specifies the policy for how the server selects the prefetched frames. This policy is the join–the–shortest–queue (JSQ) policy, which can be roughly described as follows: within each frame time the server repeatedly selects frames from the connections that have the smallest number of prefetched frames in their client buffers. The JSQ policy creates a *buffer pooling effect* so that the system behaves as if the individual client buffers are aggregated into one large buffer which is shared by all the clients. Our

empirical work with public–domain traces indicates that prefetching combined with the JSQ policy gives dramatic reductions in packet loss. In particular, if each client dedicates a small amount of buffer capacity to the VoD application, this scheme can multiplex a large number of connections over the shared link and have negligible playback starvation. In this chapter we also examine several refinements and variations of our JSQ prefetching policy, and we develop schemes for selectively discarding frames for MPEG encoded video when playback starvation is unavoidable.

With JSQ prefetching, through buffer pooling, the connections collaborate in order to minimize the overall packet loss. This collaboration among the connections contributes significantly to the protocol's outstanding performance. We also present numerical results which show that *Optimal Smoothing*, a non–collaborative prefetching policy, can have packet loss that is several orders of magnitude higher than that of JSQ prefetching for a wide range of buffer sizes.

This chapter is organized as follows. In Section 3.2 we give a description of our VoD architecture and introduce prefetching. In Section 3.2 we present our JSQ prefetch policy; Section 3.2 includes numerical results for public domain traces and a discussion of efficient implementation of the JSQ policy. In Section 3.4 we show how our VoD protocol allows for user interactivity with minimal delays; numerical results for interactive actions are presented. In Section 3.5 we compare the performance of JSQ prefetching with that of Optimal Smoothing [67, 79]. In Section 3.6 we explore a variation of our JSQ policy — the packet–based JSQ prefetch policy. In Section 3.7 we examine MPEG compression and propose several policies for selective discard when loss is unavoidable. In Section 3.8 we discuss how JSQ prefetching can be employed in residential broadband access networks using cable or ADSL. We conclude in Section 3.9.

3.2 Architecture Description

Figure 3.1 illustrates our basic model for VoD. The video server contains a large number of videos in mass storage. For notational simplicity, assume that each video consists of N frames and has a frame rate of F frames/sec. The videos are VBR encoded using MPEG 1, MPEG 2 or some other video compression algorithm. Let J denote the number of video

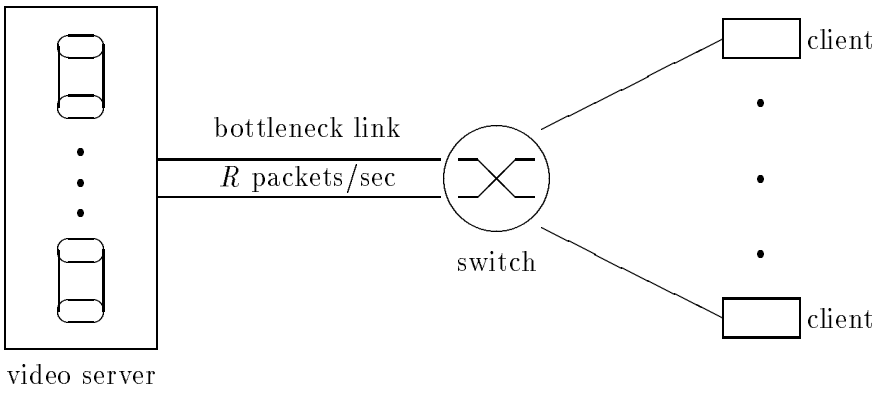


Figure 3.1: Prerecorded videos multiplexed over a link of capacity R packets/sec.

connections in progress. Although some of the connections might be transmitting the same video, the phases (i.e., the start times) are typically different. The server packetizes the frames of the ongoing connections and then statistically multiplexes and transmits the packets into its link; for simplicity, we assume throughout that the packets are of fixed length. Let $x_n(j)$ denote the number of packets in the nth frame of the jth connection. Because the videos are prerecorded, $(x_1(j), x_2(j), \ldots, x_N(j))$ is fully known at connection establishment.

When a client requests a specific video, the server makes an admission control decision by deciding whether or not to grant the request. If it grants the request, a connection is established and the server immediately begins to transmit the connection's packets into the network. The connection's packets are transmitted in a fixed, predetermined order. When packets arrive at the client, they are placed in the client's prefetch buffer. The video is displayed on the user's monitor as soon as a few frames have arrived at the client.

Under normal circumstances, every $1/F$ seconds the client removes a frame from the prefetch buffer, decompresses it, and displays it. If at one of these epochs there are no complete frames in its prefetch buffer, the client loses the current frame; the client will try to conceal the loss by, for instance, redisplaying the previous frame. At the subsequent epoch the client will attempt to display the next frame of the video.

We denote R (in packets/sec) for the maximum transmission rate of the server. We

denote $R(j)$ for the maximum reception rate of the jth client. Although our protocol allows for pause and temporal jumps, we will initially exclude these interactive features in order to simplify the discussion; thus N/F seconds elapse from when the user begins to watch the video until when the video ends. We shall also initially assume that all prefetch buffers are infinite and that $R(j) \geq R$ for $j = 1, \ldots, J$.

To make these ideas a little more precise, we divide time into slots of length $1/F$. Let $p_l(j)$ be the number of frames in the prefetch buffer for connection j at the beginning of slot l. Let $\Delta_l(j)$ be the number of frames of connection j that arrive to the prefetch buffer during the lth slot. At the end of each slot, one frame is removed from each prefetch buffer that has one or more frames. Thus

$$p_{l+1}(j) = [p_l(j) + \Delta_l(j) - 1]^+, \tag{3.1}$$

where $[x]^+ = \max(x, 0)$. Denote $\theta_l(j)$ for the frame number of connection j that is supposed to be removed at the end of slot l; thus $\theta_{l+1}(j) = \theta_l(j) + 1$.

During each slot of length $1/F$ seconds the server must decide which frames to transmit from the J ongoing videos. The *prefetch policy* is the rule that determines which frames are transmitted in each slot. The maximum number of packets that can be transmitted in a slot is R/F (which for simplicity we assume to be an integer).

For each ongoing connection j the server keeps track of the prefetch buffer contents $p_l(j)$; this can be done through the recursion (3.1) without communicating with the client. Initially, we require the server to skip the transmission of the entire frame $\theta_l(j)$ whenever $p_l(j) + \Delta_l(j) = 0$. Thus, the server does not transmit a frame that will not meet its deadline at the client.

Before defining our prefetch policy, it is useful to introduce some more notation. Let $b_l(j)$ be the number of packets in the prefetch buffer for connection j at the beginning of slot l. Let $\Delta'_l(j)$ denote the number of packets of connection j that arrive to the prefetch buffer during the the lth slot. These definitions imply

$$b_{l+1}(j) = [b_l(j) + \Delta'_l(j) - x_{\theta_l(j)}(j)]^+. \tag{3.2}$$

3.3 The JSQ Prefetch Policy

Our Join–the–Shortest–Queue (JSQ) prefetch policy attempts to balance the number of frames across all of the prefetch buffers. In describing this policy, we drop the subscript l from all notations. At the beginning of each slot the server determines the j^* with the smallest $p(j)$, transmits one frame from connection j^* and increments $p(j^*)$. Within this same slot the server repeats this procedure over and over again, at each iteration finding a new j^* that minimizes $p(j)$, transmitting a frame from connection j^* and incrementing $p(j^*)$.

Due to the finite transmission rate of the server, at some point the server must stop transmitting frames within the slot. To this end, let z be a variable that keeps track of the total number of packets sent within the slot; z is reinitialized to zero at the beginning of every slot. The stopping rule works as follows. Before transmitting a frame from connection j^* we check to see if

$$z + x_{\sigma(j^*)}(j^*) \leq R/F, \tag{3.3}$$

where $\sigma(j^*)$ is the frame of connection j^* that is being considered for transmission. If this condition holds, then we transmit the frame and update z; otherwise, we do not transmit the frame, set $p(j) = [p(j) - 1]^+$ for $j = 1, \ldots, J$ and recommence the procedure for the subsequent slot. This is our *basic stopping rule*; later we shall discuss a slightly more complicated stopping rule.

With prefetching, it is possible that all of a connection's frames have been transmitted but not all of its frames have been displayed. When a connection reaches this state, we no longer consider it in the above JSQ prefetching policy. From the server's perspective, it is as if the connection has been terminated.

3.3.1 Refinements of the JSQ policy

We now discuss a few important refinements of the JSQ policy. First, we introduce a refined stopping rule. Recall that during each slot the server transmits a sequence of frames until condition (3.3) is violated; once (3.3) is violated, the server does not transmit any more frames in the slot. An alternative stopping rule is to try to transmit more frames in the slot by removing from consideration the connection that violates (3.3) and finding a new j^*

that minimizes $p(j)$. If the condition (3.3) holds with the frame from the new connection j^*, we transmit the frame, update $p(j^*)$, and continue the procedure of transmitting frames from the connections that minimize the $p(j)$'s. Whenever a frame violates condition (3.3), we skip the corresponding connection and find a new j^*. When we have skipped over all of the connections, we set $p(j) = [p(j) - 1]^+$ for $j = 1, \ldots, J$ and move on to the next slot. This is our *refined stopping rule*. To reduce the online computational effort we can also, of course, consider rules which fall between the basic and refined stopping rules. For example we could use a rule which stops when condition (3.3) has been violated K times where $1 < K < J$.

The next refinement of the JSQ policy limits the number of packets an ongoing connection may have in its client's prefetch buffer. This important refinement is useful when the client for connection j, $j = 1, \ldots, J$, has finite buffer capacity $B(j)$. This refinement works as follows. Suppose that the server is considering transmitting frame $\sigma(j^*)$ from connection j^*. Let $b(j^*)$ be the current number of packets in the prefetch buffer for connection j^*. It transmits this frame in the current slot only if condition (3.3) and the condition

$$b(j^*) + x_{\sigma(j^*)}(j^*) \leq B(j) \tag{3.4}$$

are satisfied. Condition (3.4) ensures that the server does not overflow the prefetch buffer for connection j^*. With this additional condition, we extend the definitions of the stopping rules in the obvious way.

The final refinement we consider in this subsection is applicable when $R(j) < R$ for at least one connection j. This situation occurs when the client's access link has limited bandwidth (as with ADSL) or when the the client has a limited packet processing capability. In this case, within each slot the server would keep track of $z(j)$, the total number of packets from connection j sent in the slot. Suppose the server is considering transmitting the frame $\sigma(j^*)$ from connections j^*. It transmits the frame in the current slot if and only if conditions (3.3), (3.4) and

$$z(j^*) + x_{\sigma(j^*)}(j^*) \leq R(j^*)/F \tag{3.5}$$

are satisfied. Condition (3.5) ensures that the server does not violate the reception constraint for connection j^*. Once again, we can extend the definitions of the stopping rules in the obvious manner.

We also mention that if the switch has buffer capacity for each connection, then it may be possible to ignore (3.5) even when $R(j) < R$. The idea is to store the traffic exceeding the capacity of the jth access link in the output buffer of connection j, and to feed the access link at rate $R(j)$. An upper bound on the switch buffer capacity necessary for connection j is $[R - R(j)]B(j)/R$. This expression is based on the worst case scenario that the prefetch policy dedicates the entire link capacity R temporarily to connection j in order to fill its prefetch buffer completely. This scenario occurs if connection j starts with empty prefetch buffer while all other connections have full prefetch buffers.

To simplify the discussion, for the remainder of this chapter we shall assume that either (1) $R(j) \geq R$ for all $j = 1, \ldots, J$, or (2) the switch has sufficient buffer capacity so that condition (3.5) can be ignored.

3.3.2 System Dynamics and Pooling

We now crudely describe the dynamics of the prefetch buffer contents. Let us make the assumption that whenever all N frames of a connection are displayed, the same user immediately requests a new connection; thus, with this assumption there are always J videos in progress. Let us make the realistic assumption that

$$\sum_{j=1}^{J} x_{avg}(j) < R/F, \tag{3.6}$$

where $x_{avg}(j)$ is the average number of packets in a frame in the jth connection. The condition (3.6) says that the long–run average aggregate consumption rate of the ongoing videos is less than the maximum server supply rate.

The JSQ prefetch policy will make most of the $p(j)$'s nearly equal to each other. Moreover, because of (3.6) there will be a general tendency for each of the $p(j)$'s to increase. In other words, there is typically a "pack" of $p(j)$'s with near equal values, with each $p(j)$ in the pack drifting towards $B(j)$. A $p(j)$ can separate itself from the pack if it hits $B(j)$ while the pack continues to gain higher occupancies. It will also break from the pack if the corresponding connection reaches the state of having all its frames transmitted. In this case, the $p(j)$ will descend by one in each slot until it hits zero; when it hits zero, $p(j)$ will quickly return to the pack.

Trace	Mean bits	Peak/ Mean	Avg. Frame Size			% of info		
			I	P	B	I	P	B
lambs	7,312	18.4	38,025	7,436	3,424	43	25	32
bond	24,308	10.1	83,293	41,427	10,513	29	42	29
terminator	10,904	7.3	37,387	14,120	6,387	29	32	39
mr.bean	17,647	13.0	75,146	18,275	10,215	35	26	39

Table 3.1: Statistics of MPEG–1 traces.

One important feature of the JSQ prefetching policy is that it creates a *pooling effect*, namely, the individual buffers act as a large collected buffer of capacity $B(1)+\cdots+B(J)$. To see this, suppose that a large number of connections start simultaneously (therefore having few frames in their prefetch buffers) and the remaining connections have a large number of frames in their prefetch buffers. Also assume that the aggregate consumption rate temporarily exceeds R packets/sec. Then if the consumption rate across the connections that have just started is less than R, the JSQ policy will prevent frame loss as long as the high aggregate consumption rate does not persist for too long. This is because the prefetch policy will feed the connections that have just started until their prefetch buffers catch up with rest of the pack. Thus, the likelihood of cell loss in the near future typically depends on the $p(j)$'s only through the pooled buffer content $p(1) + p(2) + \cdots + p(J)$.

3.3.3 Experimental Results

In this subsection we present the results of a simulation study for the JSQ prefetch policy described in the previous section. Throughout we use the refined stopping rule discussed in Section 3.3.1. Our simulation study makes use of MPEG 1 encodings of the four movies in Table 3.1. The associated traces, obtained from the public domain [64], give the number of bits in each frame. (We are aware that these are low resolution traces and some critical frames are dropped; however, the traces are extremely bursty. We have obtained similar results, not reported here, for *Star Wars* and *Oz*.) Each of the movies was compressed with the GOP pattern IBBPBBPBBPBB at a frame rate of $F = 24$ frames/sec. Each of the traces has 40,000 frames, corresponding to about 28 minutes. The mean number of bits per frame and the peak–to–mean ratio are given in Table 3.1. Table 3.1 also gives the

average size of I, P and B frames and the percentage of encoding information carried in I, P and B frames. It can be argued that the average rate in bits/sec is lower than what we would expect for digital compressed video (e.g., MPEG–2 video); for this reason, we have chosen a relatively small server transmission rate of 45 Mbps. We expect VoD systems in the future to have videos with larger average rates and a proportionally larger server transmission rate. In this scaling, the number of videos that can be multiplexed will be approximately constant. We furthermore assume that each packet consists of 512 bytes of payload and 40 bytes of overhead; therefore, $R = 81,521$ packets/sec.

We define the link utilization as the average number of packets per second, summed over all connections in progress, divided by R. In our experiments we use various mixes of the the four movies. Each of the mixes has a different link utilization. Our 90% link utilization consists of 52 lambs connections, 16 bond connections, 35 terminator connections, and 22 mr.bean connections. Our 95% link utilization consists of 55 lambs connections, 17 bond connections, 37 terminator connections, and 23 mr.bean connections. With these numbers, each of the four movies accounts for roughly one fourth of the link load.

In each realization of our simulation, we generate a random starting frame $\theta(j)$ for each of the J ongoing connections. The value $\theta(j)$ is the frame that is removed from the jth prefetch buffer at the end of slot 1. The $\theta(j)$'s are independent and uniformly distributed over $[1, N]$. All connections start with empty prefetch buffers at the beginning of slot 1. When the Nth frame of a video is removed from a prefetch buffer, we assume that the corresponding user immediately requests to see the entire movie again. Thus, there are always J connections in progress.

In Figure 3.2 we set the buffer capacity of each client to 1 Mbyte and the link utilization to 95%. The figure shows the prefetch buffer contents in bytes for one of the lamb's connections over 120,000 simulated frame periods; the initial starting random phase for this connection is $\theta = 2,689$. At $l = 36,079$ frame periods all of the movie's frames have been transmitted to the prefetch buffer; this is why the buffer content drops to zero at $l = 37,311$, when the movie ends and starts over for the user. Note that when the movie restarts at $l = 37,311$ the JSQ prefetch policy fills the prefetch buffer for the movie in just a few frame periods. Also note that the buffer contents typically hug the 1 Mbyte buffer limit. Note that the buffer occupancy is nearly periodic, with period equal to the length

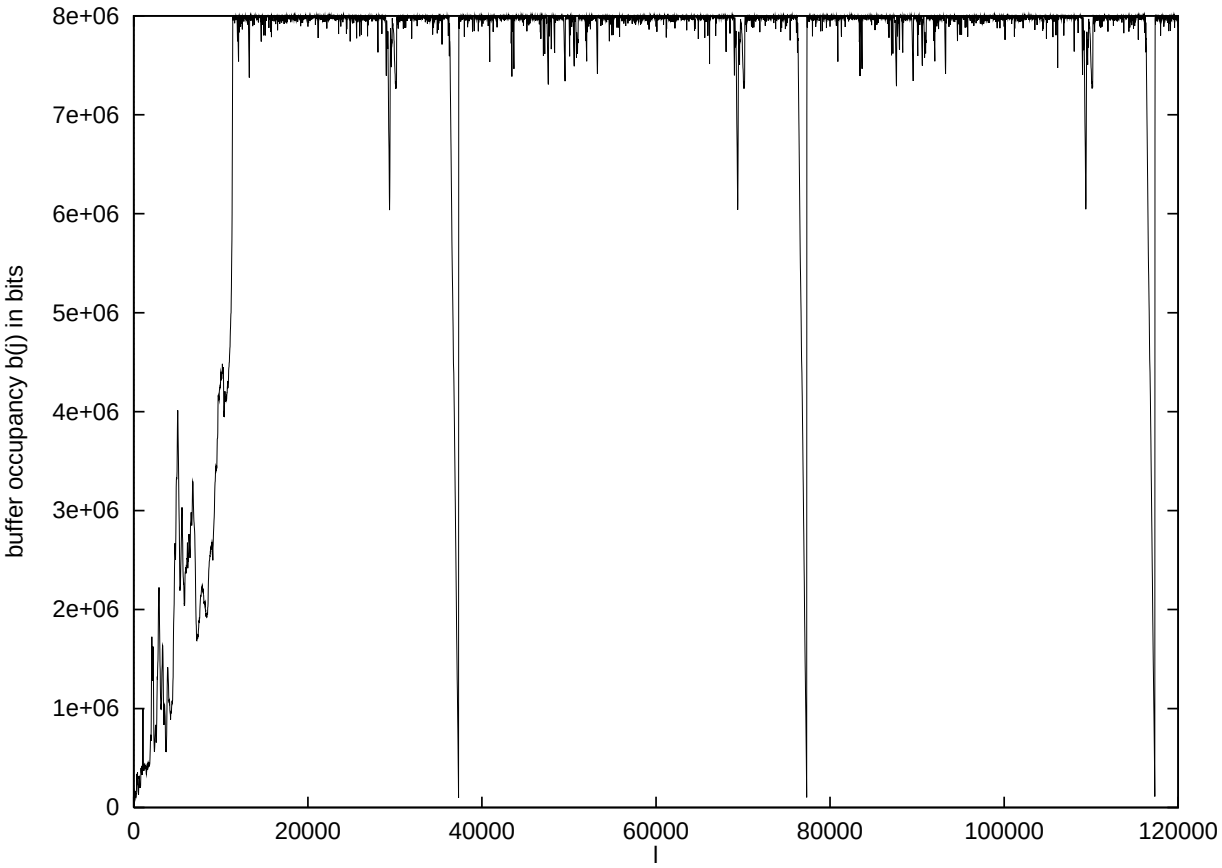


Figure 3.2: Prefetch buffer contents in a 1 Mbyte buffer in bits for a lamb's connection.

of the movies (40,000 frames).

In Figure 3.3 we plot the 90 % confidence intervals of $P_{\text{loss}}^{\text{time}}$ for the prefetch buffer sizes ranging from 2 KBytes to 256 KBytes. $P_{\text{loss}}^{\text{time}}$ is the long–run fraction of frame periods for which loss occurs for at least one of the J ongoing videos (see 2.2). Each of the confidence intervals is based on 1000 replications. On this figure $P_{\text{loss}}^{\text{time}}$ is plotted for both 95% and 90% utilizations.

Figure 3.3 shows the dramatic improvement in performance that comes from prefetching and the JSQ policy. Without any prefetching we have $P_{\text{loss}}^{\text{time}} = 0.29$ for 95 % utilization. By increasing the capacity only to 256 KBytes, the loss probability is reduced to $P_{\text{loss}}^{\text{time}} \approx 4.7 \times 10^{-6}$. By further increasing the buffer to 512 KBytes, no loss was observed for any of 8000 replications, corresponding to 3.2×10^8 frame periods! Similarly, for 90 % utilization we did not observe any loss for 1000 replications, corresponding to 4×10^7 frame periods,

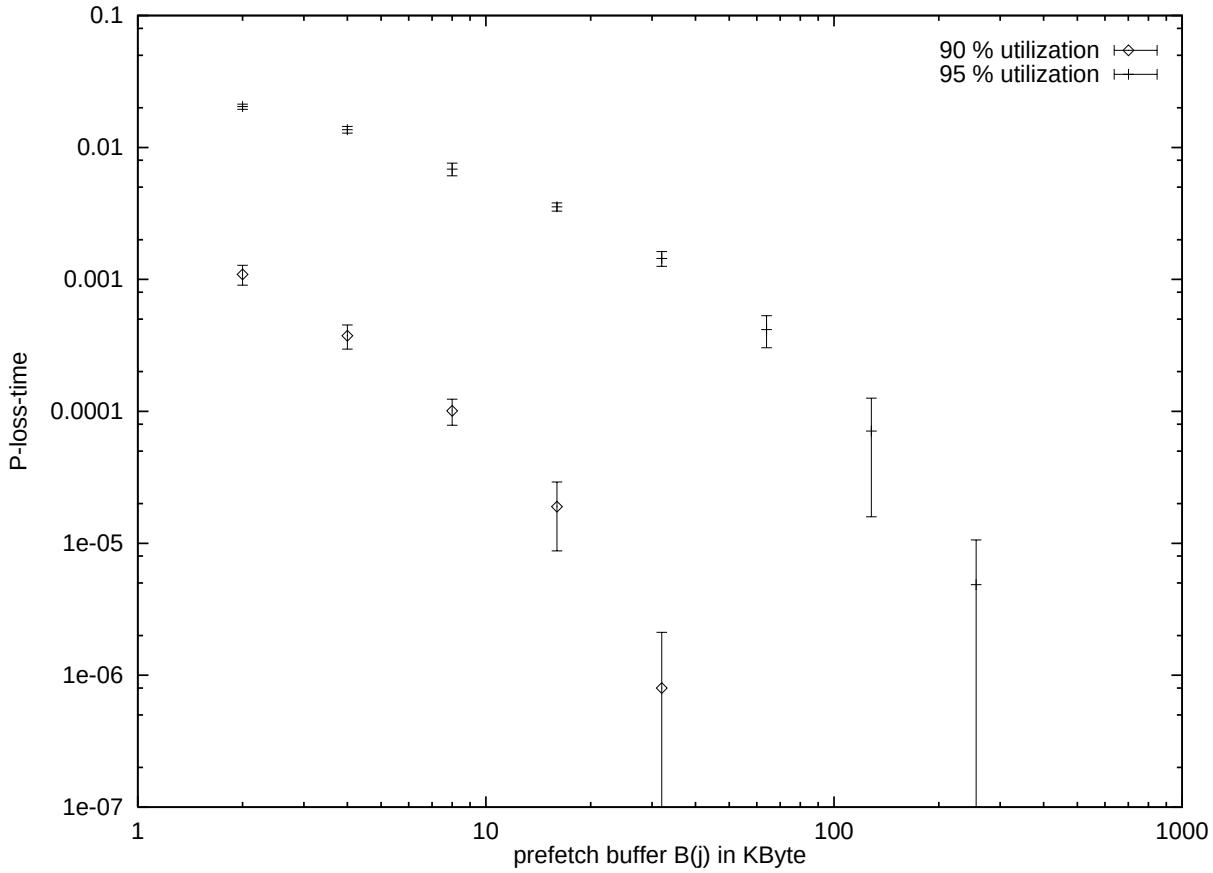


Figure 3.3: Confidence intervals of $P_{\text{loss}}^{\text{time}}$ for link utilizations of 90 % and 95 %. $P_{\text{loss}}^{\text{time}}$ is the fraction of frame times during which loss occurs.

for 64 KByte prefetch buffer. Whenever loss occurred, the buffer contents at each of the clients was nearly zero, confirming the existence of a strong pooling effect. (The existence of pooling is further justified by the analytical work of Reiman [56]).

Figure 3.3 demonstrates that with a small prefetch buffer at each client the JSQ prefetching policy will allow for almost 100% utilization with negligible packet loss. Also note that the scheme allows for near immediate playback of the video upon user request. It can be argued that for MPEG–2 traces with an order of magnitude larger average rate, the prefetch buffer will have to be an order of magnitude larger to achieve the same loss probabilities. But even with this order of magnitude increase, only 5 Mbytes of prefetch buffer is required to give negligible packet loss.

3.3.4 Efficient Implementation of the JSQ Prefetching Policy

In this subsection we describe a list based approach that allows for efficient implementation of the JSQ prefetch policy. We found in our experiments that it takes only 2.5 msec to execute the administrative steps necessary in each frame period for 132 video connections on a SPARCstation 2. We applied the refined stopping rule in these experiments and took also the buffer capacity test (3.4) into account.

The underlying data structure in our implementation is a singly linked list. Each list element is a record storing the data pertaining to an active connection; in particular, we keep track of $\theta(j)$, $p(j)$ and $b(j)$. We also keep track of an index indicating which video is being transmitted. Finally, each record contains a pointer to the next list element. The list is maintained such that the index $p(j)$ is ascending as one moves down the list, that is, the connection with the smallest number of prefetched frames is on the top. In each frame period we start by considering the connections at the top of the list. We first check whether conditions (3.3) and (3.4) (and also (3.5) when applicable) are satisfied. If theses conditions are satisfied, we transmit the frame, increment $p(j)$ and adjust $b(j)$. Next, we check whether the successor in the list has a smaller $p(j)$. If not, we try to prefetch the next frame. This is repeated until $p(j)$ is larger than that of the next connection in the list. At that point we have to rearrange the pointer references in order to maintain the order of the list. After the order has been restored we start over, trying to prefetch for

the connection on top of the list. If we find at any point that (3.3) or (3.4) (or (3.5) when applicable) is violated, we skip the connection and try to prefetch for the successor in the list, that is, we prefetch no longer for the connection on top of the list, but move down the list as we skip over connections.

3.4 Interactivity

In this section we adapt the JSQ prefetch policy to account for pauses as well as forward and backward temporal jumps. Our protocol allows these interactive actions to be performed with minimal delay. We assume that whenever a user invokes a interactive action, the client sends a message indicating the interactive action to the server.

Suppose that the user for connection j pauses the movie. Upon receiving notification of the action, the server can simply remove connection j from consideration until it receives a resume message from the client; while the connection is in the paused state, its prefetch buffer remains at a constant level. A slightly more complex policy would be to fill the corresponding buffer with frames once all the other prefetch buffers are full or reach a prespecified level.

Suppose that the user for connection j makes a temporal jump of δ frames into the future. If $\delta < p(j)$, we discard δ frames from the head of the prefetch buffer and set $p(j) = p(j) - \delta$. If $\delta \geq p(j)$ we set $p(j) = 0$ and discard all the frames in the prefetch buffer. Finally, suppose that the user for connection j makes a backward temporal jump. In this case we set $p(j) = 0$ and discard all frames in the prefetch buffer.

In terms of frame loss, pauses actually improve performance because the movies which remain active have more bandwidth to share. Frequent temporal jumps, however, can degrade performance since prefetch buffers would be frequently set to zero. Whenever we set a prefetch buffer to zero, the pool loses some of its total savings, thereby increasing the likelihood of loss.

We now present some numerical results for interactive actions. We consider only forward and backward temporal jumps and ignore pauses as pauses can only improve performance; we furthermore assume that $\delta > p(j)$ for all forward temporal jumps. Our results give therefore a conservative estimate of the actual performance. In our simulation, we

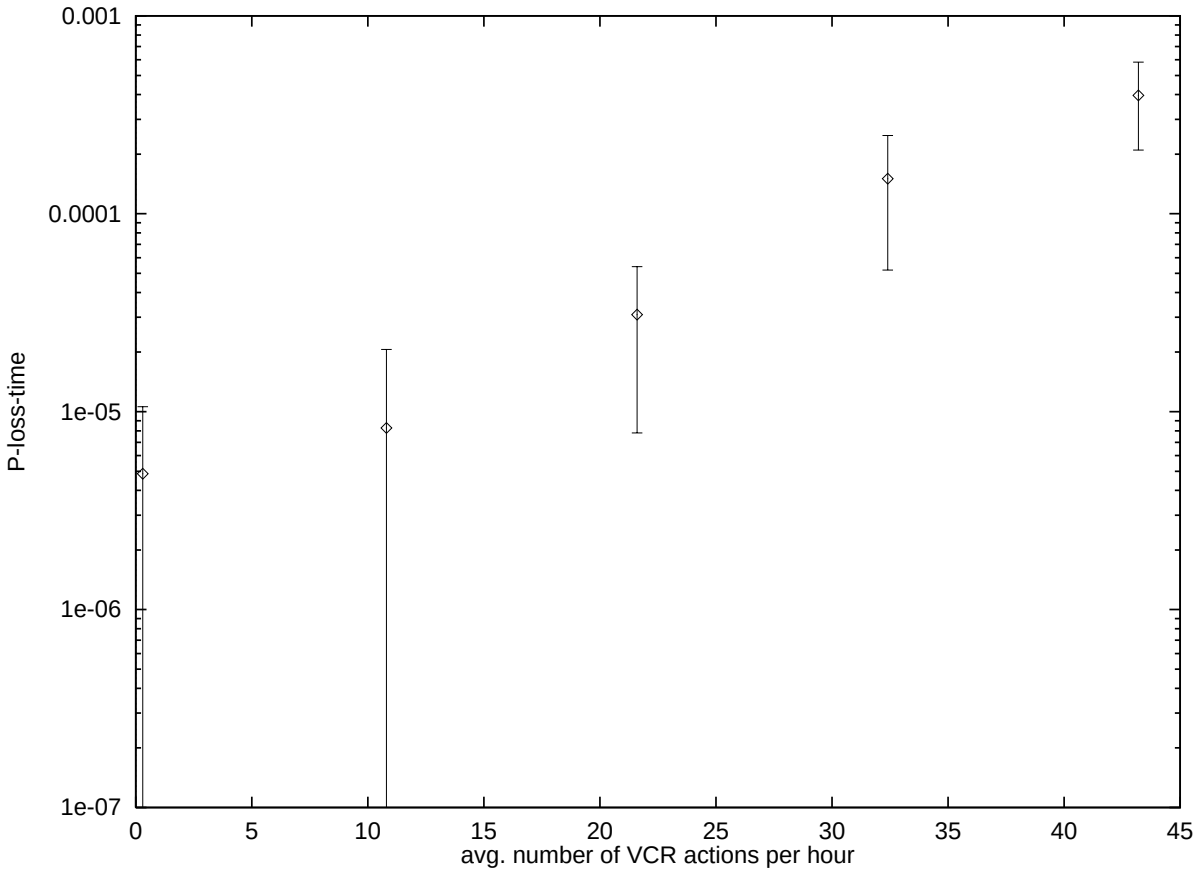


Figure 3.4: $P_{\text{loss}}^{\text{time}}$ as a function of the average number of temporal jumps per hour for 256 KByte buffers and 95 % link utilization.

assume that each user performs temporal jumps repeatedly, with the time between two successive jumps being exponentially distributed with constant rate. When a user performs such an action, her prefetch buffer is set to zero. Figure 3.4 shows the 90 % confidence interval for $P_{\text{loss}}^{\text{time}}$ for $11, 22, 32$ and 43 temporal jumps per hour (on average). This experiment was conducted with $B(j) = 256$ KBytes and a 95 % link utilization. As we would expect, loss probabilities increase as the rate of temporal jumps increase; however, the increase is not significant for a sensible number of temporal jumps.

3.5 Comparison between JSQ Prefetching and Optimal Smoothing

As mentioned in the Introduction, there are other protocols in the literature for the transport of VBR–encoded video which exploit prefetching and client buffers [24] [46] [47] [67] [15] [16]. In all of these other schemes, the transmission schedule for a given video is determined off–line and depends only on the traffic characteristics of that video; thus, the transmission schedule of a connection does not take into account the traffic requirements of the other connections in progress. For this reason, we refer to all the schemes cited just above as *non–collaborative* prefetching schemes.

On the other hand, the JSQ prefetching protocol determines the transmission schedule of a connection on–line, as a function of the buffer contents at *all* of the clients. For this reason, we refer to JSQ as a *collaborative prefetching scheme.* The purpose of this section is to show that the JSQ buffer fill policy is responsible for the outstanding performance reported in Section 3.3.3. To this end we shall compare JSQ prefetching with one of the non-collaborative schemes in the existing literature, namely, Optimal Smoothing [67] [79]. We apply the Optimal Smoothing algorithm to the traces used for the JSQ experiments (see Section 3.3.3). We then statistically multiplex the smoothed traces on a bufferless 45 Mbps link and calculate $P_{\text{loss}}^{\text{time}}$ using the Large Deviation (LD) approximation described in [57] and [79]. The LD approximation is known to be very accurate [62, 25, 13, 14, 57]. We do this for two versions of optimal smoothing: no initiation delay and a 10 frame initiation delay [67] [79] [11]. We chose Optimal Smoothing for our comparison, as optimal smoothing minimizes the peak rate and variability of the smoothed traffic for a given client buffer.

The results are given in Figure 3.5; only the point estimates of the simulation results are plotted here. The figure shows $P_{\text{loss}}^{\text{time}}$ as a function of buffer size at the client for 95% average link utilization. The figure shows that JSQ prefetching gives a $P_{\text{loss}}^{\text{time}}$ that is more than two orders of magnitude smaller than Optimal Smoothing for a 128 Kbyte buffer, and more than three orders of magnitude smaller for a 256 Kbyte buffer. Furthermore, for this case of 95% utilization, Optimal Smoothing has unacceptably high loss probabilities for all buffer sizes shown, whereas JSQ prefetching gave no loss in 4×10^8 simulated frame

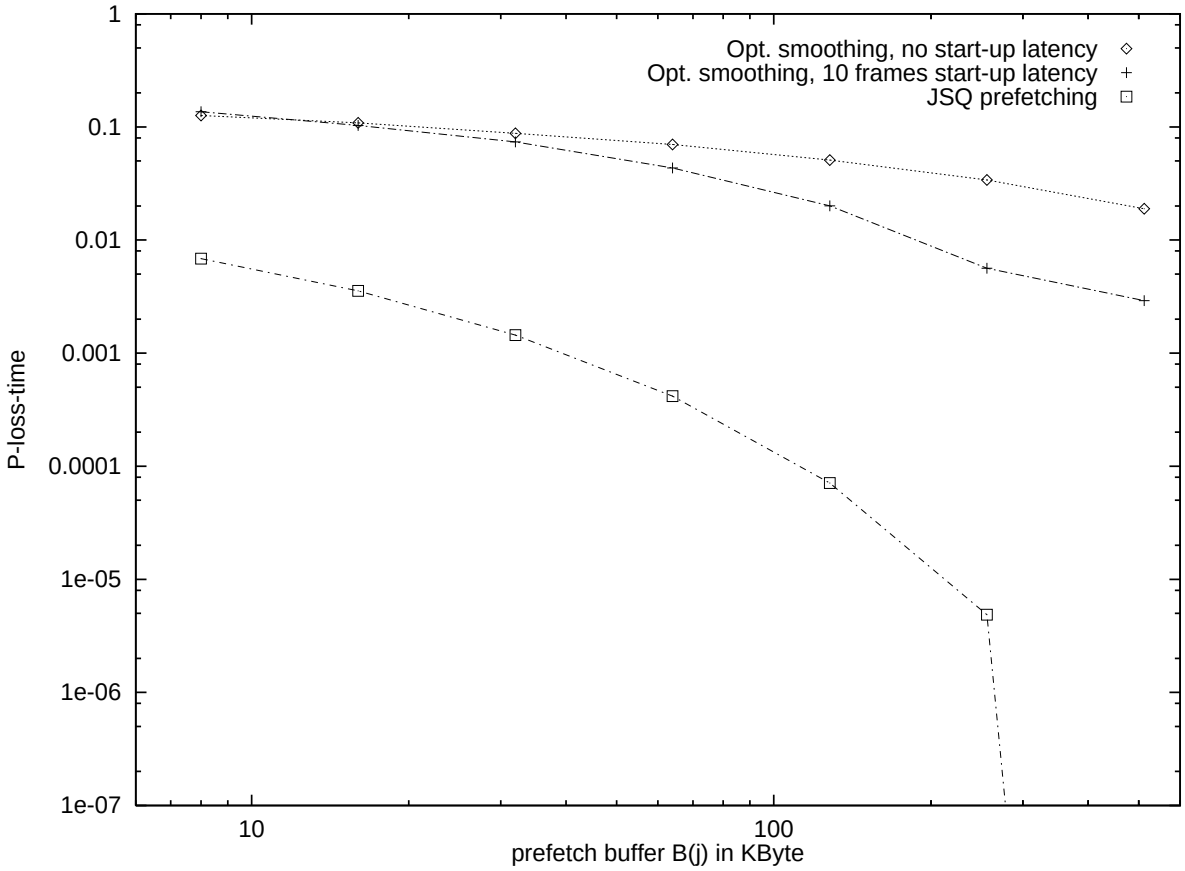


Figure 3.5: $P_{\text{loss}}^{\text{time}}$ as a function of buffer size for JSQ prefetching and statistical multiplexing of optimally smoothed traces for 95% average link utilization.

periods for a buffer size of 300 Kbytes. (To account for this point on the graph, we suppose that loss occurs for one of the 4×10^8 frame periods at 300 Kbytes.) We are thus led to the conclusion that the collaborative nature of JSQ prefetching contributes significantly to its high performance. JSQ prefetching is inspired by the least–loaded routing algorithm for circuit–switched loss networks, which is known to give excellent performance and to be extremely robust over a wide range of traffic conditions [65].

In Figure 3.6 we compare the admission region of JSQ prefetching, Optimal Smoothing and GOP smoothing. We use the lambs video, the burstiest of the mix of movies (see Table 3.1), for this experiment. We fix the client buffer size at 1 MByte. A client buffer of 1 MByte can store on average 41 seconds of the lambs video. We set the QoS criterion to $P_{\text{loss}}^{\text{time}} \leq 10^{-5}$. The plot gives the number of lambs connections that can be supported as a function of the normalized bandwidth of the bottleneck link. We define normalized

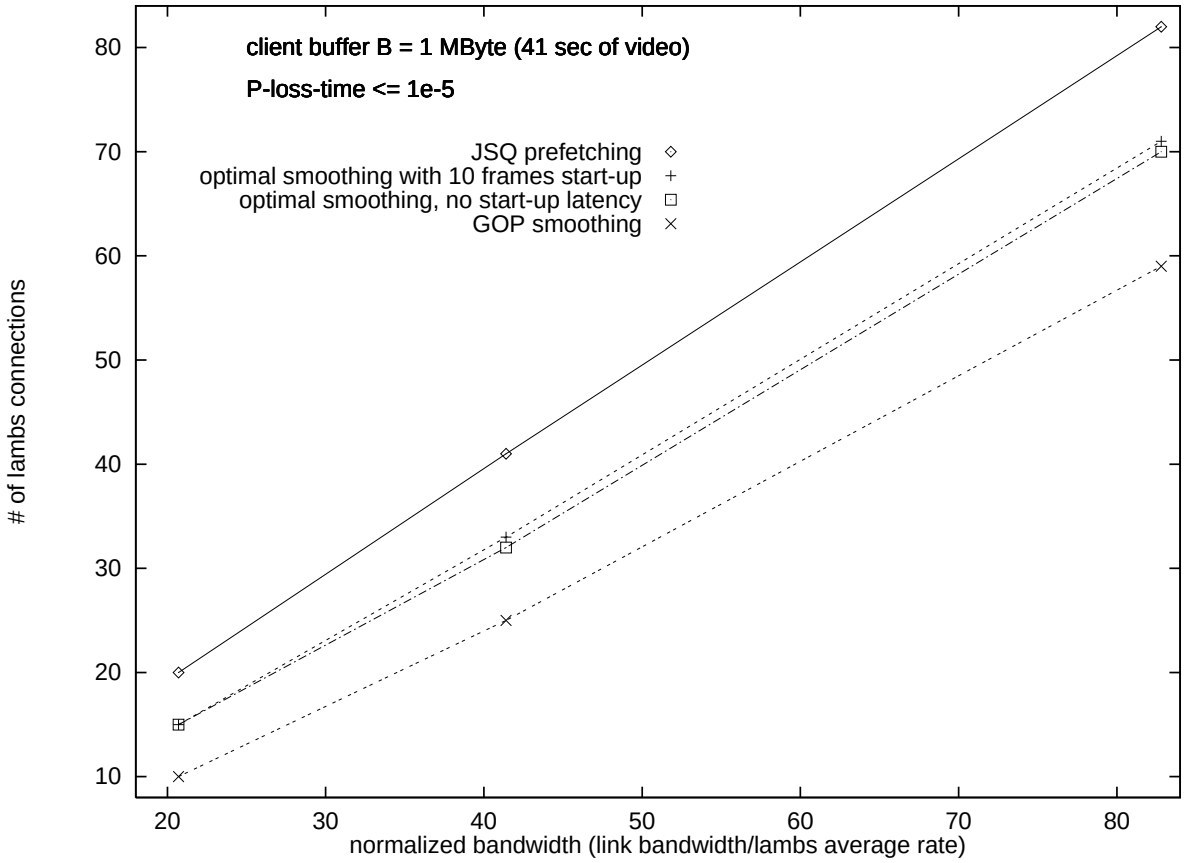


Figure 3.6: Number of lambs connections as a function of normalized bandwidth for JSQ prefetching, Optimal Smoothing and GOP smoothing. The client buffer of 1 MByte holds on average 41 seconds of the lambs video.

bandwidth as the link bandwidth divided by the average rate of the lambs video. The JSQ prefetching results are obtained from simulation. The Optimal Smoothing and GOP smoothing results are computed using the very accurate LD approximation. With GOP smoothing the video is smoothed over each Group of Pictures (GOP), that is, 12 frames in the case of the lambs movie (see Section 3.3.3). The plot shows that JSQ prefetching clearly outperforms the other smoothing schemes. With a normalized bandwidth of 21.4, corresponding to a link rate of 4 Mbps, JSQ prefetching can support 20 lambs connections (95 % average link utilization) while Optimal Smoothing can support 15 connections (70 % average link utilization). GOP smoothing can support 10 connections (48 % average link utilization).

We furthermore note that the JSQ policy allows for instantaneous interactive actions,

making the scheme amenable to highly interactive multimedia applications. These interactions cause only a modest drop in performance (see Section 3.4). An indepth study of interactivity for Optimal Smoothing is given in [11]. In general, no smoothing scheme can always give instantaneous playback after all temporal jumps, because for some playback points a build–up delay is required to ensure no starvation in the future. In [11] an algorithm is given for which the build–up delay is typically very small for the traces considered. However, this build–up delay typically increases with the client buffer size, and at playback points before long high–bandwidth scenes the delay can be large (an example is given for which the delay after a temporal jump can be as much as 63 seconds for a 4Mbyte buffer). Thus, with Optimal Smoothing, as the client buffers become larger, the link utilizations increase but interactivity performance decreases. In contrast, with JSQ prefetching, both link utilization and interactivity performance improve with increasing client buffer.

We now explain intuitively why packet loss is less with JSQ prefetching. The JSQ policy strives to (1) equalize the number of prefetched frames over all ongoing connections and (2) keep the buffers always filled. Optimal Smoothing, on the other hand, first fills the buffer when a connection starts up. The buffer is then emptied, then filled again and so on; the buffer content thus oscillates between empty and full.

Now consider a situation where a number of connections have been in progress for a while and a new connection becomes active. The JSQ policy dedicates the entire link capacity temporarily to the new connection so as to bring the number of prefetched frames up to the level of the other connections. These other connections are fed from their prefetch buffers during this period. For loss to occur, the prefetch buffers of the older connections need to be drained completely and the aggregate rate of the ongoing connections has then to exceed the link capacity. This is a highly unlikely scenario given that the prefetch buffers typically hold a large number of prefetched frames. With optimal smoothing, however, the new connection gets only a small fraction of the link capacity since the other connections have to transmit according to their fixed schedule. Losses are therefore more likely, particularly for movies requiring a large amount of bandwidth in the beginning.

Next, consider a situation where a number of connections have been active for a while and high action scenes in the movies collude and thus cause temporarily an excessive demand for bandwidth. Since JSQ prefetching keeps the buffers always full while the optimal

smoothing policy drives the buffer contents periodically up and down, the aggregate buffer contents of all the ongoing connections is larger with JSQ prefetching. The JSQ policy can therefore support longer periods of excessive demand for bandwidth.

We conclude this section by mentioning that the non–collaborative prefetching schemes do have an important advantage over our JSQ prefetching protocol — namely, they can be implemented over multiple, decentralized servers, and they can be easily adapted to run over a network with multiple bottleneck links. In particular, the server does not need to be attached to a cable headend or an ADSL switch, but can instead be distributed and placed deeper into the network. In our current research [60], we are adapting the JSQ prefetching protocol so that the server can be distributed and placed deeper into the network.

3.6 A Packet–Based JSQ Prefetch Policy

Up to now we have measured the length of the queue in a prefetch buffer by the number of frames present in the buffer. An alternative measure for length is the number of packets in the buffer. With this measure, the JSQ policy strives to equalize the number of packets in each of the prefetch buffers. In defining this policy, we drop the subscript l from all notations. At the beginning of each slot the server determines the j^* with the smallest $b(j) - x_{\theta(j)}(j)$, transmits one packet from connection j^* and increments $b(j^*)$. Within this same slot the server repeats this procedure over and over again, at each iteration finding a new j^* that minimizes $b(j) - x_{\theta(j)}(j)$, transmitting a packet from connection j^* and incrementing $b(j^*)$. For infinite prefetch buffers, the procedure stops when z (the total number of packets transmitted within the slot) equals R/F. For finite prefetch buffers, the procedure stops when $z = R/F$ or when all the prefetch buffers are full.

The packet–based JSQ policy is easier to implement than the frame–based JSQ policy. For brevity, we only describe this implementation for the case of infinite prefetch buffers. At the beginning of each slot we order the prefetch buffers according to their $y[j] = b(j) - x_{\theta(j)}(j)$ values. Let $\alpha(j)$, $j = 1, \ldots, J$, be this ordering. We fill buffer $\alpha(1)$ with $y[\alpha(2)] - y[\alpha(1)]$ packets. We then fill buffers $\alpha(1)$ and $\alpha(2)$ with $y[\alpha(3)] - y[\alpha(2)]$ packets. The procedure continues as long as $z \leq R/F$. If at any iteration we will have $z > R/F$, we spread the remaining packets evenly over the buffers in question.

The drawback of this policy is that it can produce frame levels in the prefetch buffers which are highly unbalanced; this can occur when one set of connections has a large number of packets per frame and a second set has a small number of packets per frame. An advantage of this policy is that it can fill all the prefetch buffers to the brim when drain rate is far below the link rate.

Throughout the remainder of this chapter we use the original frame–based JSQ policy.

3.7 Selective Discard Policies for MPEG Compression

There is cell loss in a slot l if and only if

$$\sum_{j \in \mathcal{J}} x_{\theta_l(j)}(j) > R/F, \tag{3.7}$$

where $\mathcal{J}$ is the set of connections that have no frames in their buffers at the beginning of the slot. However, when (3.4) holds, the server still has the freedom to select which part of the aggregate traffic,

$$\sum_{j \in \mathcal{J}} x_{\theta_l(j)}(j),$$

to transmit and which part ot discard. A rule for choosing which part of the aggregate traffic to discard is a *selective discard policy*. A good selective discard policy will discard the parts of the aggregate traffic that can most easily be estimated at the client. The estimation of lost video traffic at the client is called *error concealment*, which, for MPEG encoding, has been thoroughly discussed in the literature [43, 4, 80].

3.7.1 Selective Discard Strategies for MPEG–1

Recall that for MPEG–1 encoding the frames have a GOP pattern, e.g., IBBPBBPBB. Because B and P frames are encoded with respect to the I frames, it is important to minimize loss for each I frame, as this loss propagates through the entire GOP. Similarly, loss for P frames is more damaging than loss for B frames. These observations lead to the following simple selective discard policy. Let $\mathcal{J}(I)$, $\mathcal{J}(P)$ and $\mathcal{J}(B)$, be the set of connections in $\mathcal{J}$ of the current slot that are to transmit I, P and B frames. Note that the

sets $\mathcal{J}$, $\mathcal{J}(I)$, $\mathcal{J}(P)$ and $\mathcal{J}(B)$ change from slot to slot. If

$$\sum_{j \in \mathcal{J}(I)} x_{\theta_l(j)}(j) + \sum_{j \in \mathcal{J}(P)} x_{\theta_l(j)}(j) \leq R/F, \tag{3.8}$$

then we transmit all the frames in $\mathcal{J}(I) \cup \mathcal{J}(P)$ and discard "part" of the traffic from $\mathcal{J}(B)$. If (3.8) fails, but

$$\sum_{j \in \mathcal{J}(I)} x_{\theta_l(j)}(j) \leq R/F, \tag{3.9}$$

then we transmit all the frames in $\mathcal{J}(I)$, discard all the frames in $\mathcal{J}(B)$ and discard "part" of the traffic from $\mathcal{J}(P)$. Finally, if (3.9) fails, we discard all of the frames in $\mathcal{J}(B) \cup \mathcal{J}(P)$ and "part" of the traffic in $\mathcal{J}(I)$. The above discard policy has not been fully defined, as we have not yet specified the "part" of traffic that is discarded in each of the cases. There are several natural strategies here:

1. Slice discarding strategy: Suppose it suffices to discard traffic only from $\mathcal{J}(B)$. Then we discard slices from the B frames, spreading the discarded frames over the connections in $\mathcal{J}(B)$; furthermore, if we must discard multiple slices in a frame, we attempt to discard non–adjacent slices in order to facilitate estimation and concealment at the client. If discarding all the B frames in $\mathcal{J}(B)$ does not suffice, then we also discard slices from P frames; finally, if discarding all the B and P frames from $\mathcal{J}(B) \cup \mathcal{J}(P)$ does not suffice, then we also discard slices from I frames.

2. Frame discarding strategy: Discard entire frames in a round robin fashion across all the videos in $\mathcal{J}$. With this strategy, loss of consecutive frames of a specific connection should be rare.

3. Do not discard the frames or any "parts" of the frames, but instead to re–encode the frames on–the–fly with a coarser quantization. A related strategy is to include in the server's storage a second version of each video which has been off–line encoded at a lower rate; when $\mathcal{J}$ is nonempty and loss is unavoidable, then the server switches to the low–rate versions for all videos in $\mathcal{J}$.

Figure 3.7 illustrates the performance of this discard policy. In the simulation we simplify the policy by never transmitting any "parts" of the traffic; specifically, we drop all frames

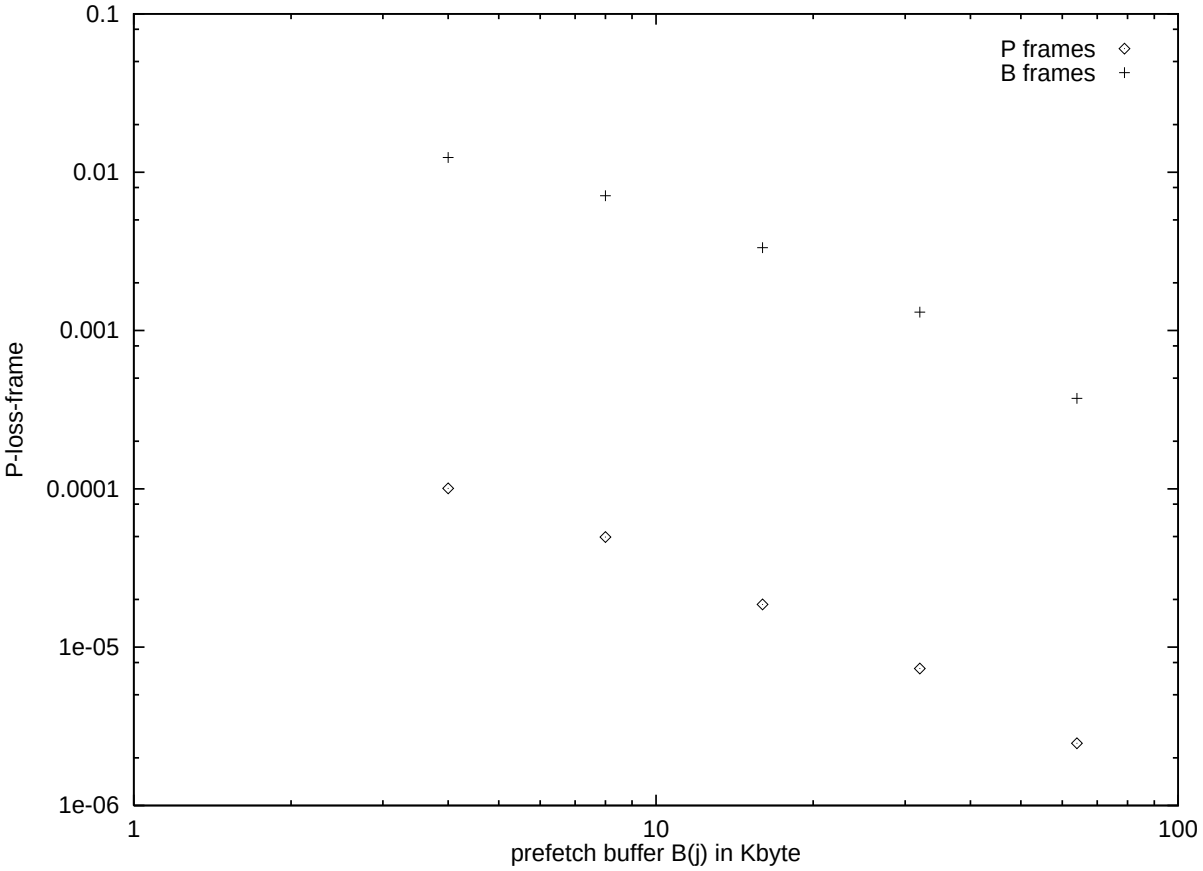


Figure 3.7: The fraction of lost P and B frames as a function of the prefetch buffer size for 95 % utilization.

from $\mathcal{J}(B)$ when (3.8) holds, all frames from $\mathcal{J}(B) \cup \mathcal{J}(P)$ when (3.8) fails and (3.9) holds; and all frames from $\mathcal{J}$ when (3.9) fails. We conducted this experiment with a link utilization of 95% and ran 1000 replications for buffer sizes ranging from 4 to 64 KByte. The figure plots the point estimates of the fraction of P and B frames that are lost (we ignore confidence intervals to avoid visual clutter). In all our simulations we never observed any loss of I frames.

3.7.2 An Admission Policy for MPEG–1

We now present a connection admission policy which, when combined with our selective discard policy, ensures that I frames experience practically no loss. To this end, observe that with our selective discard policy the loss probability for I frames will be less than that when the clients have no buffer and the traffic in the P and B frames is equal to zero.

For this conservative model, we can adapt the theory in [57] to construct a conservative large–deviation approximation for the loss of I frames. Using this approximation, we only admit a new connection if (1) the fraction of frame periods which have I–frame loss remains less than some minute number, say, 10^{-9}, and (2) the link utilization remains below, say, 95 %. This admission policy essentially guarantees no loss of I frames while simultaneously ensuring a low probability of loss for the B and P frames.

We give now a brief outline of how the large–deviation approximation applies to our scheme. First, for each video in the server we construct a modified trace $y_n(j), j = 1, \ldots, J$, with $y_n(j) = x_n(j)$ if the nth frame is an I frame and $y_n(j) = 0$ if the nth frame is either a P or B frame. We then calculate the frame size distribution for the modified trace:

$$\pi_j(l) := \frac{1}{N} \sum_{n=1}^{N} 1(y_n(j) = l),$$

The logarithmic moment generating function of the aggregate amount of traffic in a frame for the modified trace is:

$$\mu_Y(s) := \ln E[e^{sY}],$$

where

$$Y = \sum_{j=1}^{J} Y(j)$$

and $Y(j)$ is distributed according to $\pi_j(\cdot)$. The probability that at least one I frame is lost in an arbitrary frame period is approximated as

$$\begin{aligned} P_{\text{loss}}^{\text{time}}(I) &= P(\text{“all I frames cannot be transmited in a frame period”}) \qquad (3.10) \\ &\approx \frac{1}{s^\star \sqrt{2\pi \mu_Y''(s^\star)}} e^{-s^\star R/F + \mu_Y(s^\star)}, \end{aligned}$$

where s^* satisfies

$$\mu_X'(s^\star) = R/F.$$

The admission control test is thus satisfied if

$$\frac{1}{s^\star \sqrt{2\pi \mu_Y''(s^\star)}} e^{-s^\star R/F + \mu_Y(s^\star)} \leq \epsilon,$$

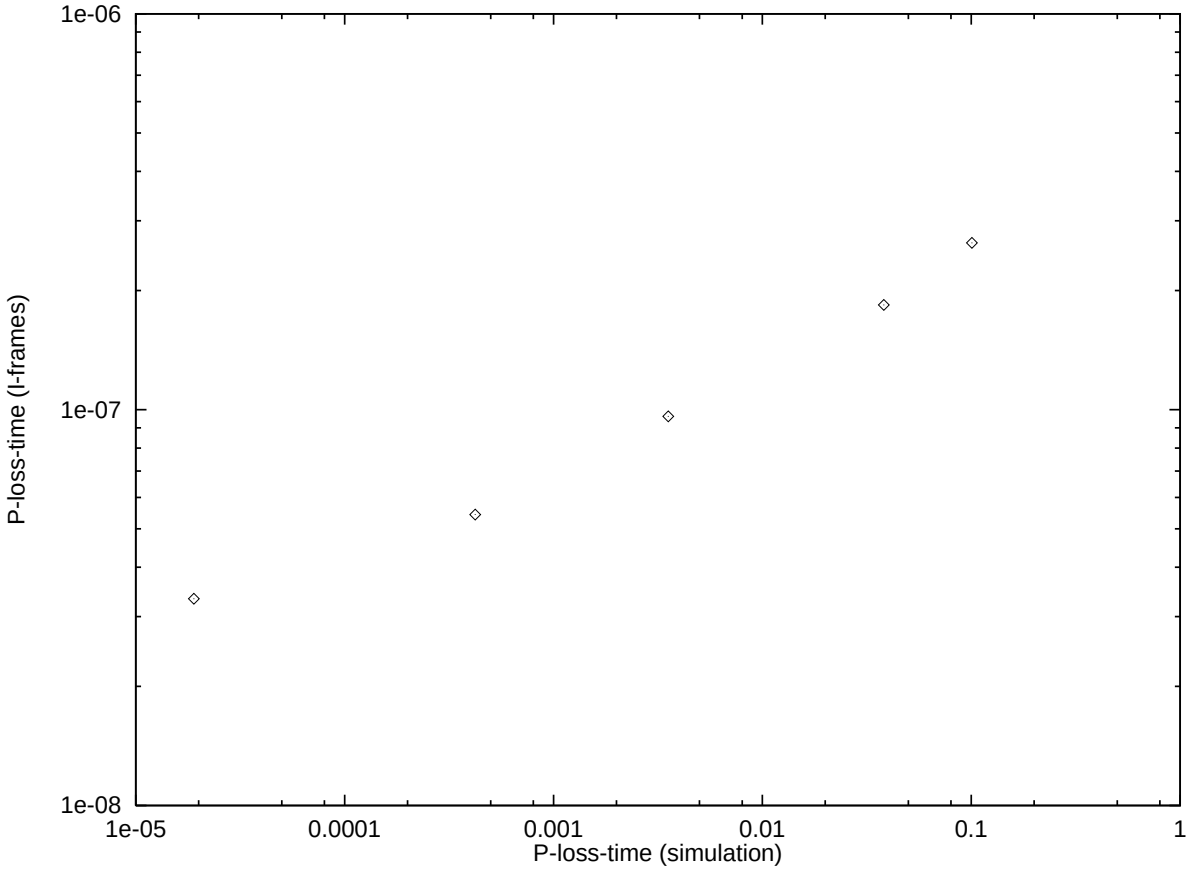


Figure 3.8: $P_{\text{loss}}^{\text{time}}(I)$ as a function of $P_{\text{loss}}^{\text{time}}$ for 16 KByte prefetch buffer.

where ϵ is some minute number such as 10^{-9}.

Figure 3.8 shows the relationship between the $P_{\text{loss}}^{\text{time}}$ (the fraction of frame periods during which loss occurs for I,P, or B frames) and $P_{\text{loss}}^{\text{time}}(I)$ (the conservative estimate for the fraction of frame periods during which loss of an I frame occurs when using selective discard). In this figure with have used a 16 KByte prefetch buffer at each of the clients. The series of points was obtained by varying the link utilization. (For visual simplicity we ignore the confidence intervals for $P_{\text{loss}}^{\text{time}}$ and only report the mid–point in the figure.) From this curve we see, for example, that when $P_{\text{loss}}^{\text{time}}$ is 2×10^{-5} then the fraction of frame periods during which I frames are lost is 3×10^{-8}. The values of $P_{\text{loss}}^{\text{time}}(I)$ reported in this figure come from the large–deviation approximation. This approximation is conservative, as it assumes that the prefetch buffers are zero; the actual loss probability for I frames is much less. In fact, for the last point (as well as the other points) on the graph, corresponding

to $P_{\text{loss}}^{\text{time}} \approx 0.1$ and 99.5% utilization, we observed absolutely no loss of I frames in the simulation!

3.7.3 MPEG–2

The MPEG–2 standard is similar to MPEG–1 but includes extensions and refinements to cover a wider range of applications. While MPEG–1 was primarily introduced for CD–ROM and has a typical bitrate of 1.5 Mbps, MPEG–2 is intended for the digital transmission of broadcast TV quality video with bitrates between 4 and 9 Mbps.

The most important extension of MPEG–2 over MPEG–1 for our discussion is scalability. Scalability offers the possibility to partition the video stream into two different layers: the base and enhancement layers. This allows for transmission with different priorities. The enhancement layer conveys refined image information at a lower priority. Transmitting a basic quality image at the base layer with high priority ensures that a minimum quality image can be decoded at the client. If both layers are received, a high quality image can be displayed. If the low priority enhancement layer is lost, the client will at least be able to display a basic quality image (provided the base layer is received intact).

Although there are four scalability modes we shall focus on SNR scalability as it offers a flexible way to partition the video data while keeping the encoder and decoder architecture simple. The basic idea of SNR scalability is to first generate the base layer by quantizing the DCT coefficients of a frame with a coarse base quantization matrix. The difference between the original DCT coefficients and the DCT coefficients reconstructed from the base layer is quantized with a finer quantization matrix and transmitted in the enhancement layer. If the low priority enhancement layer is lost, the frame is decoded using only the base layer.

An admission control policy for MPEG–2 video may proceed similar to the one outlined for MPEG–1 video in subsection 3.7.2. Again, we conduct 2 tests:

1. The admission control test for a bufferless link outlined under 3.7.2 with a very small QoS parameter ϵ such as 10^{-9}. Consider only the base layer in this test.

2. Taking base and enhancement layer into account, check whether the average link utilization is less than 0.95.

The first test practically guarantees the delivery of the base–layer image.

3.8 Video Delivery to the Home

3.8.1 ADSL

In this section we discuss how JSQ prefetching ties into the Asymmetric Digital Subscriber Line (ADSL) technology. First, we give a brief overview of the ADSL technology; for more details see [18, 44]. ADSL exploits advances in digital signal processing to provide a high speed downstream channel from the central office to the home (up to 9 Mbps total capacity) and a lower rate upstream channel (up to 640 kbps total capacity) over a single twisted pair copper loop while leaving the POTS (Plain Old Telephone Service) unaffected. The high speed downstream channel provides a switched point–to–point link for delivering video from the central office to the home. The lower rate upstream channel can be used to control the downstream video stream; in particular, the upstream channel can be used to convey the request for a video and VCR actions to the video server. The downstream channel capacity provided by the ADSL technology, however, depends on the length of the copper loop, wire gauge and interference. The maximum downstream rate of 9 Mbps can be supported over approximately 3,000 meter of regular copper wire. As the wire length increases to 6,000 meter the available downstream capacity drops to 1.5 Mbps due to increased attenuation. The bandwidth requirements for VBR transmission of video can readily be accommodated by ADSL for homes within 3,000 meter of the central office since the peakrate of VBR video is typically well below 9 Mbps.

Figure 3.9 shows the architecture of a VoD system utilizing ADSL. The video server multiplexes the video streams onto an essentially bufferless link. The videos may be transported in ATM cells or TCP/IP packets. The packet switch in the central office forwards the cells/packets to the appropriate output buffer. The cells/packets are then transmitted to the individual homes over the point–to–point ADSL operated copper lines. The ADSL Forum is developing interface and protocol guidelines for three basic distribution protocols [17] [72]:

1. The ADSL modem is configured for continuous bit–rate synchronous transmission;

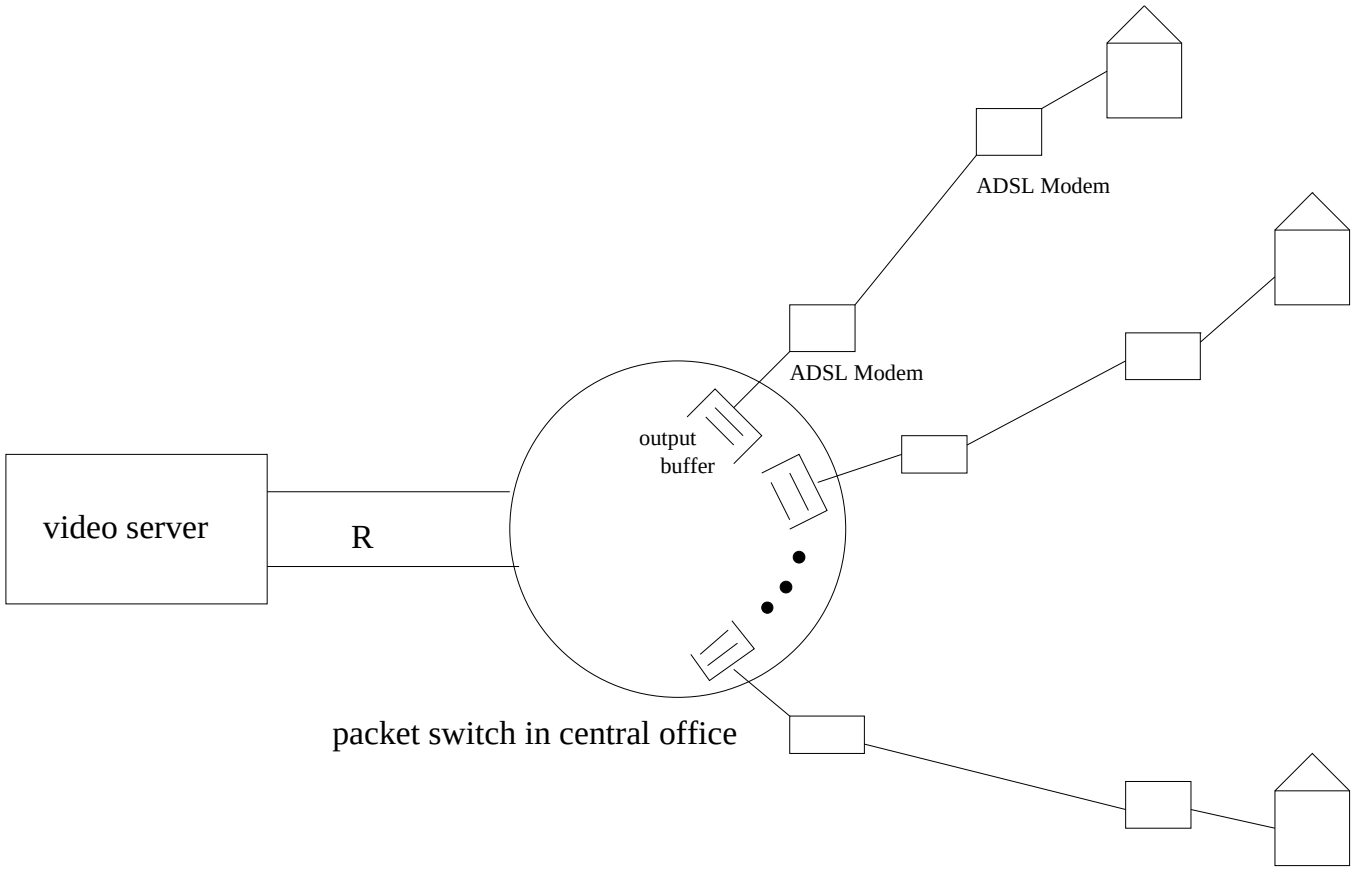


Figure 3.9: VoD architecture with ADSL

.

thus, the ADSL modem works as a bitpump.

2. The modem is configured for packet transport. The ADSL modem is configured for Ethernet packets in this case.

3. The ADSL modem is configured for an ATM interface.

In this chapter we are focusing on packet or ATM cell based transmission. Hence, the following transport modes are possible on the link joining the video server and the central office and on the ADSL link joining the central office and the home:

1. ATM cells from server through switch to home

2. IP datagrams from server through switch. After switch datagrams are encapsulated in Ethernet frames.

3. ATM cells from server through switch. After switch, cells are converted to IP packets.

This last option has the advantage of using powerful ATM switching technology in the central office while allowing the user to have inexpensive Ethernet equipment.

3.8.2 Cable

In this section we discuss how the JSQ prefetching protocol for VoD proposed in this chapter ties into the cable modem technology. First, we give a brief overview of the cable modem technology; for more details see [37, 38, 49]. The coaxial cable was installed for broadcast of one–way analog TV. Cable is a shared medium; many homes are attached to the same coaxial cable. Medium access control for the downstream video traffic is particularly simple as there is only one sender, the headend. The upstream control traffic, however, poses a problem. Carrier sensing fails for cable plants with tree–and–branch structure, where only the headend hears every source. Remedies for this problem are currently being developed [37]. As of the writing of this chapter, there are no fixed standards that specify how upstream and downstream bandwidth are allocated to homes. Typically, the upstream traffic is transmitted in the 5–40 MHz range. The downstream bandwidth from $40 - 750$ MHz is split into 6 MHz channels for analog TV. Each of these channels yields approximately 25 Mbps when 64 Quadrature Amplitude Modulation (QAM) is employed and could thus carry a couple of video streams.

Figure 3.10 shows a possible VoD architecture with cable. The video server is attached directly to the cable headend as are multiple cable trunks. We assume in this architecture that the link connecting the video server to the headend has infinite bandwidth. The bottleneck link is the cable trunk connecting the homes to the headend. Homes are attached to cable trunks via cable modems. The request for a video is relayed from the viewers home to the headend via the upstream channels. The headend forwards the request to the video server. The video server immediately starts transmitting the video frames. The video server runs the JSQ algorithm for each cable trunk. The switch in the headend forwards the frames to the appropriate output queue. All the videos requested by viewers connected to the same cable trunk are multiplexed onto the shared channel of capacity, say R bps. Our JSQ prefetching protocol allows for the efficient use of the valuable trunk bandwidth,

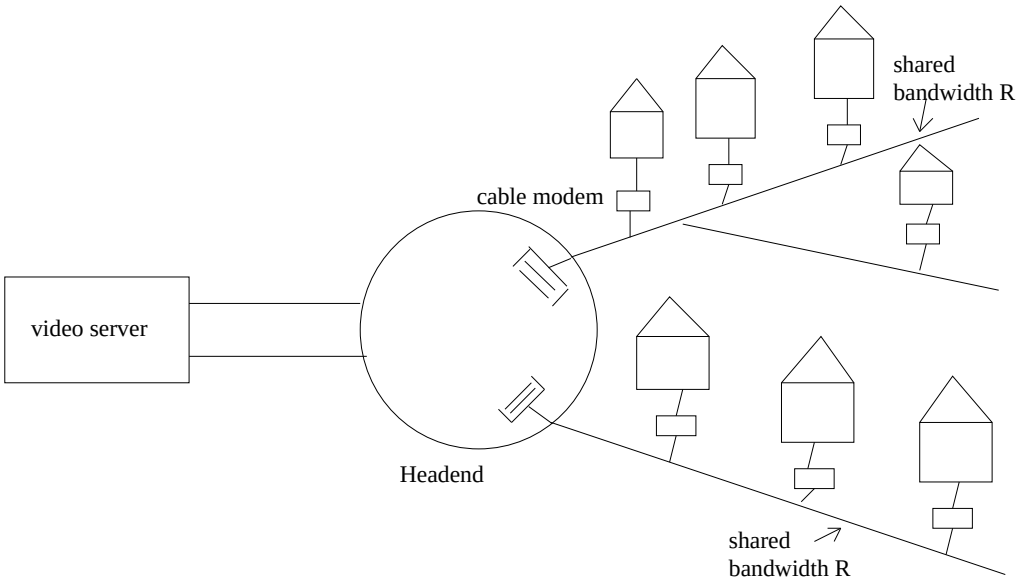


Figure 3.10: VoD architecture for cable residential access

.

R. We achieve transmission with negligible losses and thus constant high video quality for average trunk bandwidth utilizations of 95%.

3.9 Conclusion

Prerecorded video has two special properties: (1) for each video, the traffic in each video frame is known before the video session begins; (2) while the video is being played, some of the video can be prefetched into the client memory. In this chapter we have shown how these two properties can be exploited to achieve high performance when there is one shared link between the server and the clients. We have also shown how selective discard can enhance the performance when all of the video connections originate from the same server. The results should be useful for designing VoD systems that connect servers to residential broadband networks using cable or ADSL, or for VoD systems that connect the server to its clients through a LAN. Our client–server scheme can be part of a larger Internet solution to VoD, whereby the prerecorded videos are multicast to local servers at off–peak hours with the best effort service. In the next chapter we develop a variation of the JSQ prefetch protocol that allows the server to be distributed and to be placed deeper into the network.

Chapter 4

Decentralized Prefetching

4.1 Overview

In this chapter we present high–performance *decentralized* prefetching protocols for the delivery of video on demand (VoD) from servers to clients across a packet–switched network. The protocol assumes that the videos are variable–bit–rate (VBR) encoded. Not only does this protocol give constant perceptual quality for high link utilizations, but it also allows for immediate commencement of the video upon user request and near instantaneous response to viewer interactions such as pause, resume and temporal jumps.

To achieve this high performance our protocol has two requirements. First, we require that each client has a moderate amount of memory dedicated to the VoD application. Second, we require that each client sends a positive acknowledgement back to its server for each received video frame. The client could be a television with a set–top box capable of performing buffering and decoding, or it could be a household PC.

We have demonstrated in the previous chapter that JSQ prefetching achieves nearly 100% link utilization, immediate commencement of playback and instantaneous response to viewer interactions. JSQ prefetching, however, can only be applied when one centralized server feeds many clients. In this chapter we introduce decentralized and collaborative prefetching protocols that allow the video streams to emanate from multiple distributed and decentralized servers.

Our decentralized prefetching protocols perform almost as well as as JSQ prefetching:

they allows for nearly 100% link utilization, immediate commencement of playback and instantaneous response to viewer interactions.

Our decentralized prefetching protocol employs window flow control; it is inspired by the Transmission Control Protocol (TCP) [29, 2] widely used in the Internet. For simplicity, assume that each server is responsible for exactly one connection. Admission control ensures that all link utilizations do not exceed 95%. Our basic decentralized prefetching protocol works roughly as follows. The server maintains a send window, limiting the number of frames the server is allowed to send in a frame period. The send window is increased by a small increment when all acknowledgments arrive in time. Due to admission control and the VBR nature of the traffic, there are periods of time during which the network is underutilized. The send window grows larger than one during these periods, allowing the server to prefetch future frames into the client memory. In times of network congestion, frames are lost or delayed and the corresponding acknowledgements do not arrive at the server before their timeouts. In this case, the send window is reduced to throttle the server and alleviate the congestion. The reservoir of prefetched frames in the client buffer allows the client to continue playback during these periods of congestion. Starvation at the client occurs only if the reserve of prefetched frames at the client is completely depleted and the current frame is lost or delayed due to network congestion. We simulate our protocol in the context of a simple network (see Figure 4.1). The simulations are driven by frame size traces of MPEG 1 encoded videos from the public domain [64]. *Our empirical work indicates that starvation at the client rarely occurs for average link utilizations around 95% and small client buffers.*

This chapter is organized as follows. In the following subsection we briefly summarize the JSQ protocol of Chapter 3 and review Optimal Smoothing. In Section 4.2 we describe our VoD architecture. In Section 4.3 we introduce our decentralized prefetching protocol. In Section 4.4 we introduce a number of refinements of the decentralized prefetching protocol. In Section 4.5 we present simulation results for our decentralized prefetching protocol. In Section 4.6 we extend our decentralized prefetching protocol to allow for priorities and present numerical results for this modification of the prefetching protocol. In Section 4.7 we discuss how our client–server protocol can be used with cable residential access. In Section 4.8 we conclude and discuss future work.

4.1.1 Review of Transmission Schemes for VBR Video on Demand

In this subsection we summarize, Join–the–Shortest–Queue (JSQ) Prefetching and review Optimal Smoothing [66, 79, 61]. These two schemes will be used as benchmarks when evaluating our decentralized prefetching protocol.

The JSQ prefetching protocol is suited for the efficient transmission of VBR encoded videos from a video server to a large number of clients with moderate memory. The protocol allows for at most one shared link between the video server and the clients. The policy is based on the observation that due to the VBR nature of the multiplexed traffic there are frequent periods of time during which the shared link's bandwidth is under utilized. During these periods the server prefetches frames from any of the ongoing connections and sends the prefetched frames to the buffers in the appropriate clients. The JSQ policy specifies how the server selects the prefetched frames. The server always selects the next frame from the connection that has the smallest number of prefetched frames in its client's buffer. The JSQ prefetching protocol thus determines the transmission schedule of a connection on–line, as a function of the buffer contents at *all* of the clients. For this reason, JSQ is referred to as a *collaborative prefetching scheme.*

Optimal Smoothing can be applied when transmitting stored video from a server to a client with buffering capabilities across a network. Given a specific client buffer size, the optimal smoothing algorithm determines a "smooth" rate transmission schedule that ensures that the client buffer neither overflows nor underflows. The algorithm is optimal in that it achieves the greatest possible reduction in rate variability. Optimal smoothing is non–collaborative; the transmission schedule is computed before transmission begins and thus does not take the other ongoing connections into account. Admission control for the optimally smoothed trace can be based on the peak–rate of the smoothed trace; this ensures lossless transmission. Another approach is to statistically multiplex the optimally smoothed traces at an unbuffered link and base admission control on a large deviation estimate of the loss probability [57, 79]. We apply the latter approach when comparing optimal smoothing with our decentralized prefetching protocol.

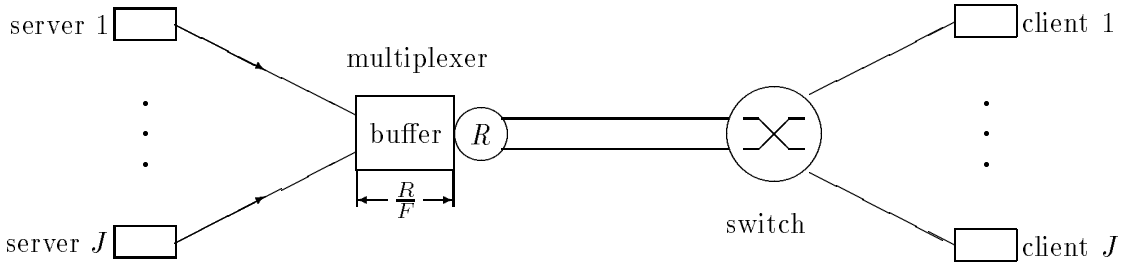


Figure 4.1: Decentralized Video on Demand Architecture.

4.2 Architecture Description

Figure 4.1 illustrates our basic model for decentralized VoD[1]. The video servers contain videos in mass storage. For notational simplicity, assume that each video consists of N frames and has a frame rate of F frames/sec. The videos are VBR encoded using MPEG 1, MPEG 2 or some other video compression algorithm. Let J denote the number of video connections in progress. We assume for the purpose of this study that each video server feeds one client; thus there are J video servers feeding J clients. In explaining the client–server interaction, we focus on a particular client–server pair. For simplicity, we assume for the following discussion that each video frame is transmitted in one packet[2]. Let x_n denote the number of bits in the nth frame. Because the videos are prerecorded, the sequence $(x_1, x_2, \ldots, x_N)$ is fully known before the transmission of the video. At the beginning of each frame period, that is, every $1/F$ seconds, the server decides according to a prefetching policy, outlined in the next section, which and how many frames to transmit. The server sends the frames to the multiplexer buffer. Frames that do not fit into the multiplexer buffer are lost. The multiplexer buffer of size R/F bit is served at rate R bps. The maximal delay incurred in the multiplexer is therefore $1/F$ seconds. For simplicity we assume that the propagation and processing delays are negligible. The client instantaneously sends a positive acknowledgment to the server for each frame received.

[1] Although we discuss our protocol in the context of a single shared link, the protocol applies to arbitrary networks with multiple shared links.

[2] In our numerical work we assume the more realistic case of fixed size packets.

With these delay assumptions, the server receives acknowledgments for all frames successfully received by the client within one frame period. The server therefore knows whether the frames sent in the previous frame period were received before deciding which frames to send in the current frame period.

The multiplexer design just described uses a finite buffer of size R/F to ensure that the multiplexer delay is $\leq 1/F$ seconds. An alternative implementation with a larger buffer is as follows. The server timestamps each of the frames it sends. When a frame reaches the front of the multiplexer buffer, the multiplexer checks to see if the delay of the frame is $\leq 1/F$. If the delay exceeds $1/F$, the multiplexer discards the frame. The multiplexer can also periodically check all the frames in the queue and purge those that have a delay exceeding $1/F$.

When a client requests a specific video, the network makes an admission control decision by deciding whether or not to grant the request. The admission control policy is to accept connections as long as the average link utilizations are $\leq 95\%$. The average link utilization is $util = F\sum_{j=1}^{J} x_{\text{avg}}(j)/R$, where $x_{\text{avg}}(j)$ is the average frame size in bits of the jth connection, which is calculated by averaging the corresponding sequence $(x_1, \ldots, x_N)$. If the network grants the request, a connection is established and the server immediately begins to transmit the connection's frames into the network. The frames arriving at the client are placed in the client's prefetch buffer. The video is displayed on the user's monitor as soon as a few frames have arrived at the client.

Under normal circumstances, every $1/F$ seconds the client removes a frame from its buffer, decompresses it, and displays it. If at one of these epochs there are no complete frames in its prefetch buffer, the client loses the current frame; the client will try to conceal the loss by, for instance, redisplaying the previous frame. At the subsequent epoch the client will attempt to display the next frame of the video.

4.3 Decentralized Prefetching Protocol

In this section we present our basic decentralized prefetching protocol that allows the server to determine how many frames to send in each frame period. This protocol strives to (1) make efficient use of the buffers at the client and (2) avoid bandwidth "hogging"

by a particular connection and thus give each connection a fair share of the bandwidth. The protocol attempts to allow each client to build up a reservoir of prefetched frames. Although our design allows for pause and temporal jumps, we will initially exclude these interactive features. We will also initially assume that the client buffers are infinite.

When discussing the server policy we again focus on a particular connection. We divide time into slots of length $1/F$. Let l denote the current slot; l is a local variable maintained by the server. In the course of the transmission of a video with N frames, l runs from 1 through N. *We do not assume any synchronization of time slots among the client–server pairs.*

Of central importance to our policy is the *send window*, denoted w_l, which limits the amount of traffic the connection can transmit in slot l. Specifically, the server is allowed to transmit $\lfloor w_l \rfloor$ frames during slot l. (We assume for simplicity that only complete frames are transmitted.) A new connection starts with a send window of $w_0 = 1$. The send window is increased by a small increment Δw, say 0.1, at the beginning of each slot, i.e. $w_l = w_{l-1} + \Delta w$. After computing the send window the server transmits $\lfloor w_l \rfloor$ frames; see Figure 4.2. Note that $w \geq 2$ allows for prefetching of future frames. To keep track of the number of prefetched frames in the client buffer, let p_l be the number of frames in the client buffer at the beginning of slot l. This variable is initialized to $p_1 = 0$. Let a_l denote the number of frames that are received and acknowledged by the client during slot l. Clearly, $0 \leq a_l \leq \lfloor w_l \rfloor$; a_l is equal to $\lfloor w_l \rfloor$ if all frames sent are received by the client. If frames are lost we have $a_l < \lfloor w_l \rfloor$. Figure 4.2 illustrates the timing of the prefetching protocol. We assume throughout that multiplexer buffer overflow is the only source of loss; the switch and interconnecting links are assumed lossless. We also assume that acknowledgements are never lost. Frame l is removed from the client buffer at the end of slot l if the client buffer contains one or more frames. The server keeps track of p_l through the following recursion:

$$p_{l+1} = [p_l + a_l - 1]^+. \tag{4.1}$$

Let s_l denote the number of bits received and acknowledged by the the client during slot l. Let b_l be the number of bits in the client buffer at the beginning of slot l; initially, $b_1 = 0$. With the given definitions, the server keeps track of b_l through the following

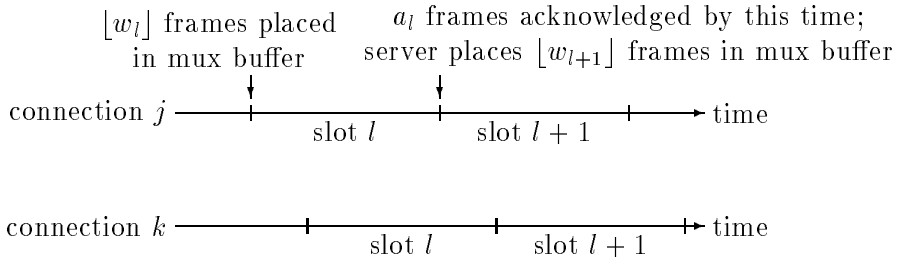


Figure 4.2: Timing diagram of prefetching policy. Server j places $\lfloor w_l \rfloor$ frames in the multiplexer buffer at the beginning of slot l. The acknowledgements for a_l frames arrive from the client by the end of slot l. The server processes the acknowledgments and puts $\lfloor w_{l+1} \rfloor$ frames in the multiplexer buffer at the beginning of slot $l+1$. There is no synchronization of slots between any distinct servers j and k.

recursion:

$$b_{l+1} = [b_l + s_l - x_l]^+. \tag{4.2}$$

If the server does not receive a positive acknowledgement for a frame sent at the beginning of the previous slot within one frame period, it assumes that the frame is lost. If a connection without any prefetched frames in the client buffer ($p_l = 0$) suffers loss, the client experiences starvation and may apply error concealment techniques to conceal the loss of video information. If the client has some prefetched frames in its buffer ($p_l > 0$), the server retransmits the lost frames. Whenever loss occurs, the server resets its send window to $w = 1$. The loss of frames is indicative of acute link overload and by reducing the send window we can throttle the server and thus alleviate the congestion. We refer to the send window policy described in this section as the *basic window policy*. It can be summarized as follows. A connection starts with a send window of one, that is, $w_0 = 1$. The window is increased by a small increment Δw (we use $\Delta w = 0.1$) at the beginning of each frame period. The number of frames a connection is allowed to send is limited by the integral part of the send window. If loss occurs, the window is reset to one.

4.4 Refinements of the Decentralized Prefetching Protocol

4.4.1 Client Buffer Constraint

We first introduce an important modification of the decentralized prefetching protocol. This modification limits the number of bits an ongoing connection may have in its client buffer. This important refinement is useful when the client has finite buffer capacity, B. This refinement works as follows. Suppose that the server is considering transmitting frame k. It transmits this frame in the current slot only if the send window allows the transmission of the frame and the *client buffer constraint*

$$b_l + x_k \leq B \tag{4.3}$$

is satisfied. Condition (4.3) ensures that the server does not overflow the client buffer.

4.4.2 Dynamic Send Window

We now introduce a refinement of the send window policy. The idea behind this refinement is to increase the send window by a large increment when the client buffer holds only a small reserve of prefetched frames and throttle the server when the client buffer contains a large reserve of prefetched frames. To this end, we compute the window increment as a function of the amount of prefetched data in the client buffer:

$$\Delta w_l = \Delta w_{\max}(1 - \frac{b_l}{B})^e, \quad \Delta w_{\max} > 0,\ e > 0. \tag{4.4}$$

Figure 4.3 illustrates this refined send window policy. When the client buffer is empty at the beginning of slot l, that is, when $b_l = 0$, the send window is incremented by $\Delta w_{\max}$. When the client buffer is full, that is, when $b_l = B$, the send window is not increased at all. We refer to this send window policy as the *dynamic window policy*. The dynamic window policy can be summarized as follows. At the beginning of slot l, the server computes Δw_l according to (4.4), calculates the new send window, $w_l = w_{l-1} + \Delta w_l$, and sends $\lfloor w_l \rfloor$ frames. As with the basic window policy, a new connection starts with a send window of $w_0 = 1$ and resets the window to $w_l = 1$ if the acknowledgments do not arrive by the end of slot l.

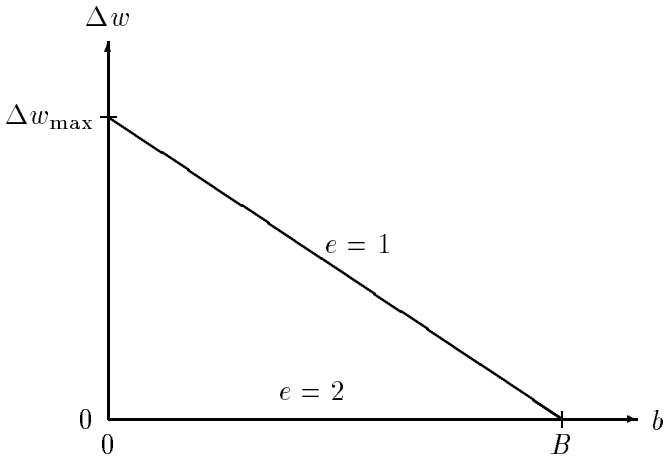


Figure 4.3: Illustration of the dynamic window policy.

The parameters $\Delta w_{\max}$ and e are used to tune the policy. We provide a detailed numerical study of the impact of these parameters on the performance of our decentralized prefetching protocol in Section 4.5. In Section 4.5 we also identify the ranges of the parameters that give good performance. We give here only a brief qualitative discussion of these parameters. A large $\Delta w_{\max}$ gives large increments Δw and thus allows the server to send more frames. The parameter Δw has to be large enough to allow for prefetching of future frames. If $\Delta w_{\max}$ is too large, however, a few connections can "swamp" the multiplexer and degrade the protocols' performance.

The parameter e can be set to give a connection with a nearly empty client buffer an increased chance of filling its client buffer. To see this, note that for $e = 1$, the window increment decreases linearly as the client buffer contents increase. For $e > 1$, connections with fairly large buffer contents are allowed substantially smaller increments (compared to when $e = 1$), while a connection with small client buffer contents has still a large window increment. This gives a connection with a small reserve of prefetched frames a better chance of filling its client buffer.

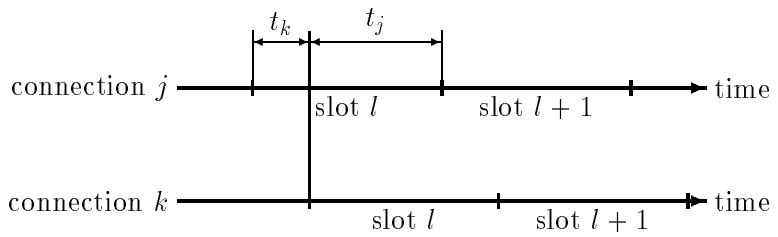


Figure 4.4: Phase alignment favoring connection j. If both connections fill the multiplexer buffer to capacity whenever they transmit, connection j can transmit Rt_j bits in a frame period, while connection k can transmit only Rt_k bits.

4.4.3 Randomized Transmission

In this subsection we introduce a refinement that helps to ensure fair bandwidth distribution among the ongoing connections. In the protocol described so far, the server transmits the first $\lfloor w_1 \rfloor$ frames of the video immediately after the request of the client has been processed. Subsequent transmissions are scheduled l/F seconds, $l = 1, \ldots, N-1$, after the initial transmission. The relative slot phases remain fixed for the entire duration of a video. To see how this can lead to unfair bandwidth distribution consider the phase alignment with $t_j \gg t_k$ depicted in Figure 4.4. Suppose connections j and k are the only connections in progress. Now consider a scenario where connection j fills the multiplexer buffer completely at the beginning of its slot l. Connection k is then able to fit Rt_k bits into the multiplexer buffer at the beginning of its slot l. When connection j is up for transmission again, at the beginning of its slot $l+1$, it can fit Rt_j bits into the multiplexer buffer. With the depicted phase alignment ($t_j \gg t_k$), connection k has clearly a disadvantage since it can transmit only Rt_k bits in a frame period as long as connection j keeps on filling the multiplexer buffer to capacity.

To avoid this unfair bandwidth distribution we introduce *randomized transmission*: The server transmits the first $\lfloor w_1 \rfloor$ frames of the video immediately after the request of the client has been processed. The server draws a random phase δ_l, $l = 1, \ldots, N-1$ from a uniform distribution over $[-1/2F, 1/2F]$ in each frame period. The subsequent transmissions are scheduled $l/F + \delta_l$ seconds, $l = 1, \ldots, N-1$ after the initial transmission. With this

transmission rule, the slot phases are constantly reshuffled. Unfair phase alignments can therefore not persist for extented periods of time.

Note that with randomized transmission, two consecutive transmissions can be spaced less than $1/F$ seconds apart. (In fact, two transmissions can be scheduled for the same time. This happens when the server draws the random phases $\delta_l = 1/2F$ and $\delta_{l+1} = -1/2F$. Note, however, that we are ignoring processing delays.) Thus, even with a maximal delay in the multiplexer of $1/F$ seconds and ignoring propagation and processing delays, the acknowledgements may not arrive before the next transmission.

We propose two solutions for this problem. The first solution relies on the multiplexer sending back an error message to the server when a frame does not fit into the multiplexer buffer. We note that the Source Quench Error Message defined in the Internet Control Message Protocol (ICMP) [70, p.160] may be used for this purpose. The server assumes that a frame is successfully received by the client if the multiplexer does not send an error message. The client is not required to send acknowledgments when this approach is used. We refer to this approach as *multiplexer feedback*.

An alternative solution is to randomly spread the transmissions not over the entire frame period but instead over half the frame period by drawing the random phases δ_l from a uniform distribution over $[-1/2F, 0]$. Setting the multiplexer buffer to $R/2F$ ensures that the acknowledgements from the client are received before the next transmission is scheduled. We refer to this approach as the *client feedback* approach. We note that by spreading out the transmissions over a smaller interval and reducing the multiplexer buffer client feedback may degrade the performance of the decentralized prefetching protocol. We provide a detailed numerical study of the impact of client feedback on the protocols performance in Section 4.5.

4.5 Experimental Results

In this section we present the results of a simulation study of the decentralized prefetching protocol. The study is based on MPEG 1 encodings of the four movies in Table 3.1. As in Chapter 3 we assume in our numerical work that the video frames are transported in packets consisting of 512 bytes of payload and 40 bytes of overhead. We fix the link rate

at $R = 45$ Mbps; the corresponding multiplexer buffer holds 234,375 bytes ($= R/F$). We define the link utilization as the sum of the mean bit rates of all ongoing connections divided by R. In our experiments we use a mix of the the four movies that achieves 95% link utilization. Specifically, we use 55 lambs connections, 17 bond connections, 37 terminator connections, and 23 mr.bean connections.

In each realization of our simulation, we generate a random starting frame $\theta(j)$ for each of the J ongoing connections. The value $\theta(j)$ is the frame that is removed from the jth client buffer at the end of slot 1. The $\theta(j)$'s are independent and uniformly distributed over $[1, N]$. All connections start with empty client buffers at the beginning of slot 1. When the Nth frame of a video is removed from a client buffer, we assume that the corresponding user immediately requests to see the entire movie again. Thus, there are always J connections in progress. For each replication of the simulation we also draw random (non–synchronized) slot phases $t(j)$ for each of the J connections. The $t(j)$'s are independent and are drawn from a uniform distribution over $[0, 1/F]$. The $t(j)$'s determine the relative starting times of the slots for the J connections. Note that the frames of connection j scheduled for transmission in slot l are placed in the multiplexer buffer at the beginning of the slot (see Figure 4.2), that is, server j puts its traffic into the queue at instants $t(j) + (l-1)/F,\ l = 1, \ldots, N$ $(t(j) + (l-1)/F + \delta_{l-1},\ l = 1, \ldots, N$ with randomized transmission). In all our simulations we assume that all clients have the same buffering capacity, B. We allow a warm–up time of 40,000 frame periods for each replication before counting frame periods with starvation. We run each simulation until the 90% confidence interval is less than 10% of the estimated loss probability. We define the loss probability as the long run fraction of frame periods for which at least one client experiences starvation.

In Figures 4.5, 4.6 and 4.7 we show typical plots of the client buffer contents, b_l, the window increment, Δw_l, and the send window, w_l, versus slot time, l, for four arbitrarily chosen connections. Figure 4.8 gives the number of frames that are successfully placed in the multiplexer buffer by each of the four connections. For this simulation run we have set the client buffer capacity to $B = 1$ MBit. We employ the dynamic window policy without randomized transmission with $\Delta w_{\max} = 5$ and $e = 2$. The plots illustrate how the client buffer contents control the window increment. The left part of Figure 4.5 shows that

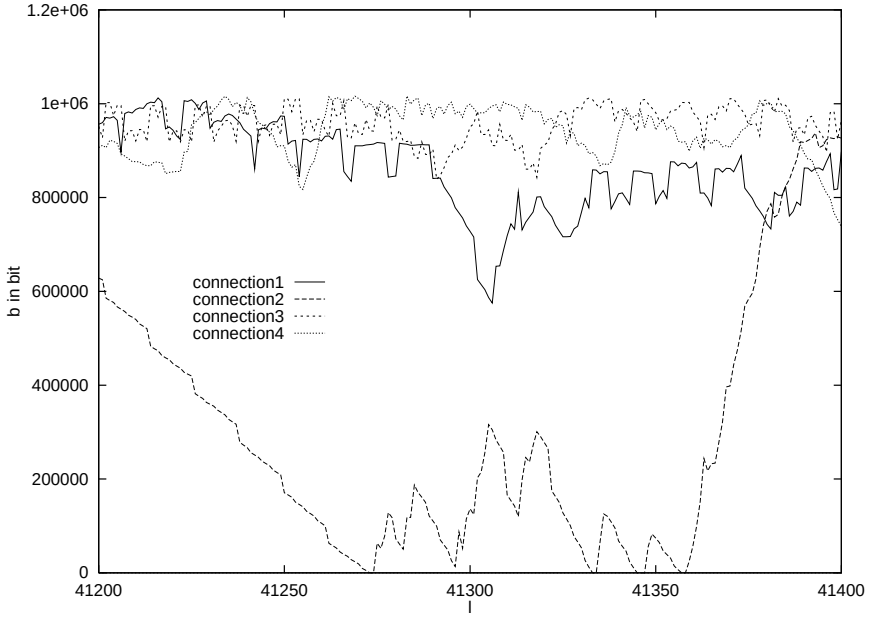


Figure 4.5: Client buffer contents versus slot time of four arbitrarily chosen connections with 1 MBit of client buffer.

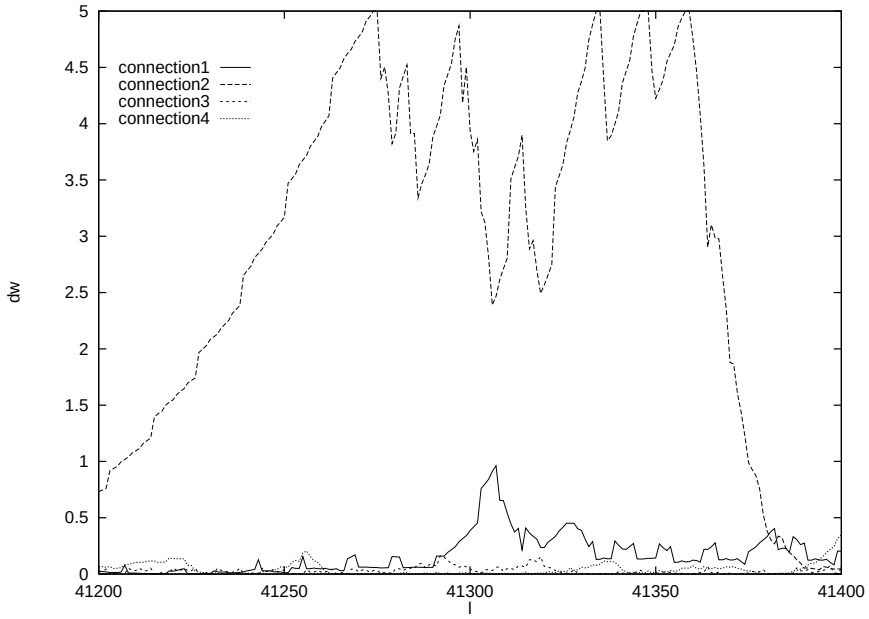


Figure 4.6: Window increment Δw versus slot time; Δw is computed according to the dynamic window policy with $\Delta w_{\max} = 5$ and $e = 2$, that is, $\Delta w_l = 5(1 - b_l/1\text{MBit})^2$.

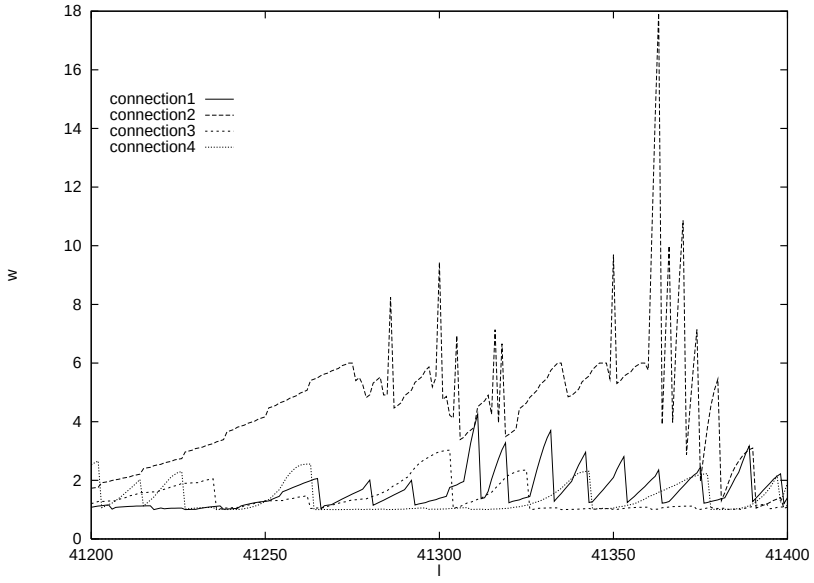


Figure 4.7: Send window versus slot time.

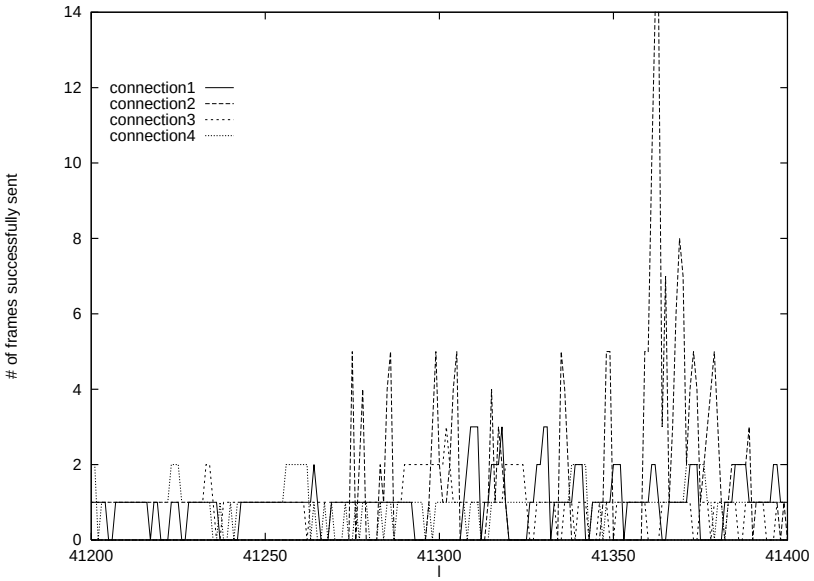


Figure 4.8: Number of frames successfully placed in the multiplexer buffer versus slot time.

the client buffer of connection 2 is drained. This is due to a high action scene in the video. We see from Figure 4.6 that the window increment of connection 2 increases as the client buffer is depleted. By the end of time slot 41,274 the client buffer is empty and the window increment has risen to $\Delta w_{\max} = 5$. This allows connection 2 to transmit very aggressively. We observe from Figure 4.7 that the send window becomes as large as 17.9 in time slot 41,363. Figure 4.8 shows that the first 14 frames of the 17 frames sent in time slot 41,363 can be accommodated by the multiplexer buffer. The remaining 3 frames are lost. The send window is therefore reset to $w_{41,363} = 1$ at the end of slot 41,363. At the beginning of slot 41,364 the new send window is computed as $w_{41,364} = w_{41,363} + \Delta w_{41,364} = 1 + 2.9 = 3.9$, allowing connection 2 to send 3 frames. We see from Figure 4.8 that all 3 frames fit into the multiplexer buffer and the send window increases to $w_{41,365} = 3.9 + 3.1 = 7.0$ in slot 41,365. The right part of Figure 4.5 shows that the large send windows enable connection 2 to fill its client buffer again. By time slot 41380 the client buffer is filled to 90% of its capacity.

We now espouse the problem of finding the values of the dynamic window parameters $\Delta w_{\max}$ and e that optimize the performance of the decentralized prefetching protocol. Toward this end we first focus on the impact of the parameter $\Delta w_{\max}$. In Figure 4.9 we arbitrarily set $e = 2$ and plot the loss probability as a function of $\Delta w_{\max}$ for the dynamic window policy without and with randomized transmission. For this plot we have set the client buffer capacity to $B = 128$ KByte. (We are currently working on similar plots for other client buffer capacities.) The figure seems to indicate that the dynamic window policy works well for a wide range of $\Delta w_{\max}$ values. Any $\Delta w_{\max}$ between 2 and 11 seems to work well for the dynamic window policy without randomized transmission while any value between 2 and 8 gives good performance when randomized transmission is employed. Throughout the rest of this chapter we use the $\Delta w_{\max}$ values that attain the minima in Figure 4.9; $\Delta w_{\max} = 5$ for the dynamic window policy without and with randomized transmission.

We also observe from Figure 4.9 that the loss probability grows larger for small and very large $\Delta w_{\max}$ values. The server can not prefetch a sufficient number of frames when $\Delta w_{\max}$ is too small (<2) and starvation at the client is therefore more likely. When $\Delta w_{\max}$ is too large (>11 without randomized transmission, > 8 with randomized transmission) a few

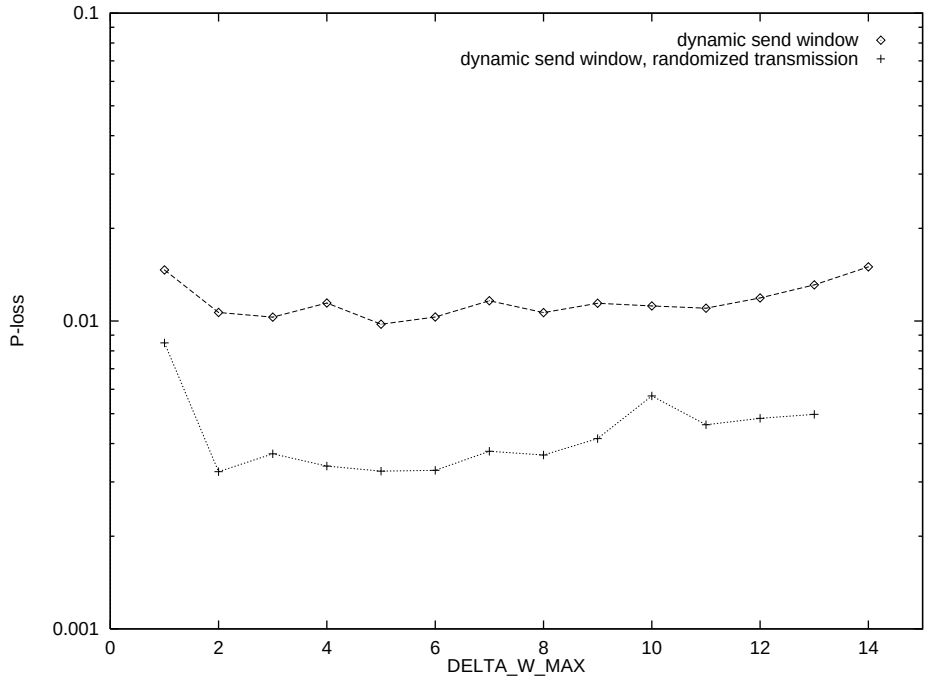


Figure 4.9: Loss probability as a function of $\Delta w_{\max}$ for the dynamic window policy without and with randomized transmission; $e = 2$ fixed.

connections can "swamp" the multiplexer buffer preventing the other ongoing connections from maintaining a full client buffer.

We next study the impact of the parameter e on the performance of the dynamic window policy. In Figure 4.10 we plot the loss probability as a function of the parameter e for the dynamic window policy without randomized transmission. (The plot for randomized transmission is omitted as it is very similar and leads to exactly the same conclusions.) For this experiment we use $\Delta w_{\max} = 5$ and 128 KByte of client buffer. The parameter e appears to have significant impact on the performance of the dynamic window policy. The minimum of the loss probability, which is attained for $e = 6$, is about half the loss probability for $e = 3$ or $e = 4$. We use $e = 6$ throughout the remainder of this chapter.

Connections with empty client buffers are allotted significantly larger window increments than connections with full buffers as e gets larger (see Figure 4.3). This gives connections with empty buffers an increased chance of fitting their frames into the multiplexer buffer as connections with full buffers are throttled. This effect is reflected in Figure 4.10. When e is small connections with empty client buffers are allotted large

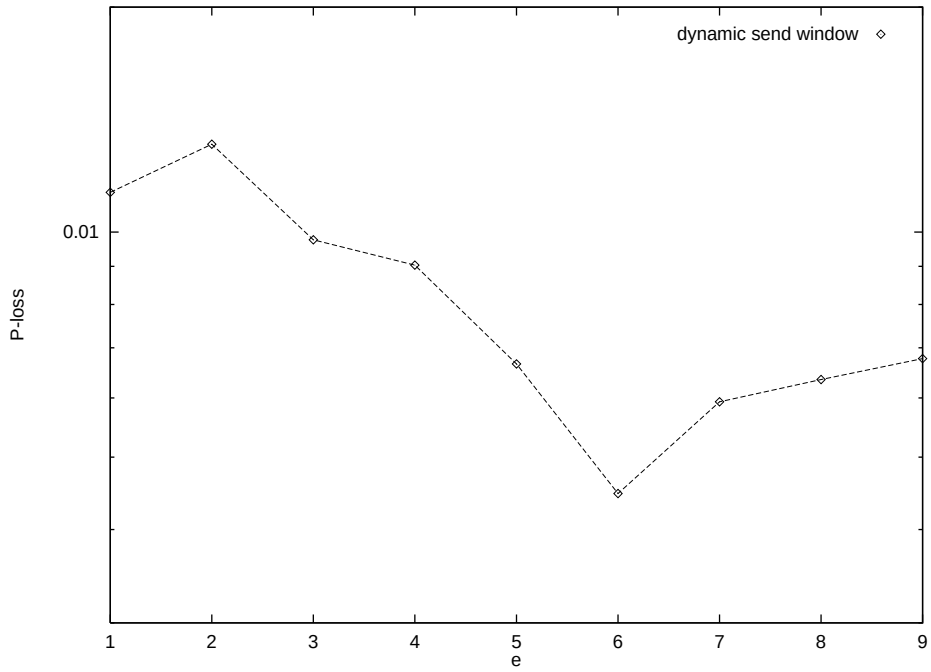


Figure 4.10: Loss probability as a function of e for the dynamic window policy without randomized transmission; $\Delta w_{\max} = 5$ fixed.

window increments, but the window increments of connections with full buffers are also relatively large. The frames of connections with empty buffers therefore have to compete with frames of connections with full buffers for bandwidth. It is hence quite likely that frames of connections with empty buffers are lost at the multiplexer, leading to starvation at the client. For large e, the window increments of connections with full buffers are minute compared to the window increments of connections with empty buffers. The connections with full buffers are thus throttled and the connections with empty buffers have an increased chance of getting their frames through to the client. When e is too large (≥ 7) the dynamic window policy gives connections with a few prefetched frames an extremely small window increment and the send window hardly grows to 2 or beyond. This prevents the clients from prefetching more frames. The clients are thus able to build up only a small reserve of prefetched frames and starvation is therefore more likely.

Figure 4.11 shows the performance of our basic decentralized prefetching protocol, and its various refinements. We plot the loss probability as a function of the client buffer size for 95% link utilization. For the basic window policy we use a fixed window increment

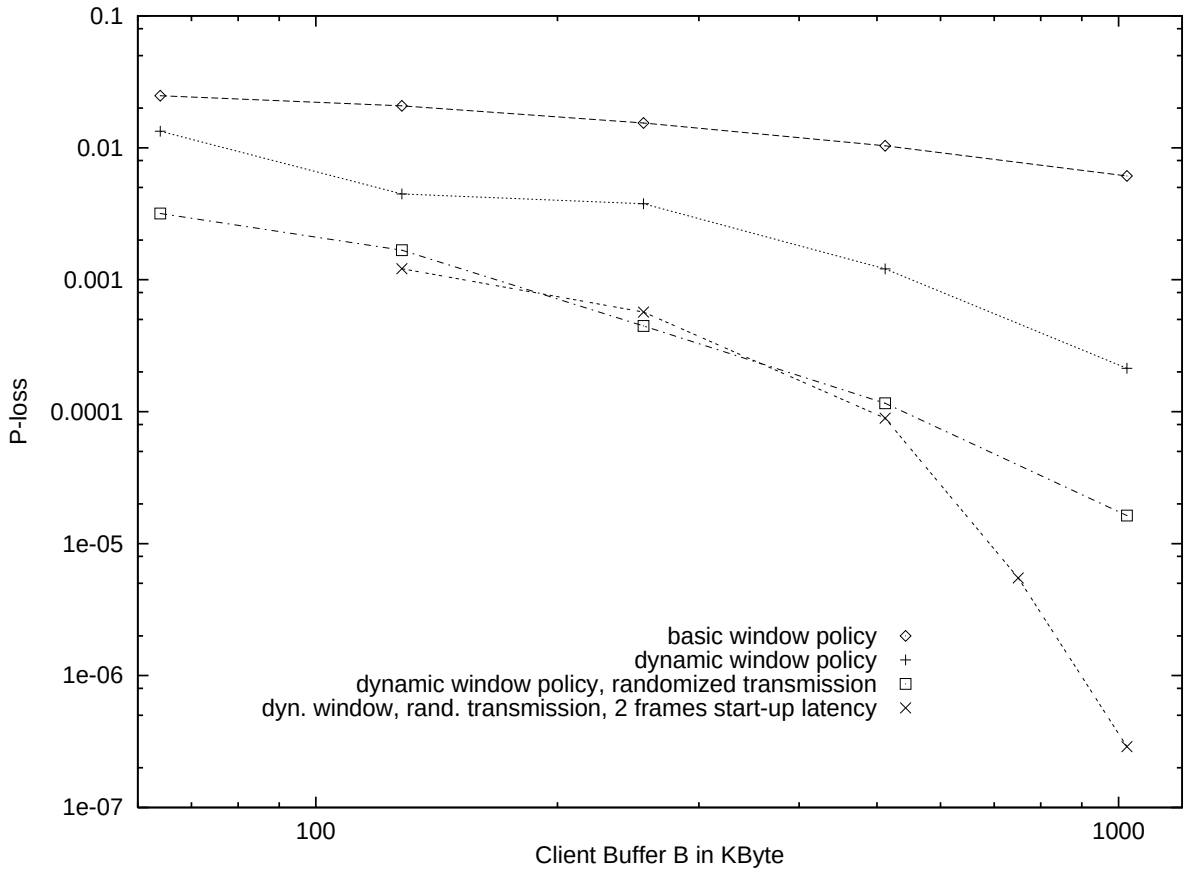


Figure 4.11: Loss probability as a function of client buffer size for the basic decentralized prefetching protocol and its refinements.

of $\Delta w = 0.1$. The parameters of the dynamic window policy are set to $\Delta w_{\max} = 5$ and $e = 6$. The figure shows that the basic window policy has unacceptably high losses. The loss probability is about 8×10^{-3} for 1 MByte of client buffer. We also see that the dynamic window policy brings significant improvement over the basic window policy. The loss probability for the dynamic window policy is almost one order of magnitude smaller. Adding randomized transmission further reduces the loss probability significantly. The loss probability for the dynamic window policy with randomized transmission for 1 MByte of client buffer is about 1.5×10^{-5}. We employ multiplexer feedback here.

Our experiments showed that the loss probability for decentralized prefetching with the dynamic window policy and randomized transmission does not drop below 10^{-5} even for very large buffers ($>$ 1Mbyte). We observed that with very large client buffers, losses occur almost exclusively right at the beginning of the movie when the client has no prefetched

frames. We are therefore motivated to introduce a short start–up latency allowing the client to prefetch for a couple of frame periods without removing and displaying frames. We found that a very short start–up latency of just 2 frame periods brings dramatic improvements in performance. With a start–up latency of 2 frame periods the client prefetches during the first and second time slot without removing frames; the first frame is removed and displayed at the end of the third slot. The loss probability with 2 frames start–up latency is 2×10^{-7} for 1 MByte of client buffer; almost two orders of magnitude lower that without start–up latency. For client buffers smaller that half a MByte, however, the start–up latency does not reduce the loss probability.

This can be explained by the two typical loss scenarios that we observed in our experiments. One loss scenario is due to high action scenes in the movies. With moderately sized client buffers ($\leq$ 500 KByte), a high action scene which requires large frames is likely to drain the client buffer completely and lead to subsequent losses. For large client buffers ($>$ 500 KByte), it is highly unlikely that a high action scene drains the buffer completely. Losses due to a high action scene are therefore extremely rare.

The other loss scenario occurs at the start of a movie. When a movie starts, loss occurs when due to an unfair phase alignment none of the first $\lfloor w_1 \rfloor$ frames of the movie get through to the client. Since we are drawing a new random phase in each slot, it is unlikely that this unfair phase alignment persists for the next slot. By allowing a new connection a short start–up latency of only 2 frames we can therefore avoid most of the initial losses.

In Figure 4.12 we study the two feedback schemes described in Section 4.4.3. Recall from Section 4.4.3 that in order to provide the server with timely feedback when randomized transmission is employed we may either use multiplexer feedback or client feedback. With multiplexer feedback the multiplexer alerts the server whenever one of its packets does not fit into the multiplexer buffer. Multiplexer feedback allows the server to spread out the transmissions randomly over the entire frame period. Client feedback relies on the client sending an acknowledgement for each received packet to the server. In order to ensure that the acknowledgements arrive before the next transmission is scheduled the transmissions can be spread out over only half of the frame period and the multiplexer buffer size is restricted to $R/2F$ (the delay in the multiplexer is then at most half a frame period). The figure gives the loss probability as a function of the client buffer size for 95%

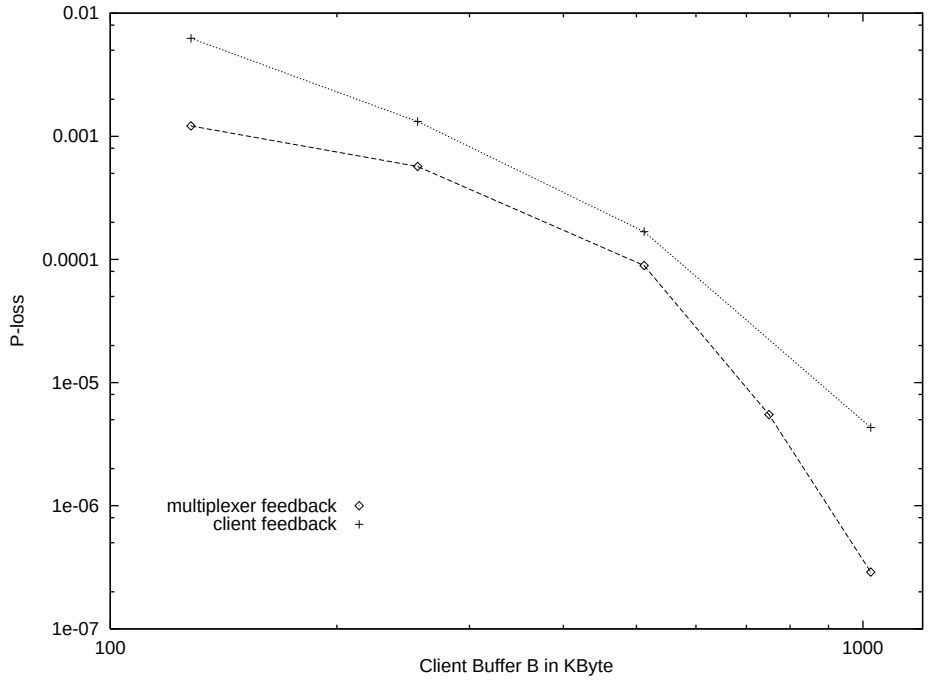


Figure 4.12: Loss probability as a function of client buffer size for randomized transmission with multiplexer feedback and client feedback.

link utilization. We employ the dynamic window policy with randomized transmission and allow for a 2 frame start–up latency. The curve for multiplexer feedback already appeared in Figure 4.11. The figure shows that client feedback performs slightly worse that multiplexer feedback. The loss probability for client feedback is about half an order of magnitude larger. This can be explained by noting that the transmissions are spread out over a smaller interval and the multiplexer buffer is smaller for client feedback. It is therefore more likely that a frame does not fit into the multiplexer buffer. This in turn leads to an increased probability of starvation at the client.

In Figure 4.13 we compare our decentralized prefetching protocols with Join–the–Shortest–Queue (JSQ) Prefetching [58] and Optimal Smoothing [66, 79, 61]. The plot gives the loss probability as a function of the client buffer size for 95% link utilization. The optimal smoothing curves are obtained by applying the optimal smoothing algorithm [66, 79, 61] to the traces used for the simulation of the prefetch policy. We then compute the loss probability for statistically multiplexing the smoothed traces on a bufferless 45 Mbps link with the Large Deviation approximation [57, 79]. We do this for two versions

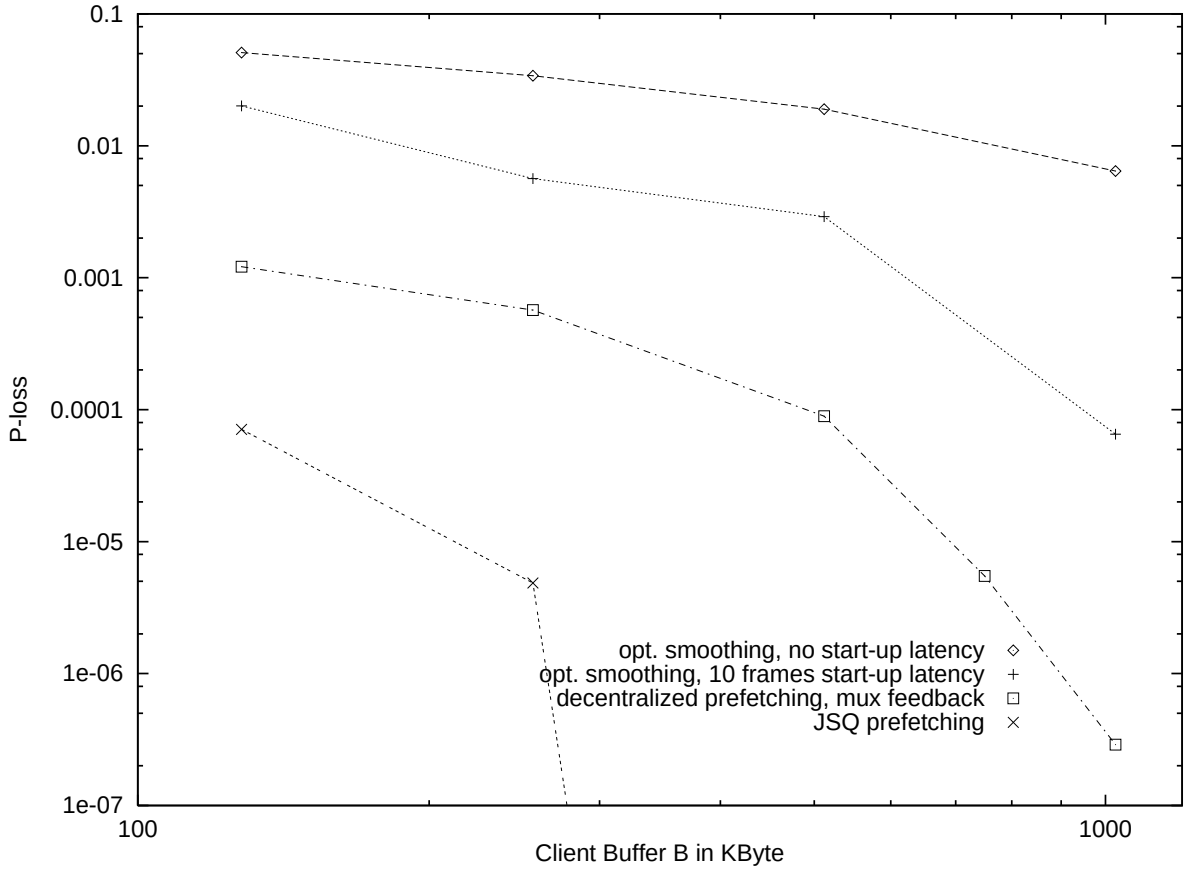


Figure 4.13: Loss probability as a function of client buffer size for optimal smoothing, decentralized prefetching and JSQ prefetching.

of optimal smoothing: no initiation delay and a 10 frame initiation delay [67, 79, 11]. The decentralized prefetching results are for the dynamic window policy with randomized transmission, multiplexer feedback and 2 frames start–up latency. The JSQ prefetching results are from [58]. Decentralized prefetching clearly outperforms optimal smoothing, both without and with start–up latency. The loss probability for decentralized prefetching is over one order of magnitude smaller than the loss probability for optimal smoothing with start–up latency. The gap widens to over two orders of magnitude for 1 MByte of client buffer. The performance of the decentralized prefetching protocol, however, does not come close to the remarkable JSQ performance. In the next section we study the gain in performance that can be achieved by adding priorities to the decentralized prefetching protocol.

4.6 Prefetching with Priorities

We now attempt to improve the performance of decentralized prefetching by having the sever mark certain frames as priority frames. Frames are sent as low priority when the client has one or more prefetched frames in its buffer, that is, when $p_l > 0$. Frames are sent as high priority when there are no prefetched frames in the client buffer, that is, when $p_l = 0$.

We assume in this study that the multiplexer implements a non–preemptive priority policy which works as follows. A high priority frame entering the queue never interrupts the transmission of the frame currently in service, irrespective of its priority. The new high priority frame is placed behind any high priority frames already in the queue and ahead of all low priority frames. If the multiplexer is implemented with the finite buffer of size R/F, low priority frames are removed if necessary from the queue in order to accommodate high priority frames. A high priority frame, however, never pushes another high priority frame out of the queue. Low priority frames are always added at the end of the queue, provided there is space. The multiplexer with finite buffer size R/F furthermore timestamps every low priority frame entering the queue. It periodically checks the low priority frames in the queue and removes frames that have been in the queue for more than $1/F$ seconds. This prevents low priority frames from getting stuck in the back of the queue while high priority frames are served for extended periods of time.

With this priority policy, a high priority frame is lost if and only if the queue is filled up to capacity with other high priority frames. Note that the client suffers starvation when a high priority frame is lost. This is because a frame is sent as high priority if and only if the client has no prefetched frames in its buffer ($p_l = 0$) and needs the high priority frame by the end of the slot in order to ensure continuous playback. When a low priority frame is dropped in order to accommodate a high priority frame the server does not receive an acknowledgement, times out and retransmits the frame. Starvation at the client occurs only if the dropped frame has to be retransmitted as high priority (because the client has exhausted its reserve of prefetched frames) and the high priority frame does not fit into the queue.

The preceding discussion applies if the multiplexer is implemented with a finite buffer of

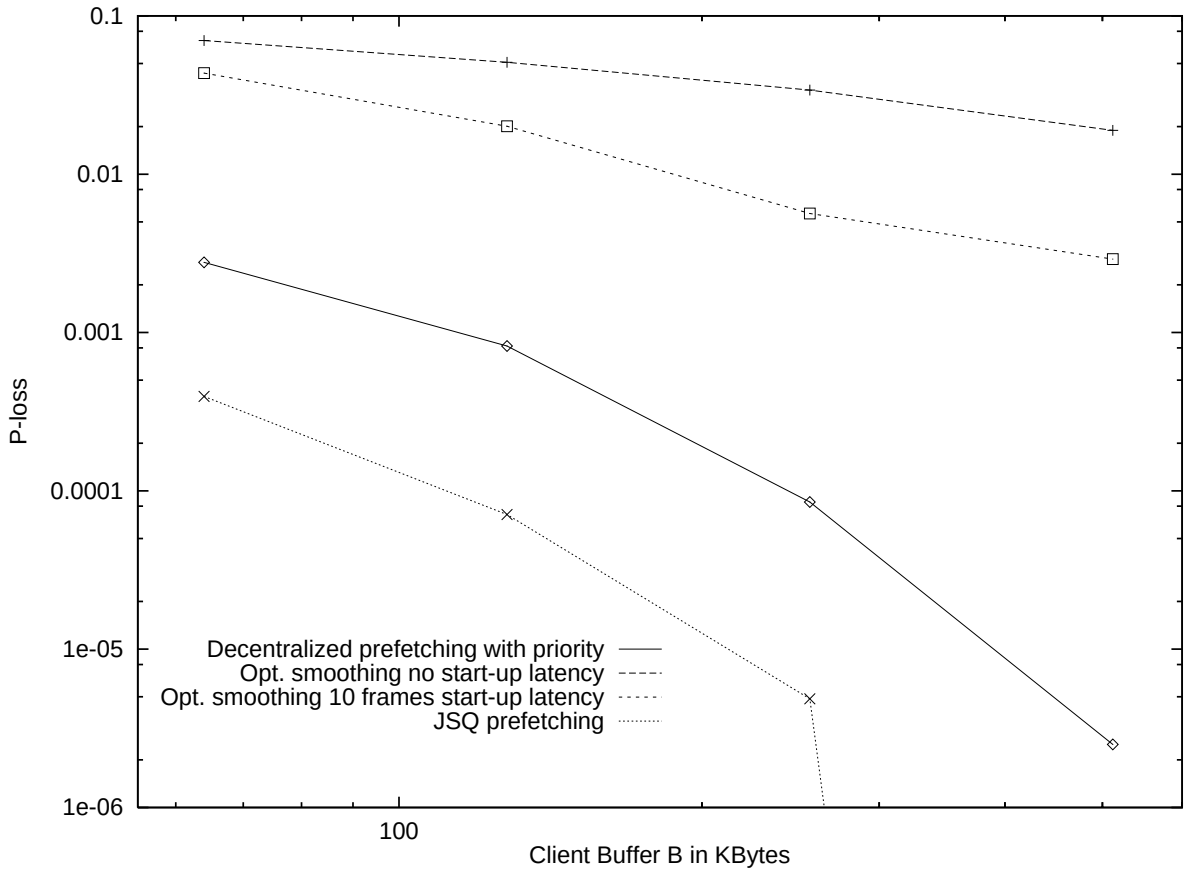


Figure 4.14: Loss probability as a function of client buffer size for 95% link utilization.

capacity R/F. If the multiplexer is implemented with a larger buffer, all frames entering the queue are timestamped, irrespective of their priority. The multiplexer periodically checks the frames in the queue and drops those with a delay exceeding $1/F$ seconds.

4.6.1 Experimental Results

In Figure 4.14 we plot the loss probability as a function of the client buffer size for 95% link utilization. The solid line gives the performance of the decentralized prefetching protocol with priorities; we apply the basic send window policy here. (We are currently working on combining priorities with the dynamic window policy and randomized transmission.) We observe that the decentralized prefetching protocol with priorities clearly outperforms optimal smoothing. For 512 KByte client buffer the loss probability for the decentralized prefetching protocol with priorities is about 3 orders of magnitude smaller than the loss

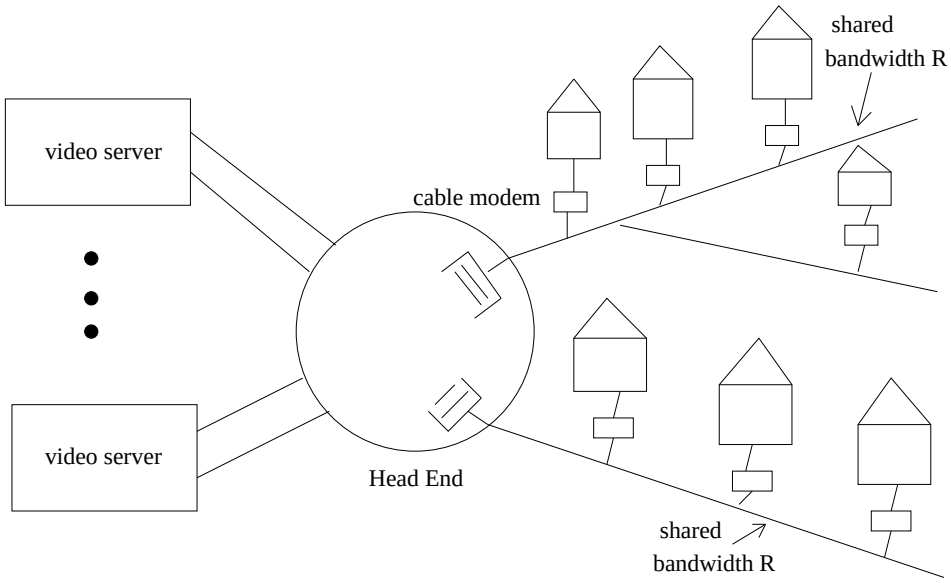


Figure 4.15: Decentralized VoD architecture for cable residential access

.

probability for statistically multiplexing the optimally smoothed traces. JSQ prefetching still performs better than decentralized prefetching, but the addition of priorities has narrowed the gap to about one order of magnitude.

It can be argued that the average rate in bits/sec of the traces driving the simulation (see Table 3.1) is lower than what we would expect for digital compressed video (e.g., MPEG–2 video). We expect VoD systems in the future to provide MPEG–2 encoded video with an order of magnitude larger average rates. We also expect that the server transmission rate and client buffer grow proportionally. In this scaling, the number of videos that can be multiplexed will be approximately constant, and only 5–10 MByte of client buffer is required to give negligible loss.

4.7 Decentralized Prefetching and Residential Broadband Access

In this section we discuss how the decentralized prefetching protocol for VoD proposed in this chapter ties into the cable modem technology. Figure 4.15 shows a possible decentralized VoD architecture with cable. Multiple video servers attach directly to the cable

headend as do multiple cable trunks. Homes are attached to cable trunks via cable modems. The video servers could be owned by one video service provider or by multiple competing service providers (all of whom run their applications over our decentralized prefetching protocol). The request for a video is relayed from the viewers home to the headend via the upstream channels. The headend, acting as an Ethernet switch, ATM switch, or router, forwards the request to the appropriate video server. The video server immediately starts transmitting the video frames. The switch in the headend forwards the frames to the appropriate output queue. All the videos requested by viewers connected to the same cable trunk are multiplexed onto the shared channel of capacity, say R bps. Our decentralized prefetching protocol allows for the efficient use of the valuable trunk bandwidth, R. We achieve transmission with negligible losses and thus constant high video quality for average trunk bandwidth utilizations of 95%.

4.8 Conclusion

Prerecorded video has two special properties: (1) for each video, the traffic in each video frame is known before the video session begins; (2) while the video is being played, some of the video can be prefetched into the client memory. In this chapter we have shown how these two properties can be exploited to achieve high performance when servers transmit VBR video across a packet–switched network to clients. Our simulation results indicate that our decentralized prefetching protocols give good performance, better than that of other decentralized prefetching protocols in the existing literature. Even though the server has to decide on a transmission schedule without any direct knowledge of the state of the other ongoing connections, our decentralized prefetching protocol with priorities does almost as well as JSQ prefetching.

Chapter 5

Buffered Multiplexers with Regulated Traffic

5.1 Overview

In this chapter we consider a finite–buffer packet multiplexer to which traffic arrives from several independent sources. For the multiplexer to provide quality of service (QoS) guarantees, such as limits on packet loss probabilities, it must have some knowledge about the traffic characteristics of the sources. Because the reliability of statistical models of traffic is questionable for many source types, in recent years there have been several studies on the performance of packet–switched nodes that multiplex *regulated traffic*, e.g., traffic which conforms to known constraints imposed by leaky buckets. These studies suppose that the traffic from the sources is *adversarial* to the extent permitted by the regulators [6] [3] [75] [32] [23] [14] [42] [36] [51] [63] [50]. Some of these studies assume that the multiplexer provides deterministic QoS guarantees (e.g., no packet loss) whereas others assume that multiplexer provides the less stringent probabilistic QoS guarantees (e.g., a limit on packet loss probability).

In a recent paper, LoPresti *et al.* [42] examine a packet–switched node with regulated traffic. Motivated by earlier work of Elwalid *et al.* [14], LoPresti *et al.* consider both deterministic QoS guarantees and probabilistic QoS guarantees. They assume that each source is regulated by a *simple regulator*, namely, a regulator that consists of a peak–rate

controller in series with a leaky bucket. For deterministic QoS, LoPresti *et al.* show that if the multiplexer has sufficient link bandwidth and buffer capacity to provide lossless multiplexing, then the multiplexer's buffer and bandwidth can be allocated among the sources so that the resulting segregated systems are lossless. For probabilistic QoS, they develop a new approach to estimate the loss probability. Specifically, they transform the two–resource (bandwidth and buffer) allocation problem into two independent single–resource allocation problems; they then analyze these simpler, independent resource problems, taking on–off periodic sources for their adversarial sources.

Although the simple regulator is a popular policing mechanism within several standards bodies, it has been observed that it can often be a poor characterization of a source's worst–case traffic. A tighter and more powerful characterization is given by a more general regulator consisting of a cascade of multiple leaky buckets [75] [23]. For example, when the sources are VBR video sources, it is often possible to admit significantly more connections by replacing the simple regulator with cascaded–leaky–bucket regulators [75]. It is therefore desirable to extend the important work of [42] and [14] to the case of more general regulators.

In this chapter we reexamine the model of [42] in the context of *generalized regulators*, which are even more general than cascaded leaky buckets. We first reexamine the lossless multiplexer of LoPresti *et al.*, and extend their lossless results to generalized regulators. Using elementary tools from calculus, we show that if the original multiplexer is lossless, then it is possible to allocate bandwidth and buffer to the sources so that the resulting segregated systems are also lossless. We determine the optimal resource allocations and show that the buffer–bandwidth tradeoff curve is convex for generalized regulators. We also show that the segregation result does not necessarily hold for the delay–based QoS metric, even when the regulators are the simple regulators.

We then examine the multiplexer for probabilistic loss guarantees. We use our results for lossless multiplexing to estimate the loss probability of the multiplexer. As in [42], our estimate involves the following three steps: (i) choose a point on the buffer–bandwidth tradeoff curve and transform the original system into two independent resource systems; (ii) use adversarial sources for the two independent resources to obtain a bound on the loss probabilities for the transformed system; (iii) minimize the bound by searching over all

points on the buffer–bandwidth tradeoff curve. Our principle contribution for probabilistic loss guarantees is an explicit characterization of the adversarial source for the transformed problem in Step (ii). Importantly, *the most adversarial source is not a periodic on–off source for the transformed problem consisting of two independent resources.* In fact, even in the case of simple regulated sources as studied in [42], the most adversarial source is not a periodic on–off source. Thus, in addition to generalizing the theory in [42] to the case of general regulated sources, we provide the true adversarial source for the case of the simple regulator. We also provide an algorithm to calculate the estimate of loss probability, assuming the truly adversarial sources.

We mention here that in [14] the original multiplexor problem is transformed into a bufferless multiplexer problem, and then the loss probability is bounded with the Chernoff bound. In this case, the worst–case adversarial sources are indeed on–off periodic sources. But when the original problem is transformed into a problem consisting of two independent resources, one bufferless resource and one buffered resource, the worst–case sources are no longer on–off periodic sources, even for simple regulators.

This chapter is organized as follows. In Section 5.2 we define the model and the generalized regulators. In Sectin 5.3 we address lossless multiplexing. In Section 5.4 we address lossy multiplexing. In Section 5.5 we provide numerical results for lossy multiplexing of simple regulators, i.e., regulators consisting of a peak rate controller in series with a leaky bucket.

5.2 Regulated Traffic

We consider a link of rate C which is preceded by a finite buffer. Let J be the number of sources that send traffic to the buffer, and let $j = 1, \ldots, J$ index the sources. Each source j has an associated regulator function, denoted by $\mathcal{E}_j(t)$, $t \geq 0$. The regulator function constrains the amount of traffic that the jth source can send over an time interval of length t to $\mathcal{E}_j(t)$. More explicitly, if $A_j(t)$ is the amount of traffic that the jth source sends to the buffer over the interval $[0, t]$, then $A_j(\cdot)$ is required to satisfy

$$A_j(t+\tau) - A_j(\tau) \leq \mathcal{E}_j(t) \text{ for all } \tau \geq 0, \quad t \geq 0.$$

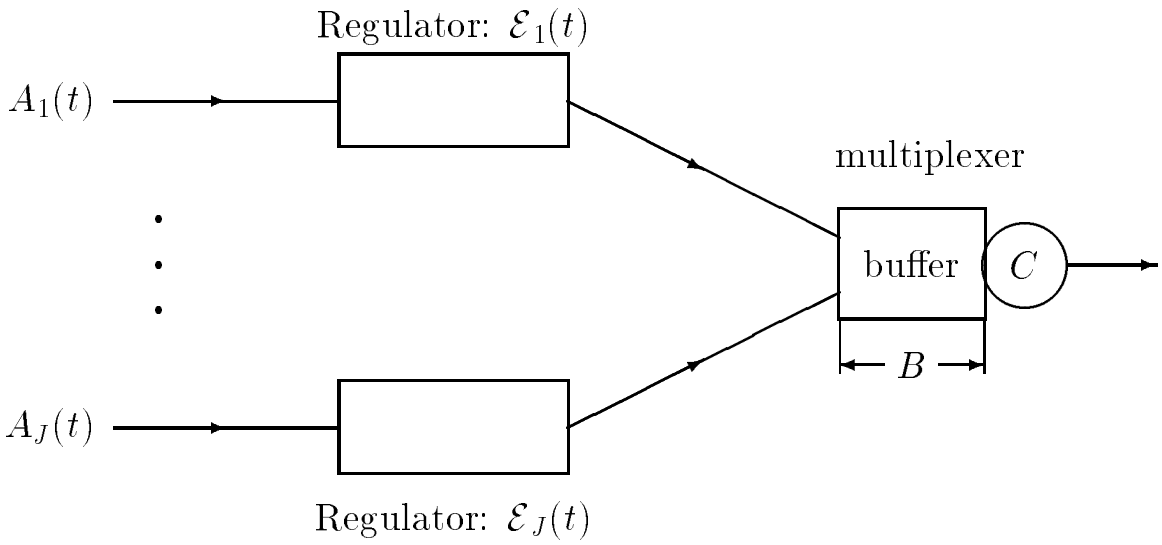


Figure 5.1: Link of capacity C, buffer of capacity B, and J regulators.

Figure 5.2 illustrates a multiplexer consisting of a link of rate C, a buffer of capacity B, and J sources with regulated traffic functions, $\mathcal{E}_j(t)$, $j = 1, \ldots, J$.

A popular regulator is the simple regulator, which consists of a peak–rate controller in series with a leaky bucket; for the simple regulator, the regulator function takes the following form:

$$\mathcal{E}_j(t) = \min\{\rho_j^1 t \ , \sigma_j^2 + \rho_j^2 t\}.$$

For a given source type, the bound on the traffic provided by the simple regulator may be loose and lead to overly conservative admission control decisions. For many source types (e.g., for VBR video [75]), it is possible to get a tighter bound on the traffic and dramatically increase the admission region. In particular, regulator functions of the form

$$\mathcal{E}_j(t) = \min\{\rho_j^1 t, \sigma_j^2 + \rho_j^2 t, \ldots, \sigma_j^{L_j} + \rho_j^{L_j} t\}$$

are easily implemented with cascaded leaky buckets and can lead to improved admission regions (see [75]).

In this chapter we shall consider extremely general regulator functions, which include as special cases the forms mentioned above. To avoid certain trivialities, however, we shall

always assume that $\mathcal{E}_j(0) = 0$, $\mathcal{E}_j(t)$ is non–decreasing in t, and that $\mathcal{E}_j(t)$ is subadditive in t (i.e., $\mathcal{E}_j(t_1 + t_2) \leq \mathcal{E}_j(t_1) + \mathcal{E}_j(t_2)$ for all t_1 and t_2). Also, unless explicitly mentioned otherwise, we shall assume that each $\mathcal{E}_j(t)$ is concave in t. Let

$$\mathcal{E}(t) = \sum_{j=1}^{J} \mathcal{E}_j(t)$$

be the aggregate regulator function. Due to the concavity of the $\mathcal{E}_j(t)$'s, the aggregate regulator function $\mathcal{E}(t)$ is also concave.

Before preceding with our analysis of the lossless systems, it is convenient at this point to introduce some notation and state a few technical facts. Let $\mathcal{E}_j^+(t)$ denote the right derivative for $\mathcal{E}_j(t)$ and $\mathcal{E}_j^-(t)$ denote the left derivative for $\mathcal{E}_j(t)$. Let $\mathcal{E}_j'(t)$ denote the derivative of $\mathcal{E}_j(t)$ whenever the derivative exists at t. Similarly define $\mathcal{E}^+(t)$, $\mathcal{E}^-(t)$ and $\mathcal{E}'(t)$. We will make use of the following fact: If $\mathcal{E}(t)$ is differentiable at t^*, then all of the $\mathcal{E}_j(t)$'s are differentiable at t^* (due to the concavity of the $\mathcal{E}_j(t)$'s).

5.3 Guaranteed Lossless Service and Optimal Segregation

It is well known [6] that the amount of traffic in the buffer does not exceed $B_{\min}$, where

$$B_{\min} = \max_{t \geq 0} \{\mathcal{E}(t) - Ct\} \ . \tag{5.1}$$

(To avoid trivialities we assume that the maximum is attained in (5.1).) Furthermore, due to subadditivity, it is possible to define traffic functions $A_j(t)$, $j = 1, \ldots, J$, such that the buffer contents will attain $B_{\min}$. Thus the minimum buffer size that will guarantee lossless operation is $B_{\min}$. Throughout the remainder of this section we assume that the multiplexer is lossless, i.e., we assume that the multiplexer buffer B satisfies $B \geq B_{\min}$.

It will be useful to write (5.1) in a more convenient form. If $\mathcal{E}(t)$ is differentiable then from (5.1) we have

$$B_{\min} = \mathcal{E}(t_{\max}) - Ct_{\max} \ , \tag{5.2}$$

where $t_{\max}$ is any solution to $\mathcal{E}'(t) = C$. More generally, there exists a $t_{\max}$ such that

$$\mathcal{E}^+(t_{\max}) \leq C \leq \mathcal{E}^-(t_{\max}) \ , \tag{5.3}$$

and any $t_{\max}$ which satisfies (5.3) also satisfies (5.2). Throughout the remainder of this section, fix a $t_{\max}$ that satisfies (5.3) (and therefore (5.2) as well).

We now address the following question: Is it possible to allocate bandwidth and buffer to the J sources so that each of the resulting segregated systems is also lossless? We shall see that the answer to this question is yes, but depends critically on the concavity of the $\mathcal{E}_j(t)$'s.

To address this issue, consider a new system which consists of a link of rate c preceded by a finite buffer. Suppose only the traffic from source j is sent to this system. The minimum buffer size that will ensure lossless operation is

$$B_{\min}(j,c) = \max_{t\geq 0}\{\mathcal{E}_j(t) - ct\} \ . \tag{5.4}$$

We say that a collection of J positive numbers $c_1,\ldots,c_J$ is a *bandwidth allocation* if $c_1 + \cdots + c_J = C$. For a given bandwidth allocation, we create J segregated systems, with the jth segregated system having link rate c_j and receiving traffic only from source j.

Theorem 1 *1. For all allocations $B_{\min} \leq \sum_{j=1}^{J} B_{\min}(j,c_j)$.*

2. If one or more of the $\mathcal{E}_j(t)$'s is not concave then we may have $B_{\min} < \sum_{j=1}^{J} B_{\min}(j,c_j)$ for all allocations $c_1,\ldots,c_J$.

3. If each $\mathcal{E}_j(t)$ is concave then $B_{\min} = \sum_{j=1}^{J} B_{\min}(j,c_j^*)$ where $c_j^* = \mathcal{E}_j'(t_{max})$ if $\mathcal{E}(t)$ is differentiable at $t = t_{\max}$ and where

$$c_j^* = \mathcal{E}_j^+(t_{\max}) + \alpha[\mathcal{E}_j^-(t_{\max}) - \mathcal{E}_j^+(t_{\max})]$$

with

$$\alpha = \frac{C - \mathcal{E}^+(t_{\max})}{\mathcal{E}^-(t_{\max}) - \mathcal{E}^+(t_{\max})}$$

if $\mathcal{E}(t)$ is non–differentiable at $t = t_{\max}$.

Proof. The proof of the first claim follows from (5.1) and (5.4):

$$\begin{aligned} B_{\min} &= \max_{t\geq 0}\{\mathcal{E}(t) - Ct\} = \max_{t\geq 0}\sum_{j=1}^{J}\{\mathcal{E}_j(t) - c_j t\} \\ &\leq \sum_{j=1}^{J}\max_{t\geq 0}\{\mathcal{E}_j(t) - c_j t\} = \sum_{j=1}^{J} B_{\min}(j,c_j) \end{aligned}$$

For the second claim, we offer the following counterexample with $J = 2$, $C = 1$. The envelope function for the first source is:

$$\mathcal{E}_1(t) = \begin{cases} t & \text{if } 0 \le t \le 1 \\ 1 & \text{if } 1 \le t \le 3 \\ 1 + (t-3) & \text{if } 3 \le t \le 4 \\ 2 & \text{if } t \ge 4 \ . \end{cases}$$

The envelope function for the second source is:

$$\mathcal{E}_2(t) = \begin{cases} 2t & \text{if } 0 \le t \le 2 \\ 4 & \text{if } t \ge 2 \ . \end{cases}$$

It is easily seen that $B_{\min} = 3$ whereas $B_{\min}(1, c_1) + B_{\min}(2, c_2) \ge 10/3$ for all allocations. Note that both $\mathcal{E}_1(t)$ and $\mathcal{E}_2(t)$ are non–decreasing and sub–additive. However, $\mathcal{E}_1(t)$ is not concave.

For the third claim, we first show that $c_1^*, \ldots, c_J^*$ is a feasible allocation. Suppose that $\mathcal{E}(t)$ is differentiable at $t_{\max}$. Due to the concavity assumption, this implies that each of the $\mathcal{E}_j(t)$'s is differentiable at $t_{\max}$. Thus

$$\begin{aligned} \sum_{j=1}^{J} c_j^* &= \sum_{j=1}^{J} \mathcal{E}_j'(t_{\max}) \\ &= \mathcal{E}'(t_{\max}) = C \ . \end{aligned}$$

If $\mathcal{E}(t)$ is not differentiable at $t = t_{\max}$, then it is easy to show directly from the definition of the c_j^*'s that $c_1^* + \cdots + c_J^* = C$. It remains to show that $B_{\min} = \sum_{j=1}^{J} B_{\min}(j, c_j^*)$. For a fixed transmission rate c, the concavity of the $\mathcal{E}_j(t)$'s and (5.4) imply

$$B_{\min}(j, c) = \mathcal{E}_j(t^*) - ct^* \ ,$$

where t^* is any t that satisfies

$$\mathcal{E}_j^+(t) \le c \le \mathcal{E}_j^-(t) \ . \tag{5.5}$$

By the definition of c_j^*,

$$\mathcal{E}_j^+(t_{\max}) \le c_j^* \le \mathcal{E}_j^-(t_{\max}) \ .$$

Thus, $t_{\max}$ is a t that satisfies (5.5) for $c = c_j^*$. Therefore,

$$B_{\min}(j, c_j^*) = \mathcal{E}_j(t_{\max}) - c_j^* t_{\max} \ ,$$

which in turn implies

$$\begin{aligned}\sum_{j=1}^{J} B_{\min}(j, c_j^*) &= \sum_{j=1}^{J} [\mathcal{E}_j(t_{\max}) - c_j^* t_{\max}] \\ &= \mathcal{E}(t_{\max}) - C t_{\max} = B_{\min} \ .\end{aligned}$$

■

From Theorem 1 we know that it is possible to allocate bandwidth and buffer so that the resulting segregated systems are lossless, provided that the regulator functions are concave. This result generalizes a result in [42], in which all regulators were assumed to be simple regulators. This result also provides a motivation for the approach we take in Section 5.4 when we study probabilistic QoS.

Theorem 1 also gives fairly explicit formulas for these optimal allocations. In the following subsection we outline an efficient algorithm for calculating the allocations.

5.3.1 Algorithm to Calculate Allocations

In this subsection suppose that each of the regulator functions takes the form of cascaded leaky buckets:

$$\mathcal{E}_j(t) = \min\{\rho_j^1 t, \sigma_j^2 + \rho_j^2 t, \ldots, \sigma_j^{L_j} + \rho_j^{L_j} t\} \ .$$

Without loss of generality we may assume that

$$0 = \sigma_j^1 < \sigma_j^2 < \cdots < \sigma_j^{L_j} \tag{5.6}$$

and

$$\rho_j^1 > \rho_j^2 > \cdots > \rho_j^{L_j} \ . \tag{5.7}$$

Let

$$T_j^l = \frac{\sigma_j^{l+1} - \sigma_j^l}{\rho_j^l - \rho_j^{l+1}}, \quad l = 1, 2, \ldots, L_j - 1 \ .$$

In order to avoid trivialities we assume that

$$T_j^1 < T_j^2 < \cdots < T_j^{L_j - 1} \ . \tag{5.8}$$

With these assumptions, $T_j^1 < T_j^2 < \cdots < T_j^{L_j-1}$ are the breakpoints of $\mathcal{E}_j(t)$.

Here is an efficient algorithm for determining the optimal allocations $c_1^*, \ldots, c_J^*$ defined in Theorem 1. First sort T_j^l, $l = 1, \ldots, L_j$, $j = 1, \ldots, J$, in increasing order. Number

them as $T_1, T_2, \ldots, T_L$. These points are the break points of $\mathcal{E}(t)$. Let k be the maximum l such that $\mathcal{E}^-(T_l) \le C$. Note that to calculate $\mathcal{E}^-(T_l)$ it suffices to calculate $\mathcal{E}_j^-(T_l)$ for $j = 1, \ldots, J$; and to calculate $\mathcal{E}_j^-(T_l)$, we can determine the l_j such that $T_j^{l_j} \le T_l < T_j^{l_j+1}$ and then set $\mathcal{E}_j^-(T_l) = \rho_j^{l_j+1}$ if $T_j^{l_j} < T_l$ and set $\mathcal{E}_j^-(T_l) = \rho_j^{l_j}$ if $T_j^{l_j} = T_l$.

The $t_{\max}$ in Theorem 1 is T_k. Once having determined k, find k_j such that $T_j^{k_j} \le T_k < T_j^{k_j+1}$ and set $c_j^* = \rho_j^{k_j+1}$ if $T_j^{k_j} < T_k$ or set $c_j^* = \rho_j^{k_j} + \alpha(\rho_j^{k_j+1} - \rho_j^{k_j})$ if $T_j^{k_j} = T_k$, where α is defined in Theorem 1 and can also be determined directly from the ρ_j^l's.

5.3.2 The Buffer–Bandwidth Tradeoff Curve

For a given link rate C let $B_{\min}(C)$ be the maximum buffer contents defined by (5.1).The function $B_{\min}(C)$ is called the buffer–bandwidth tradeoff curve. For a probabilistic analysis in the next section, it will be useful to understand the behavior of the buffer–bandwidth tradeoff curve. To this end, for each fixed C let $t(C)$ be a value of $t_{\max}$ that satisfies (5.3). It is easily seen that $t(C)$ is non–increasing in C.

Theorem 2 *$B_{\min}(C)$ is non–increasing and convex in C.*

Proof. We first show that $B_{\min}(C)$ is non–increasing. Let $h > 0$. From (5.2) we have

$$B_{\min}(C) - B_{\min}(C+h) = \mathcal{E}(t(C)) - \mathcal{E}(t(C+h)) + t(C+h)(C+h) - t(C)C \; . \qquad (5.9)$$

From the concavity of $\mathcal{E}(t)$ we have

$$\mathcal{E}^-(t(C)) \le \frac{\mathcal{E}(t(C)) - \mathcal{E}(t(C+h))}{t(C) - t(C+h)} \; . \qquad (5.10)$$

From (5.3) we have

$$C \le \mathcal{E}^-(t(C)) \; . \qquad (5.11)$$

Combining (5.9)–(5.11) gives

$$\begin{aligned} B_{\min}(C) - B_{\min}(C+h) &\ge \mathcal{E}^-(t(C))[t(C) - t(C+h)] + t(C+h)(C+h) - t(C)C \\ &\ge C[t(C) - t(C+h)] + t(C+h)(C+h) - t(C)C \\ &= t(C+h)h \ge 0, \end{aligned}$$

which proves the first statement.

For the convexity of $B_{\min}(C)$, let $C_1 \leq C_2$ and let $h > 0$. We must show

$$B_{\min}(C_2 + h) - B_{\min}(C_2) \geq B_{\min}(C_1 + h) - B_{\min}(C_1) \ . \tag{5.12}$$

By (5.2) it is equivalent to show

$$\begin{aligned} &\mathcal{E}(t(C_2 + h)) - \mathcal{E}(t(C_2)) + \mathcal{E}(t(C_1)) - \mathcal{E}(t(C_1 + h)) \\ &\geq t(C_2 + h)(C_2 + h) - t(C_2)C_2 - t(C_1 + h)(C_1 + h) + t(C_1)C_1 \ . \end{aligned} \tag{5.13}$$

Using the arguments in the proof of monotonicity, we have

$$\frac{\mathcal{E}(t(C_2)) - \mathcal{E}(t(C_2 + h))}{t(C_2) - t(C_2 + h)} \leq C_2 + h \tag{5.14}$$

and

$$\frac{\mathcal{E}(t(C_1)) - \mathcal{E}(t(C_1 + h))}{t(C_1) - t(C_1 + h)} \geq C_1 \ . \tag{5.15}$$

Combining (5.13), (5.14) and (5.15) we obtain (5.12). ■

From Theorem 2 we know that $B_{\min}(C)$ is a decreasing convex function of C. If each of the regulator functions $\mathcal{E}_j(t)$ is piecewise linear, then it is easily shown that $B_{\min}(C)$ is a decreasing convex piecewise–linear function. Using the arguments in the proof of Theorem 2, it is straightforward to show that the optimal allocation c_j^* for the jth segregated system is increasing in C and that the buffer requirement for the jth segregated system, $B_{\min}(j, c_j^*)$, is decreasing in C.

5.3.3 Delay Metric

In Subsection 5.3.1 we showed how to allocate bandwidth so that, for lossless operation, the collective buffer requirements of the segregated system is equal to the buffer requirement of the multiplexed system. In other words, for the *buffer metric* we can find a bandwidth allocation such that the segregated system performs as well as the multiplexed system. In this subsection we briefly consider a natural *delay metric*. We show that it is not generally true that the segregated system performs as well as the multiplexed system for the delay metric.

For the multiplexed system the maximum delay is $d := B_{\min}/C$. For the jth segregated system with bandwidth c_j the maximum delay is $d(j, c_j) := B_{\min}(j, c_j)/c_j$. For a given

allocation, we define the maximum delay of the collective segregated system to be the maximum of the maximum delays of the individual segregated systems, that is,

$$d_{seg} := \max_{1 \le j \le J} d(j, c_j) \ .$$

The following theorem draws comparisons between the maximum delay of the multiplexed system, d, and the maximum delay of the collective segregated system, d_{seg}.

Theorem 3 *1. For all allocations* $d \le \max_{1 \le j \le J} d(j, c_j)$.

2. There exist concave $\mathcal{E}_j(t)$*'s such that* $d < \max_{1 \le j \le J} d(j, c_j)$ *for all allocations.*

3. If $\mathcal{E}_1(t) = \cdots = \mathcal{E}_J(t)$ *(homogeneous regulator functions), then* $d = \max_{1 \le j \le J} d(j, c/J)$.

Proof. From Theorem 1 we have

$$B_{\min} \le \sum_{j=1}^{J} B_{\min}(j, c_j) \ .$$

Dividing both sides of the above by $C = c_1 + \ldots + c_J$ and using the inequality

$$\frac{x_1 + \cdots + x_J}{y_1 + \cdots + y_J} \le \max_{1 \le j \le J} \frac{x_j}{y_j}$$

we obtain

$$\begin{aligned} d &\le \frac{\sum_{j=1}^{J} B_{\min}(j, c_j)}{\sum_{j=1}^{J} c_j} \le \max_{1 \le j \le J} \{ \frac{B_{\min}(j, c_j)}{c_j} \} \\ &= \max_{1 \le j \le J} d(j, c_j), \end{aligned}$$

which establishes the first claim.

For the second claim we offer the following example: $C = 1$, $J = 2$, $\mathcal{E}_1(t) = 10$ for all $t \ge 0$ and

$$\mathcal{E}_2(t) = \begin{cases} 2t & \text{if } 0 \le t \le 5 \\ 10 & \text{if } t \ge 5 \ . \end{cases}$$

From (5.1) we have $d = 15$, $d(1, c_1) = 10/c_1$, and $d(2, c_2) = 10/c_2 - 5$. It is easily seen that for all allocations $\max(10/c_1, 10/c_2 - 5) > 15$.

The third statement follows directly from (5.1) and the definitions of d and $d(j, C/J)$.

■

For the remainder of the chapter we will use the original buffer metric.

5.4 Statistical Multiplexing with Small Loss Probabilities

For VBR sources the admission region can typically be made significantly larger by allowing loss to occur with minute probabilities, e.g., loss probabilities on the order of 10^{-6}. In this section we use our results of Section 5.3 to derive the worst–case loss probabilities for the multiplexer with regulated traffic.

We consider the same system defined in Section 5.2: The multiplexer consists of a link of rate C which is preceded by a finite buffer. There are J sources and the jth source has an associated regulator function, denoted by $\mathcal{E}_j(t)$, $t \geq 0$. In this section we suppose that the system resources are not sufficient to provide guaranteed lossless service. In other words, we assume $B < B_{\min}(C)$, so that there exists arrival processes which meet the regulator constraints but which cause the buffer to overflow. Let P_{loss} denote the expected fraction of time during which the buffer overflows. Our goal is to determine a bound for P_{loss} that holds for *all* combinations of arrival processes which meet the regulator constraints. To this end, we follow the methodology in [42] (which in turn is inspired by the paper [14]).

Let $a_j(t)$ be the rate at which source j transmits traffic at time t. We view $\{a_j(t),\ t \geq 0\}$ as a stochastic process. Our goal is to find independent rate processes $\{a_j(t),\ t \geq 0\}$, $j = 1, \ldots, J$, which maximize the loss probability over the class of all rate processes that meet the regulator constraints. To simplify the analysis, however, we only consider rate processes of the form

$$a_j(t) = b_j(t + \theta_j),$$

where $b_j(t)$ is a deterministic *periodic* function with some period T_j, and θ_j is a random variable, uniformly distributed over $[0, T_j]$. We assume that the phases $\theta_1, \ldots, \theta_J$ are independent, which implies that the rate processes $\{a_j(t),\ t \geq 0\}$, $j = 1, \ldots, J$, are also independent. We refer to $b_j(t)$ as a *source–j rate function.*

We say that a source–j rate function $b_j(t)$ is *feasible* if

$$\int_{\tau}^{t+\tau} b_j(s)ds \leq \mathcal{E}_j(t) \text{ for all } \tau \geq 0, \quad t \geq 0. \tag{5.16}$$

Note that for a given rate function $b_j(t)$ and phase θ_j the amount of source–j traffic sent to the multiplexer over the interval $[0, t]$ is

$$A_j(t) = \int_0^t b_j(s + \theta_j)ds.$$

Thus the regulator constraint

$$A_j(t+\tau) - A_j(\tau) \leq \mathcal{E}_j(t) \text{ for all } \tau \geq 0, \quad t \geq 0$$

is satisfied if and only if $b_j(t)$ is a feasible rate function.

As in [42], our derivation of a bound for P_{loss} involves the following three steps: (i) choose a point on the buffer–bandwidth tradeoff curve and transform the original system into two independent resource systems; (ii) use adversarial rate functions for the two independent resources to obtain a bound on the loss probabilities for the transformed system; (iii) minimize the bound by searching over all points on the buffer–bandwidth tradeoff curve. LoPresti *et al.* use an on–off rate function for their worst case rate function. Our approach differs from that of [42] in two respects. First, we allow for generalized regulators as opposed to simple regulators. Second, we derive the true adversarial rate functions, and employ these true adversarial rate functions in the bound for P_{loss} for both simple and generalized regulators.

5.4.1 The Virtual Segregated System

Fix a point (C_ν, B_ν) on the buffer–bandwidth tradeoff curve, and consider a lossless multiplexer with total amount of bandwidth C_ν and buffer space B_ν. Because the system resource pair (B, C) lies below the buffer–bandwidth tradeoff curve, we must have either $C_\nu > C$ or $B_\nu > B$ or both. For this lossless system we use Theorem 1 to allocate bandwidths $c_1^\nu, \ldots, c_J^\nu$ from C_ν and buffers $b_1^\nu, \ldots, b_J^\nu$ from B_ν such that each of the corresponding J segregated systems is lossless. This collection of J segregated systems is called the virtual segregated system [42].

For each $j = 1, \ldots, J$, fix a feasible rate function $b_j(t)$. Each rate function generates a stochastic arrival process

$$A_j(t) = \int_0^t b_j(s + \theta_j) ds.$$

For this arrival process, let U_j be a random variable that corresponds to the steady–state utilization of the jth segregated system; similarly, let V_j be the random variable that corresponds to the steady–state buffer contents of the jth segregated system. Because the θ_j's are independent across the sources, $U_1, \ldots, U_J$ are independent of each other and $V_1, \ldots, V_J$ are independent of each other.

For these fixed rate functions it can be argued [42] that

$$P_{\text{loss}} \leq P_\nu(\sum_{j=1}^{J} U_j > C) + P_\nu(\sum_{j=1}^{J} V_j > B)\,. \tag{5.17}$$

(The argument in [42] is for a simple regulator. It can be easily extended to our generalized regulators.) The equation (5.17) is the starting point of our own analysis.

Using the Chernoff bound we get

$$P_{\text{loss}} \leq \min_{\alpha \geq 0} \left\{ \frac{\prod_{j=1}^{J} M_{U_j}^{\nu}(\alpha)}{e^{\alpha C}} \right\} + \min_{\alpha \geq 0} \left\{ \frac{\prod_{j=1}^{J} M_{V_j}^{\nu}(\alpha)}{e^{\alpha B}} \right\} \tag{5.18}$$

where $M_{U_j}^{\nu}(\alpha)$ and $M_{V_j}^{\nu}(\alpha)$ are the moment genrating functions of U_j and V_j repectively, i.e., $M_{U_j}^{\nu}(\alpha) = E[e^{\alpha U_j}]$ and $M_{V_j}^{\nu}(\alpha) = E[e^{\alpha V_j}]$. Since (5.18) is valid for all points (C_ν, B_ν) on the buffer-bandwidth tradeoff curve, we have

$$P_{\text{loss}} \leq \min_{(C_\nu, B_\nu)} \left[\min_{\alpha \geq 0} \left\{ \frac{\prod_{j=1}^{J} M_{U_j}^{\nu}(\alpha)}{e^{\alpha C}} \right\} + \min_{\alpha \geq 0} \left\{ \frac{\prod_{j=1}^{J} M_{V_j}^{\nu}(\alpha)}{e^{\alpha B}} \right\} \right] \tag{5.19}$$

We emphasize that the right–hand side of (5.19) depends on the fixed feasible rate functions. In order to give a bound that holds for *all* feasible rate functions we need to maximize the right–hand side of (5.19) over the set of all feasible rate functions. To this end, we introduce the notion of a *source–j adversarial rate function.*

Corresponding to each choice of (ν, α), we say that a source–j rate function is *adversarial* if (i) it is feasible, and (ii) it has the largest value of $M_{U_j}^{\nu}(\alpha)$ and $M_{V_j}^{\nu}(\alpha)$ among all feasible source–j rate functions. Now suppose that we can find the source–j adversarial rate functions for each choice of (ν, α); let U_j^*, V_j^*, $j = 1, \ldots, J$, be the corresponding steady–state random variables. We then have the following bound on P_{loss}:

$$P_{\text{loss}} \leq \min_{(C_\nu, B_\nu)} \left[\min_{\alpha \geq 0} \left\{ \frac{\prod_{j=1}^{J} M_{U_j^*}^{\nu}(\alpha)}{e^{\alpha C}} \right\} + \min_{\alpha \geq 0} \left\{ \frac{\prod_{j=1}^{J} M_{V_j^*}^{\nu}(\alpha)}{e^{\alpha B}} \right\} \right] \tag{5.20}$$

Note that by using $M_{U_j^*}^{\nu}(\alpha)$ and $M_{V_j^*}^{\nu}(\alpha)$, which corresponds to the source–j adversarial rate function, we have obtained in (5.20) a bound on P_{loss} that is valid for all combinations of feasible arrival functions. We now proceed to characterize the adversarial rate functions.

5.4.2 Adversarial Sources

Throughout this subsection fix a ν, α and j. We now focus on determining a feasible rate function which maximizes both $M_{U_j}^{\nu}(\alpha)$ and $M_{V_j}^{\nu}(\alpha)$ over the set of feasible rate functions.

We assume that the regulator functions have the form

$$\mathcal{E}_j(t) = \min\{\rho_j^1 t, \sigma_j^2 + \rho_j^2 t, \ldots, \sigma_j^{L_j} + \rho_j^{L_j} t\} \ .$$

Note that $\mathcal{E}_j(t)$ is non–decreasing, concave, piecewise–linear and sub–additive. (The analysis that follows can easily be extended to the case of more general $\mathcal{E}_j(t)$ which are non–decreasing, concave and sub–additive.) Without loss of generality we also assume that (5.6), (5.7) and (5.8) hold. Note that the manner in which the allocations $(c_1^\nu, \ldots, c_J^\nu)$ are chosen (see Theorem 1) ensures that $\rho_j^{L_j} \leq c_j^\nu \leq \rho_j^1$ for all $j = 1, 2, \ldots, J$.

For a given feasible rate function $b_j(t)$ with period T_j, the arrival rate at time t is $a_j(t) = b_j(t+\theta_j)$ where θ_j is uniformly distributed over $[0, T_j]$. Corresponding to this $a_j(t)$ arrival rate process, let $v_j(t)$ be the buffer contents and $u_j(t)$ be the link utilization at time t. Note that $v_j(t)$ and $u_j(t)$ are periodic with period T_j. Also the steady–state random variables corresponding to $v_j(t)$ and $u_j(t)$ have distributions

$$P(V_j \leq x) = \frac{1}{T_j} \int_0^{T_j} 1(v_j(s) \leq x) ds$$

and

$$P(U_j \leq x) = \frac{1}{T_j} \int_0^{T_j} 1(u_j(s) \leq x) ds \ .$$

Note that these distributions do not depend on the phase θ_j and are completely determined by the rate function $b_j(t)$ and the link rate c_j^ν.

Throughout the remainder of this subsection we treat the case $c_j^\nu > \rho_j^{L_j}$. In the following subsection we deal with the simpler case $c_j^\nu = \rho_j^{L_j}$. Let

$$\delta_j = \max\{t > 0 : \frac{\mathcal{E}_j(t)}{t} \geq c_j^\nu\} \ . \tag{5.21}$$

Note that since $\rho_j^{L_j} < c_j^\nu \leq \rho_j^1$ and since $\mathcal{E}_j(\cdot)$ is an increasing concave function, δ_j is a uniquely defined, finite and strictly positive number. We now define an important class of rate functions. Let T_{off} be such that $0 < T_{\text{off}} \leq \delta_j$ and let

$$T_j = \frac{\mathcal{E}_j(T_{\text{off}})}{\rho_j^{L_j}}.$$

Now consider a rate function $b_j(t)$ with period T_j defined as follows:

$$b_j(t) = \begin{cases} \mathcal{E}_j^+(t) & 0 \leq t \leq T_{\text{off}} \\ 0 & T_{\text{off}} \leq t \leq T_j \end{cases}$$

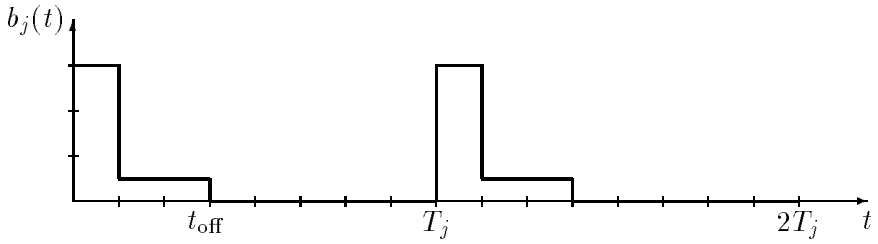


Figure 5.2: Example of a rate function in Set S_j when $t_{\text{off}} = 3$ and $\mathcal{E}_j(t) = \min\{3t, 2.5 + 0.5t\}$.

Such a rate function is pictured in Figure 5.2.

This rate function is completely characterized by the parameter T_{off}. Note that the average arrival rate for this rate function is simply $\rho_j^{L_j}$. Let S_j be the collection of all rate functions of this form. Each rate function in S_j is identified through its T_{off} parameter.

We will show that the set S_j has the following important properties:

1. Each member of S_j is a feasible source–j rate function.

2. All members in S_j have identical $M_{U_j}^{\nu}(\alpha)$, and the members of S_j maximize $M_{U_j}^{\nu}(\alpha)$ over the set of all feasible source–j rate functions.

3. The member in S_j which has the largest $M_{V_j}^{\nu}(\alpha)$ has, in fact, the largest $M_{V_j}^{\nu}(\alpha)$ among all feasible source–j rate functions.

Hence, we will have shown that in order to find the source–j adversarial rate function corresponding to each choice (ν, α) we need only consider the rate functions in the set S_j. Further, since the rate functions in S_j are characterized by a single parameter, T_{off}, this essentially involves a single–parameter optimization problem. We now proceed to formally state and prove the properties listed above.

Theorem 4 *Every member of S_j is a feasible rate function.*

Proof. Fix a T_{off} and let $b_j(t)$ be the corresponding member of S_j. It follows immediately from the definition of $b_j(t)$ that

$$\int_0^t b_j(s)ds \leq \mathcal{E}_j(t) \text{ for all } 0 \leq t \leq T_j. \tag{5.22}$$

We can, in fact, show that

$$\int_0^t b_j(s)ds \le \mathcal{E}_j(t) \text{ for all } t \ge 0. \tag{5.23}$$

To see this consider any arbitrary $t = nT_j + s$, where n is some non–negative integer and $0 \le s \le T_j$.

$$\begin{aligned}
\int_0^t b_j(s)ds &= \int_0^{T_j} b_j(s)ds + \ldots + \int_{(n-1)T_j}^{nT_j} b_j(s)ds + \int_{nT_j}^{nT_j+s} b_j(s)ds \\
&\le nT_j\rho_j^{L_j} + \mathcal{E}_j(s) \\
&\le (\mathcal{E}_j(nT_j + s) - \mathcal{E}_j(s)) + \mathcal{E}_j(s) \\
&= \mathcal{E}_j(t)\,.
\end{aligned}$$

The first inequality follows from (5.22) and from the fact that the average rate of $b_j(t)$ over any period is $\rho_j^{L_j}$. The second inequality follows because the slope of $\mathcal{E}_j(t)$ is never less than $\rho_j^{L_j}$.

Also, because $b_j(t)$ is non–increasing over each of its periods, we have

$$\int_\tau^{t+\tau} b_j(s)ds \le \int_0^t b_j(s)ds \text{ for all } \tau \ge 0, \quad t \ge 0. \tag{5.24}$$

Combining (5.23) and (5.24) gives the desired result. ■

Theorem 5 *Each member of S_j maximizes $M_{U_j}^{\nu}(\alpha)$ over the set of all feasible rate functions.*

Proof. Each rate function in S_j leads to the following form for $u_j(t)$, the utilization of the jth segregated system: $u_j(t)$ is periodic with period T_j; and

$$u_j(t) = \begin{cases} c_j^{\nu} & 0 \le t \le D_{\text{on}} \\ 0 & D_{\text{on}} \le t \le T_j \end{cases}$$

where $D_{\text{on}} = \frac{\mathcal{E}_j(T_{\text{off}})}{c_j^{\nu}} = (\frac{\rho_j^{L_j}}{c_j^{\nu}})T_j$.

The corresponding steady–state random variable is

$$U_j = \begin{cases} c_j^{\nu} & \text{with probability } \frac{\rho_j^{L_j}}{c_j^{\nu}} \\ 0 & \text{with probability } (1 - \frac{\rho_j^{L_j}}{c_j^{\nu}}) \end{cases} \tag{5.25}$$

Note that $E[U_j] = \rho_j^{L_j}$.

For any feasible source, the steady state rate at which traffic leaves the jth segregated system, $U_j^{'}$ (say), must have a peak value less than or equal to c_j^{ν}. Further, because the segregated system is lossless, the long–run average rate at which traffic departs the jth segregated system must equal the long–run average rate at which traffic enters the system, which is at most $\rho_j^{L_j}$. Hence, we must have $E[U_j^{'}] \leq \rho_j^{L_j}$. Among all random variables which have a peak value less than or equal to c_j^{ν} and a mean value less than or equal to $\rho_j^{L_j}$, U_j as defined in (5.25) has the highest moment generating function, $M_{U_j}^{\nu}(\alpha)$. This is shown in the following argument (adapted from [48]). Let $U_j^{'}$ be any non–negative random variable with distribution $F_{U_j^{'}}(x)$ with a peak value $c^{'} \leq c_j^{\nu}$ and mean value $\mu^{'} \leq \rho_j^{L_j}$. Then, since $\alpha \geq 0$,

$$\begin{aligned} M_{U_j}^{\nu}(\alpha) - M_{U_j^{'}}^{\nu}(\alpha) &= (\frac{\rho_j^{L_j}}{c_j^{\nu}})e^{\alpha c_j^{\nu}} - \frac{\rho_j^{L_j}}{c_j^{\nu}} + 1 - \int_0^{c^{'}} e^{\alpha x} dF_{U_j^{'}}(x) \\ &\geq (\frac{\mu^{'}}{c_j^{\nu}})e^{\alpha c_j^{\nu}} - \frac{\mu^{'}}{c_j^{\nu}} - \int_0^{c^{'}} (e^{\alpha x} - 1) dF_{U_j^{'}}(x) \\ &= \frac{1}{c_j^{\nu}} \int_0^{c^{'}} [x(e^{\alpha c_j^{\nu}} - 1) - c_j^{\nu}(e^{\alpha x} - 1)] dF_{U_j^{'}}(x) \\ &\geq 0 \, . \end{aligned}$$

■

Let $b_j^*(t)$ be a rate function in S_j that has the largest $M_{V_j}^{\nu}(\alpha)$.

Theorem 6 *$b_j^*(t)$ maximizes $M_{V_j}^{\nu}(\alpha)$ among all feasible rate functions.*

Proof. Consider any feasible source–j rate function $b_j(t)$ with period T_j. The actual arrival rate at time t is $a_j(t) = b_j(t + \theta_j)$ where θ_j is the random phase. Here, we are concerned only with the steady–state distributions of the buffer contents and the utilization rate of the jth segregated system which are independent of the phase. Hence, in the rest of the proof, we will, without loss of generality, set the phase to zero and consider $b_j(t)$ to be the arrival rate at time t. The corresponding buffer contents process, $v_j(t)$, is also periodic with period T_j.

In general, both $b_j(t)$ and $v_j(t)$ can have rather complicated forms with several intervals within a period where each is non–zero. However, we will first show the desired result for

feasible rate functions that give a buffer content process of the form $v_j(t) > 0$ for $0 < t < \tau_j$ and $v_j(t) = 0$ for $\tau_j \leq t \leq T_j$, for some $0 < \tau_j < T_j$. For rate processes of this form we have

$$v_j(t) = \begin{cases} \int_0^t b_j(s)ds - c_j^\nu t & 0 \leq t \leq \tau_j \\ 0 & \tau_j \leq t \leq T_j \end{cases}$$

Note that, since $v_j(t) > 0$ for all $0 < t < \tau_j$, we must have

$$\tau_j \leq \delta_j \ . \tag{5.26}$$

We show next that $M_{V_j}^\nu(\alpha)$ corresponding to such a feasible rate function is smaller than that corresponding to $b_j^*(t)$. We do this by showing that there is a rate function in set S_j, $\bar{b}_j(t)$, with steady–state buffer contents $\bar{V}_j$ which is stochastically larger than V_j and which, hence, has a larger MGF (moment generating function).

Let T_{off} be such that $\mathcal{E}_j(T_{\text{off}}) = c_j\tau_j$. From (5.26) and (5.21) we get, $\mathcal{E}_j(T_{\text{off}}) < \mathcal{E}_j(\tau_j)$ if $\tau_j < \delta_j$ and $\mathcal{E}_j(T_{\text{off}}) = \mathcal{E}_j(\delta_j)$ if $\tau_j = \delta_j$. Hence, since $\mathcal{E}_j(\cdot)$ is non–decreasing and δ_j is uniquely defined, $T_{\text{off}} \leq \tau_j \leq \delta_j$. By definition, the rate function in S_j corresponding to this T_{off} is periodic with period $\bar{T}_j = \frac{\mathcal{E}_j(T_{\text{off}})}{\frac{L_j}{\rho_j}}$ and has the form

$$\bar{b}_j(t) = \begin{cases} \mathcal{E}_j^+(t) & 0 \leq t \leq T_{\text{off}} \\ 0 & T_{\text{off}} \leq t \leq \bar{T}_j \ . \end{cases}$$

The corresponding buffer contents at time t, $\bar{v}_j(t)$, is given as

$$\bar{v}_j(t) = \begin{cases} \mathcal{E}_j(t) - c_j^\nu t & 0 \leq t \leq T_{\text{off}} \\ \mathcal{E}_j(T_{\text{off}}) - c_j^\nu t & T_{\text{off}} \leq t \leq \tau_j \\ 0 & \tau_j \leq t \leq \bar{T}_j \end{cases}$$

Denote the corresponding steady–state random variable as $\bar{V}_j$.

Clearly, $v_j(t) \leq \bar{v}_j(t)$ for all $0 \leq t \leq T_{\text{off}}$. Note, also, that we cannot have $v_j(t) > \bar{v}_j(t)$ for any $T_{\text{off}} \leq t \leq \tau_j$ since that would require $v_j(t)$ to decrease at a rate strictly faster than c_j^ν, in order for both $v_j(t)$ and $\bar{v}_j(t)$ to be zero at τ_j. Hence, we get

$$v_j(t) \leq \bar{v}_j(t) \text{ for all } 0 \leq t \leq \tau_j \ . \tag{5.27}$$

Also, we can show that

$$T_j \geq \bar{T}_j \ . \tag{5.28}$$

To see this, note that the utilization rate of the jth segregated system with arrival rate $b_j(t)$ is c_j^ν whenever $v_j(t)$ is non–zero. Hence, $P(U_j = c_j^\nu) \geq \frac{\tau_j}{T_j}$. Also, since the average utilization rate must be equal to the average arrival rate, which in turn is smaller than $\rho_j^{L_j}$,

$$\rho_j^{L_j} \geq E[U_j] \geq c_j^\nu P(U_j = c_j^\nu) \geq c_j^\nu \frac{\tau_j}{T_j}$$

and so,

$$T_j \geq \frac{c_j^\nu \tau_j}{\rho_j^{L_j}} = \frac{\mathcal{E}_j(T_{\text{off}})}{\rho_j^{L_j}} = \bar{T}_j \ .$$

Equations (5.28) and (5.27) imply that

$$P(V_j > x) \leq P(\bar{V}_j > x) \text{ for all } x \geq 0 \ .$$

We have thus shown that V_j is stochastically smaller than $\bar{V}_j$ and hence has a smaller MGF. It is immediate from the definition of $b_j^*(t)$ that $M_{V_j}^\nu(\alpha)$ is smaller than that corresponding to $b_j^*(t)$.

We now extend this argument to the case of a general feasible rate function $b_j(t)$. Assume, without loss of generality, that the corresponding buffer content process $v_j(t)$ has m (some positive integer) non–zero portions within a single period, identified by $v_j^1, v_j^2, \ldots, v_j^m$ in the following manner:

$$v_j(t) = \begin{cases} 0 & 0 \leq t \leq t_j^1 \\ v_j^1(t - t_j^1) & t_j^1 \leq t \leq t_j^1 + \tau_j^1 \\ v_j^2(t - t_j^2) & t_j^2 \leq t \leq t_j^2 + \tau_j^2 \\ \vdots & \\ v_j^m(t - t_j^m) & t_j^m \leq t \leq t_j^m + \tau_j^m \\ 0 & t_j^m + \tau_j^m \leq t \leq T_j \end{cases}$$

where $\tau_j^i > 0$, $i = 1, 2, \ldots, m$, and $t_j^i \geq t_j^{i-1} + \tau_j^{i-1}$, $i = 2, \ldots, m$. Here, t_j^i and $t_j^i + \tau_j^i$ represent the endpoints of the ith non–zero portion.

We can express each non–zero portion $v_j^i(t)$ as a periodic function, with period T_j, of the following form:

$$v_j^i(t) = \begin{cases} \displaystyle\int_{t_j^i}^{t_j^i + t} b_j(s)ds - c_j^\nu t & 0 \leq t \leq \tau_j^i \\ 0 & \tau_j^i \leq t \leq T_j \ . \end{cases}$$

Let V_j^i denote the corresponding steady–state random variable with MGF $M_{V_j^i}^{\nu}(\alpha)$.

It is easily seen that

$$V_j = \begin{cases} V_j^1 & \text{with probability } \left(\frac{\tau_j^1}{\sum_{i=1}^m \tau_j^i}\right) T_j \\ \vdots & \\ V_j^m & \text{with probability } \left(\frac{\tau_j^m}{\sum_{i=1}^m \tau_j^i}\right) T_j \end{cases}$$

and hence,

$$M_{V_j}^{\nu}(\alpha) = \sum_{l=1}^{m} \left(\frac{\tau_j^l}{\sum_{i=1}^m \tau_j^i}\right) M_{V_j^l}^{\nu}(\alpha) \; . \tag{5.29}$$

Now, the ith non–zero portion, when viewed in isolation, has the simple form assumed in the earlier part of the proof, and can be viewed as the buffer contents at time t of the jth segregated system subject to the following arrival rate:

$$b_j^i(t) = \begin{cases} b_j(t_j^i + t) & 0 \le t \le \tau_j^i \\ 0 & \tau_j^i \le t \le T_j \; . \end{cases}$$

Note that $b_j^i(t)$ is also a feasible source–j rate function with period T_j. Hence, from our earlier argument, we know that $M_{V_j^i}^{\nu}(\alpha)$ is smaller than the MGF that corresponds to $b_j^*(t)$. Hence, from (5.29), we get that $M_{V_j}^{\nu}(\alpha)$ is also smaller than that corresponding to $b_j^*(t)$. We have thus shown that $b_j^*(t)$ maximizes $M_{V_j}^{\nu}(\alpha)$ over the set of all feasible source–j rate functions. ■

From Theorems 5 and 6 the following corollary is immediate.

Corollary 1 *There exists a rate function belonging to S_j which maximizes both $M_{U_j}^{\nu}(\alpha)$ and $M_{V_j}^{\nu}(\alpha)$ over the set of all feasible source–j rate functions. This rate function is the required source–j adversarial rate function corresponding to (ν, α).*

Thus, when $c_j^{\nu} > \rho_j^{L_j}$, in order to find the source–j adversarial rate function corresponding to any choice of (ν, α) we need only consider the rate functions in set S_j.

5.4.3 The Case of $c_j^{\nu} = \rho_j^{L_j}$

We now deal with the special case of $c_j^{\nu} = \rho_j^{L_j}$. When $c_j^{\nu} = \rho_j^{L_j}$ it is easily seen that the adversarial source–j rate function has the following form:

$$b_j(t) = \mathcal{E}_j^{+}(t) \;\; \text{for all } t \ge 0.$$

Clearly, this rate function satisfies (5.16). We will drop the requirement of periodicity for this special case and consider this rate function to be feasible. (Alternatively, we could consider this rate function to be trivially periodic with a period of $+\infty$.) This rate function leads to the following degenerate form of the corresponding steady–state random variables:

$$\begin{aligned} U_j^* &= c_j^\nu \quad \text{with probability 1} \\ V_j^* &= b_j^\nu \quad \text{with probability 1} \end{aligned}$$

with corresponding MGFs

$$\begin{aligned} M_{U_j^*}^\nu(\alpha) &= e^{\alpha c_j^\nu} \\ M_{V_j^*}^\nu(\alpha) &= e^{\alpha b_j^\nu} \end{aligned}$$

which are clearly the largest possible values for these quantities.

In the next section we consider input sources that are constrained by simple regulators and describe a heuristic procedure to efficiently compute P_{loss} for this case.

5.5 Simple Regulators

In the last section we showed that for each segregated system there exists a rate function in S_j which is adversarial to the greatest extent possible permitted by the regulator constraint $\mathcal{E}_j(t)$. The set S_j includes the extremal periodic on–off rate functions studied in LoPresti *et al.* [42]. It is therefore natural to pose the following question: Is the extremal periodic on–off rate function adversarial?

In this section we focus our attention on simple regulators $\mathcal{E}_j(t) = \min\{\rho_j^1, \sigma_j^2 + \rho_j^2 t\}$. We first show that the adversarial rate function in S_j *is not* the extremal on–off rate function used in LoPresti *et al.* This implies that the use of on–off rate functions, as in LoPresti *et al.*, can lead to overly optimistic admission regions. We then present an algorithm for calculating P_{loss} using the adversarial rate functions for each of the sources. This involves, for each source j, a search to find the T_{off} that leads to the most adversarial behavior.

5.5.1 Sub–Adversariality of On–Off Rate Functions

Fix a segregated system j. For ease of notation, let $P_j = \rho_j^1$, $\rho_j = \rho_j^2$ and $\sigma_j = \sigma_j^2$; the traffic constraint function is thus given by $\mathcal{E}_j(t) = \min(P_j t, \sigma_j + \rho_j t)$. We study 3 different

Rate Function	T_{on}	T_{off}	T	D_{on}	D_{off}
1	$\frac{\sigma}{P-\rho}$	$\frac{\sigma}{\rho}$	$\frac{\sigma P}{\rho(P-\rho)}$	$\frac{\sigma P}{c(P-\rho)}$	$\sigma\frac{1-\rho+P(1-1/c)}{\rho(P-\rho)}$
2	$\frac{\sigma}{c-\rho}$	$\frac{\sigma}{\rho}$	$\frac{\sigma c}{\rho(c-\rho)}$	$\frac{\sigma}{c-\rho}$	$\frac{\sigma}{\rho}$
3	u	$\frac{\sigma}{\rho}$	$u+\frac{\sigma}{\rho}$	$\frac{\sigma+u\rho}{c}$	$\frac{(u\rho+\sigma)(c-\rho)}{\rho c}$

Table 5.1: On and off times of rate functions and corresponding segregated systems.

rate functions, all complying with the imposed traffic constraint function. All these rate functions belong to S_j. Figure 5.3a gives the plots of the traffic constraint function, $\mathcal{E}_j(t)$, and the actual arrivals, $A_j(t)$, of the studied rate functions. Figure 5.3b depicts the arrival rate function $b_j(t)$. Figure 5.3c gives the link utilization $u_j(t)$. Figure 5.3d shows the buffer contents of the segregated system. Note that traffic leaves the segregated system at rate c_j whenever the buffer is nonempty. For the remainder of this section, we remove the subscript j from all notations.

Rate function 1 is the extremal on–off rate function used by Elwalid *et al.* [14] and LoPresti *et al.* [42]. It transmits at peak rate P for $T_{\text{on}_1} = \sigma/(P-\rho)$, at which time the token pool is completely emptied. The rate function then turns off and waits for $T_{\text{off}_1} = \sigma/\rho$, allowing the token pool to be refilled with σ tokens. The rate function then transmits the next burst of size PT_{on_1} at peak rate. The buffer is filled at rate $P-c$ while the source transmits at rate P. The maximum buffer contents is therefore $b = (P-c)T_{\text{on}_1}$. After the source has turned off, the buffer is drained at rate c. The utilization of the segregated system is c for $D_{\text{on}_1} = T_{\text{on}_1} + b/c$ and 0 for $D_{\text{off}_1} = T_{\text{off}_1} - b/c$. Rate Function 1 along with the other two rate functions are summarized in Table 5.1.

Rate function 2 transmits at peak rate P for T_{on_1}, it then continues sending traffic at rate ρ into the segregated system until the corresponding buffer process hits zero. As is the case for rate function 1, the buffer is filled up to b at rate $P-c$; it is now however drained at rate $c-\rho$. The source transmits therefore greedily for $T_{\text{on}_2} = T_{\text{on}_1} + b/(c-\rho)$. It then shuts off, waits until the token pool is replenished and repeats the described transmission pattern.

Rate function 3 generalizes the rate function behaviors discussed so far. It transmits greedily for u, $T_{\text{on}_1} \le u \le T_{\text{on}_2}$, that is, it transmits at rate P for T_{on_1} and then at rate

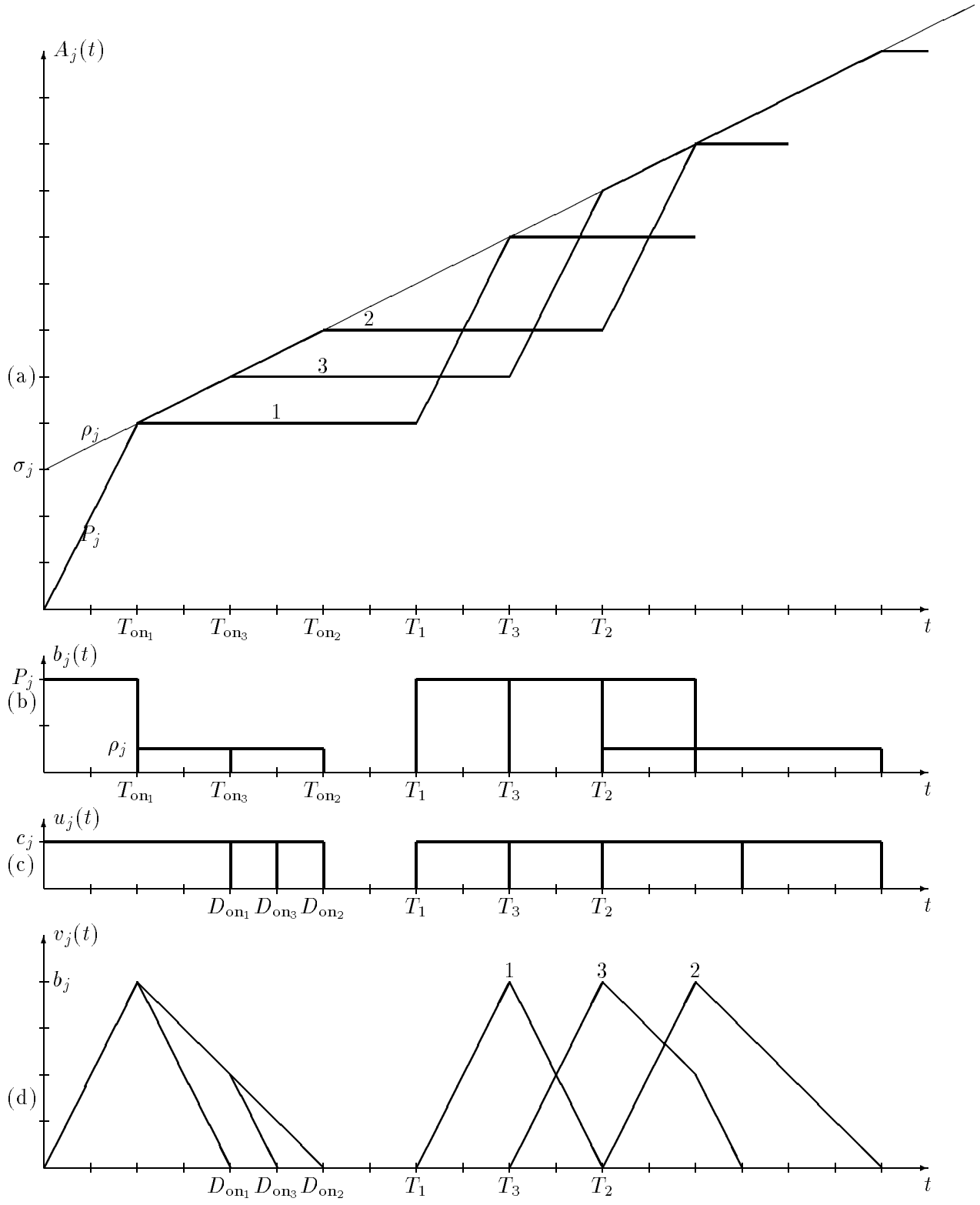


Figure 5.3: Illustration of rate functions 1, 2 and 3. (a) Amount of traffic arriving to the segregated system $A_j(t)$. (b) Arrival rate process $b_j(t)$. (c) Utilization process $u_j(t)$. (d) Buffer content process $v_j(t)$.

ρ for $u - T_{\text{on}_2}$. The corresponding buffer process is depicted in Figure 5.3d. The buffer is filled to b at rate $P-c$. It is then drained at rate $c-\rho$ during the interval $[T_{\text{on}_1}, u]$. Let $v(u)$ denote the buffer contents at time u; clearly, $v(u) = \sigma + u(\rho - c)$. The remaining traffic $v(u)$ is drained at rate c. Loosely speaking, rate function 3 lies between the extremes of rate function 1 and rate function 2: it is equivalent to rate function 1 for $u = T_{\text{on}_1}$ and is equivalent to rate function 2 for $u = T_{\text{on}_2}$.

We now turn our attention to the buffer processes of the described rate functions. Let V_1, V_2 and V_3 be random variables denoting the buffer contents corresponding to rate function 1, 2 and 3. It can be easily verified that V_1 and V_2 have identical distribution functions:

$$P(V_1 \leq x) = P(V_2 \leq x) = 1 - \omega + x\frac{\omega}{b} \quad 0 \leq x \leq b, \tag{5.30}$$

where $\omega = \rho/c$ is the long run probability that the segregated system is busy. The distribution function of V_3 is given by

$$P(V_3 \leq x) = \begin{cases} 1 - \omega + x\frac{\omega}{b}\frac{P\sigma}{(P-\rho)(\rho u+\sigma)} & \text{for } 0 \leq x \leq v(u) \\ 1 - \omega + x\frac{\omega}{b}\frac{c\sigma}{(c-\rho)(u\rho+\sigma)} + \frac{u\rho^2}{(c-\rho)(u\rho+\sigma)} - \frac{\rho^2}{(c-\rho)c} & \text{for } v(u) \leq x \leq b. \end{cases}$$

Next we show that V_3 is strictly stochastically larger than V_1 and V_2 whenever $T_{\text{on}_1} < u < T_{\text{on}_2}$. First, note that

$$\frac{P\sigma}{(P-\rho)(\rho u+\sigma)} < 1 \quad \text{for } u > T_{\text{on}_1}.$$

Furthermore, it can be shown that

$$x\frac{\omega}{b}\frac{c\sigma}{(c-\rho)(u\rho+\sigma)} + \frac{u\rho^2}{(c-\rho)(u\rho+\sigma)} - \frac{\rho^2}{(c-\rho)c} < x\frac{\omega}{b}$$

for $u < T_{\text{on}_2}$ and $x < b$. Hence,

$$P(V_3 \leq x) < P(V_1 \leq x) \quad \text{for } 0 \leq x < b. \tag{5.31}$$

Thus V_3 is strictly stochastically larger than V_1 and V_2. This implies that the moment generating function of V_3 is larger than that of V_1 and V_2. The loss probability computed with rate function 3 is therefore larger than that corresponding to rate functions 1 and 2. Rate function 1, which is used in LoPresti *et al.* , can therefore lead to overly optimistic admission decisions.

$M_{V_3}(\alpha)$	$1-\omega+\frac{\omega\sigma}{\alpha b(\rho u+\sigma)}\{\frac{\rho(c-P)}{(P-\rho)(c-\rho)}e^{\alpha v(u)}+\frac{1}{1-\omega}e^{\alpha b}-\frac{P}{P-\rho}\}$
$\frac{\partial M_{V_3}(\alpha)}{\partial\alpha}$	$\frac{\omega\sigma}{\alpha^2 b(\rho u+\sigma)}\{\frac{\rho(c-P)}{(P-\rho)(c-\rho)}[\alpha v(u)-1]e^{\alpha v(u)}+\frac{1}{1-\omega}(b\alpha-1)e^{\alpha b}+\frac{P}{P-\rho}\}$
$\frac{\partial^2 M_{V_3}(\alpha)}{\partial\alpha^2}$	$\frac{\omega\sigma}{\alpha^3 b(\rho u+\sigma)}\{\frac{\rho(c-P)}{(P-\rho)(c-\rho)}[\alpha^2 v^2(u)-2\alpha v(u)+2]e^{\alpha v(u)}+\frac{1}{1-\omega}(\alpha^2 b^2-2\alpha b+2)e^{\alpha b}-\frac{2P}{P-\rho}\}$
$\frac{\partial M_{V_3}(\alpha)}{\partial u}$	$\frac{\omega\sigma}{\alpha b(\rho u+\sigma)^2}\{\frac{\rho(P-c)}{(P-\rho)}[\alpha\rho u+\alpha\sigma+\frac{\omega}{1-\omega}]e^{\alpha v(u)}-\frac{\rho}{1-\omega}e^{\alpha b}+\frac{P\rho}{P-\rho}\}$

Table 5.2: The moment generating function of the buffer process V_3 and its derivatives

5.5.2 Finding the most adversarial Rate Function

In this subsection we espouse the problem of finding the most adversarial rate function among the rate functions fitting the template of rate function 3. Toward this end we need to find the on–time u that maximizes the moment generating function of V_3. The moment generating function of V_3, defined as $M_{V_3}(\alpha) = E[e^{\alpha V_3}]$, and its derivative with respect to u are given in Table 5.2. The table gives furthermore the first and second derivative of $M_{V_3}(\alpha)$ with respect to s. These expressions are needed for the computation of P_{loss} (see Section refbum:numex).

Setting $\partial M_{V_3}(\alpha)/\partial u$ to zero, we obtain

$$(\alpha\rho u+\alpha\sigma+\frac{\omega}{1-\omega})e^{-\alpha(c-\rho)u}=\frac{(P-\rho)e^{\alpha b}-P(1-\omega)}{(1-\omega)(P-c)e^{\alpha\sigma}}. \tag{5.32}$$

It can be shown that (5.32) has exactly one solution in $[T_{\text{on}_1}, T_{\text{on}_2}]$. It can be computed efficiently with Newtons method [53] using $(T_{\text{on}_1}+T_{\text{on}_2})/2$ as initial solution. We observed in our numerical investigations that $(T_{\text{on}_1}+T_{\text{on}_2})/2$ provides in many cases a good approximation of the solution of (5.32). Rate function 3 with $u=(T_{\text{on}_1}+T_{\text{on}_2})/2$ may therefore be used as an approximation of the most adversarial rate function.

5.5.3 Numerical Examples

In this subsection we report on some numerical investigations with the most adversarial rate function. For the computation of P_{loss} we essentially follow the numerical procedure outlined in LoPresti *et al.* [42]. In addition to the computations conducted by LoPresti *et al.*, however, we solve (5.32) in order to find the most adversarial rate function.

We compare our approach with that of Elwalid *et al.* [14] and LoPretsi *et al.* in Figure 5.3. We use the same two source classes (see Table 5.3) as LoPresti *et al.* in [42,

class	ρ(Mbps)	P(Mbps)	σ(cells)
1	0.15	1.5	225
2	0.15	6	24.4

Table 5.3: Leaky Bucket parameters of sources.

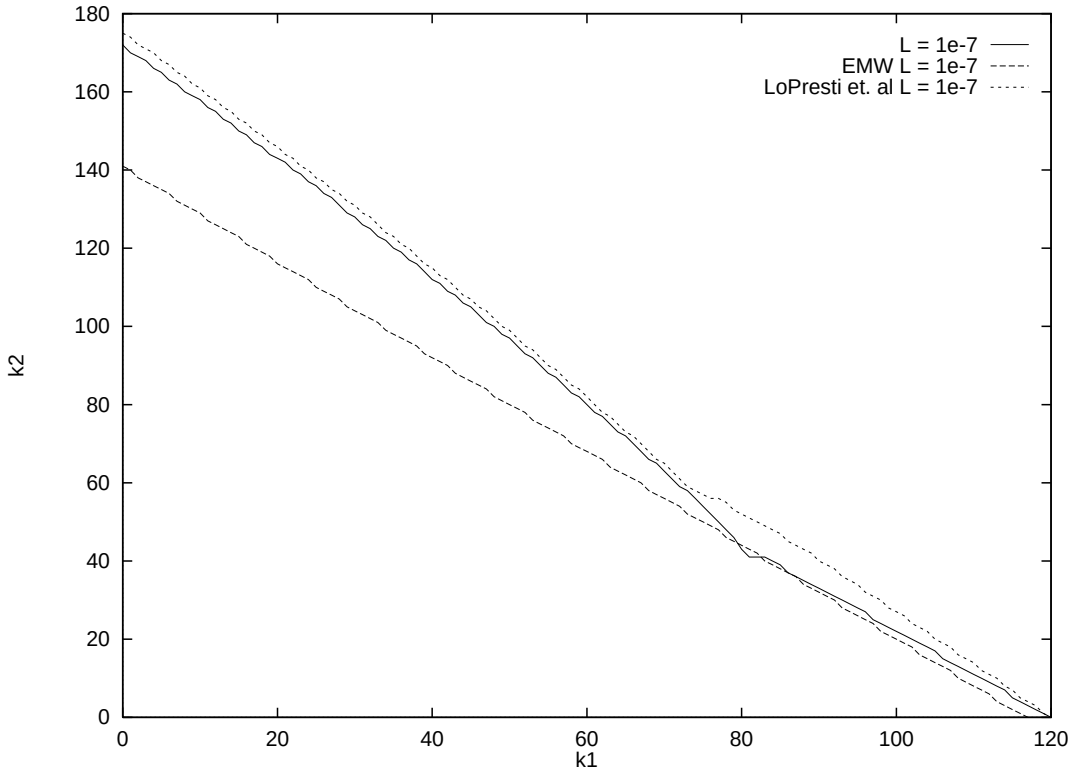


Figure 5.4: Comparison of our approach with Elwalid *et al.* [14] and LoPresti *et al.* [42].

Fig. 15]. They in turn use the same parameters as Elwalid *et al.* in [14, Fig. 13]. The bandwidth and buffer size are $C = 45$ Mbps and $B = 1000$ cells (1 cell = 53 bytes) in this example. The figure depicts the admission region corresponding to the admission control criterion $P_{\text{loss}} \leq 10^{-7}$. We observe that employing the truly adversarial rate function results in an admission region that lies generally between that of Elwalid *et al.* and LoPresti *et al.*. Because we are using the truly adversarial sources, our approach has a smaller admission region than LoPresti *et al.*. Our approach admits slightly less connections than the approach of LoPresti *et al.* in the range $0 \leq k_1 \leq 75$. For $k_1 = 0$, we admit 172 connections of class 2 while LoPresti *et al.* allow 175 connections. The gap between the two

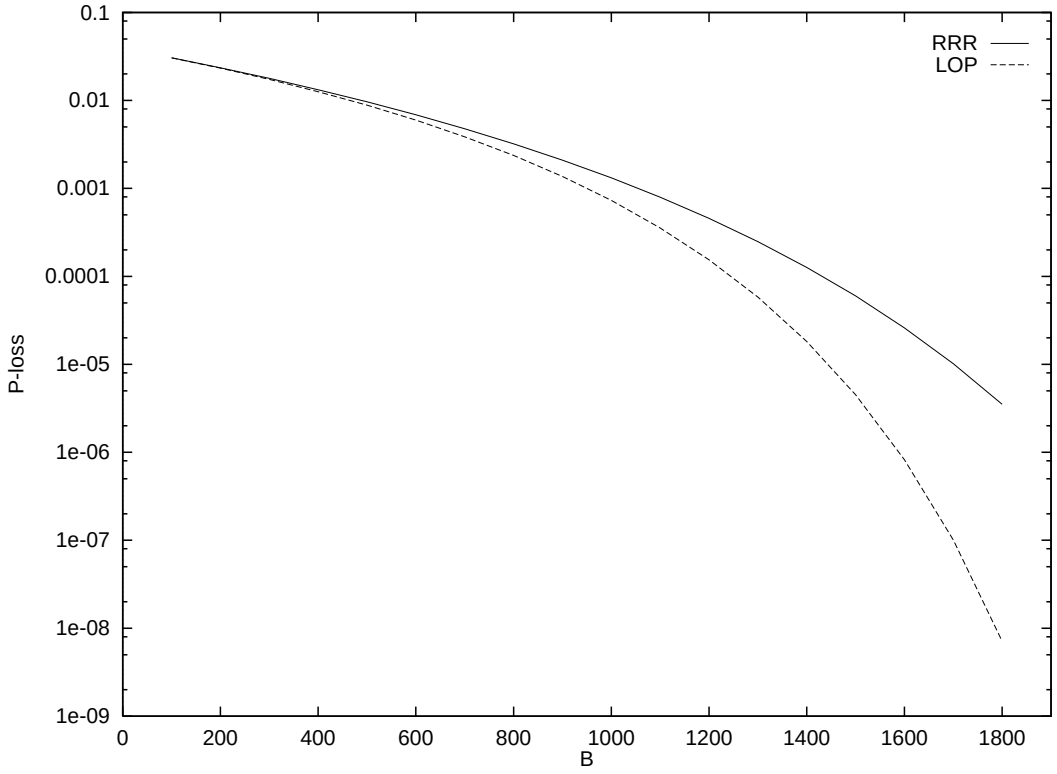


Figure 5.5: P_{loss} as a function of buffer size B.

approaches widens for $k_1 > 75$. This is due to the fact that the optimal resource allocation according to Theorem 1 allocates $c_2^{\nu} = \rho_2$ in this region. Rate function 3 degenerates to the form described in Section 5.4.3 for this allocation. The moment generating function of this rate function is significantly larger that that corresponding to rate function 1, resulting in a noticably smaller admission region for our approach. The gap is at its widest for $k_1 = 81$. Our approach admits 41 connections of class 2 wile LoPresti *et al.* admit 51 connections.

In Figure 5.5 we consider a single source class with $P = 5$ cells/sec, $\rho = 2.5$ cells/sec and $\sigma = 20$ cells. (This choice of parameters is inspired by Oechslin [50].) We consider transmitting the traffic of 200 of these sources over a link of capacity $C = 575$ cells/sec. The figure shows P_{loss} computed according to our approach (RRR) and LoPresti *et al.* as a function of the buffer size B. We observe that both approaches give about the same loss probability for buffers smaller than 800 cells. For large buffers, however, the approaches differ greatly. For $B = 1400$ cells the loss probability computed according to LoPresti *et al.* is about one order of magnitude smaller that that computed with the

most adversarial rate function. For $B = 1700$ cells the gap widens to roughly two orders of magnitude. We conclude from the figure that the approach of LoPresti *et al.* can significantly underestimate the loss probability.

Chapter 6

Smoothing and Bufferless Multiplexing of Regulated Traffic

6.1 Overview

Over the past ten years, significant research effort has addressed the important problem of guaranteeing QoS to multimedia traffic in a packet–switched network. The goal has been to develop traffic management schemes that allow for high link utilizations while simultaneously guaranteeing that the QoS requirements of the ongoing connections are met. It is generally agreed that high link utilizations can only be achieved by allowing traffic to be statistically multiplexed, i.e., by allowing each connection's traffic to have a small amount of loss and exploiting the statistical independence of the connections' traffic [62][32][33][25]. It is also the view of many researchers that QoS can only be guaranteed by requiring the traffic to be regulated (e.g., by leaky buckets) at the edges of the network [35][75] [23][52][14][42][54].

In recent years the problem of providing QoS guarantees to regulated sources which are statistically multiplexed in a shared buffer has been carefully studied [14][42][54]. The existing solutions, however, do not extend to the network environment in a satisfactory manner. Also in recent years, the problem of providing end–to–end deterministic guarantees to regulated traffic in *networks* has been adequately solved [78][77][23][52]. The deterministic QoS guarantees, however, typically imply a small connection–carrying capacity

for networks with bursty multimedia traffic. In this chapter we lay the groundwork for a traffic–management architecture that provides end–to–end QoS guarantees while simultaneously giving a relatively large connection–carrying capacity. We restrict our attention to a network consisting of a single node in this thesis.

In this chapter we view traffic as fluid. The fluid model, which closely approximates a packetized model with small packets, permits us to focus on the central issues and significantly simplifies notation. We suppose that the traffic sent into the node by each connection is regulated by a connection–specific cascade of leaky buckets. A cascade of leaky buckets is more general than the two–leaky–bucket regulator, commonly used in the literature [14][42], and can more accurately characterize a source's traffic. Moreover, cascaded–leaky–bucket traffic can easily be policed. For admission control, all that we know about a connection's traffic is its regulator constraint defined by its cascade of leaky buckets; in particular, we do not have available statistical characterizations of the traffic.

We also assume that the following natural QoS requirement is in force: the fraction of traffic that exceeds a specific delay limit must be below a prescribed bound. Traffic which overflows at a buffer is considered as having infinite delay, and therefore violates the QoS requirement. Importantly, we permit each connection to have its own limit on the nodal delay and its own bound on the fraction of traffic that exceeds this delay limit. This QoS requirement is particularly appropriate for multimedia traffic, whereby timestamping and a playout buffer can ensure the continuous playout of video or audio without jitter.

Given each connection's traffic characterization and its QoS requirement, we address the following problem: How should we manage the traffic and perform admission control in order to guarantee QoS while maintaining a large connection–admission region? We advocate the following simple and pragmatic scheme: (i) smooth each connection's traffic at the connection's input as much as allowed by the connection's delay constraint; (ii) employ bufferless statistical multiplexing within the node; (iii) base admission control on the worst–case assumption that sources are adversarial to the extent permitted by the connection's regulator, while concurrently assuming the connections generate traffic independently. This scheme enjoys the following features:

- Admission control is solely based on the connections' regulator parameters, which

are policable. It is not based on more complex, difficult–to–police statistical characterizations.

- It allows for statistical multiplexing at the node while meeting the QoS requirements. The smoothing at the input increases the statistical multiplexing gain.

- It allows for per–connection QoS requirements: the connections can have vastly different delay and loss requirements.

- Because the multiplexing is bufferless, the switch requires only small input buffers (when traffic is packetized), thereby reducing switch cost.

- A connection's traffic characterization does not change as the traffic passes through the bufferless multiplexer.

It is this last feature that is particularly useful when extending the traffic management scheme to a multihop network. With our scheme the traffic leaving the network node conforms to the same regulator constraints as the traffic entering the node. With shared buffer multiplexers it is difficult (if not impossible) to tightly characterize a connection's traffic once the traffic passes through a shared buffer. One solution to this "down–stream" problem proposed in the literature [78][77][23][52] is to place resmoothers after the buffered multiplexer; the resmoothers smooth each connection's traffic so that it conforms to its original regulator constraints. We show in this chapter that our approach consisting of ingress smoothing and subsequent bufferless multiplexing is a viable alternative to the buffered multiplexing/resmoothing scheme in the literature.

This chapter is organized as follows. In Section 6.2 we formally define the cascaded leaky–bucket regulators and the QoS requirement. In Section 6.3 we determine the worst–case traffic for a single–link and outline our smoothing and admission control procedure. We also consider general smoothers and show that the optimal smoother is a single–buffer smoother which smoothes traffic as much as the delay limit permits. In Section 6.4 we present numerical results using MPEG–encoded traces. In Section 6.5 we compare our scheme to designs based on buffered statistical multiplexing. We conclude in Section 6.6.

6.2 Regulated Traffic and the QoS Requirement

In this chapter we focus on a single node consisting of a bufferless multiplexer that feeds into a link of capacity C. We view traffic as fluid, i.e., packets are infinitesimal. Consider a set of J connections. Each connection j has an associated regulator function, denoted by $\mathcal{E}_j(t), t \geq 0$. The regulator function constrains the amount of traffic that the jth connection can send into the node over all time intervals. Specifically, if $A_j(t)$ is the amount of traffic that the jth connection sends to the node over the interval $[0, t]$, then $A_j(\cdot)$ is required to satisfy

$$A_j(t+\tau) - A_j(\tau) \leq \mathcal{E}_j(t) \text{ for all } \tau \geq 0, \quad t \geq 0. \tag{6.1}$$

A popular regulator is the simple regulator, which consists of a peak–rate controller in series with a leaky bucket; for the simple regulator, the regulator function takes the following form:

$$\mathcal{E}_j(t) = \min\{\rho_j^1 t,\ \sigma_j^2 + \rho_j^2 t\}.$$

For a given source type, the bound on the traffic provided by the simple regulator may be loose and lead to overly conservative admission control decisions. For many source types (e.g., for VBR video), it is possible to get a tighter bound on the traffic and dramatically increase the admission region. In particular, regulator functions of the form

$$\mathcal{E}_j(t) = \min\{\rho_j^1 t,\ \sigma_j^2 + \rho_j^2 t, \ldots,\ \sigma_j^{L_j} + \rho_j^{L_j} t\} \tag{6.2}$$

are easily implemented with cascaded leaky buckets; it is shown in (see [75]) that the additional leaky buckets can lead to substantially larger admission regions for deterministic multiplexing. We shall show that this is also true for statistical multiplexing. Throughout this chapter we assume that each regulator has the form (6.2). Without loss of generality we may assume that $\rho_j^1 > \rho_j^2 > \cdots > \rho_j^{L_j}$ and $\sigma_j^2 < \sigma_j^3 < \cdots < \sigma_j^{L_j}$. For ease of notation, we set $\rho_j = \rho_j^{L_j}$. Note that for connection–j traffic, the long–run average rate is no greater than ρ_j and the peak rate is never greater than ρ_j^1.

Each connection also has a QoS requirement. In this chapter we consider a QoS requirement that is particularly appropriate for multimedia traffic, such as audio and video traffic. Specifically, each connection has a connection–specific delay limit and a connection–specific loss bound. Denote d_j and ϵ_j for the delay limit and loss bound for the jth connection.

Any traffic that overflows at a buffer is considered to have infinite delay, and therefore violates the delay limit. The QoS requirement is as follows: for each connection j, the long–run fraction of traffic that is delayed by more than d_j seconds must be less than ϵ_j.

This QoS requirement can assure continuous, uninterrupted playback for a multimedia connection as follows. Each bit (or packet for packetized traffic) is time–stamped at the source. If a bit from connection j is time–stamped with value x, the bit (if not lost in the node) arrives at the receiver no later than $x + d_j$. The receiver delays playout of the bit until time $x + d_j$. Thus, by including a buffer at each receiver, the receiver can playback a multimedia stream without jitter with a fixed delay of d_j and with bit loss probability of at most ϵ_j.

The strategy that we take in this chapter is to pass each connection's traffic through a smoother at the connection's input to the node. We design the smoother for the jth connection so that the jth connection's traffic is *never* delayed at the smoother by more than d_j. After having smoothed a connection's traffic, we pass the smoothed traffic to the node. At the link the connection's traffic is multiplexed with traffic from other connections. The second aspect of our strategy is to remove all of the buffers in the node; that is, we use bufferless statistical multiplexing rather than buffered multiplexing before the link. In our fluid model, a connection's traffic that arrives to a bufferless link either flows through the link without any delay or overflows at the link, and therefore has infinite delay. In order to satisfy the jth connection's QoS requirement, it therefore suffices that the fraction of connection–j traffic that overflows the link be less than ϵ_j. Also, if the loss at the link is small, we can reasonably approximate a connection's traffic at the output of the multiplexer as being identical to its traffic at the input to the multiplexer. In other words, a connection that satisfies the regulator constraint $\mathcal{E}_j(t)$ at the input of the node satisfies the same regulator constraint $\mathcal{E}_j(t)$ at the output of the node. Our scheme extends therefore in a straightforward manner from a single node to a general network. Our approach is illustrated in Figure 6.1.

For the smoother at the jth connection's input, initially we use a buffer which serves the traffic at rate c_j^*. When the smoother buffer is nonempty, traffic is drained from the smoother at rate c_j^*. When the smoother buffer is empty and connection–j's traffic is arriving at a rate less than c_j^*, traffic leaves the smoother exactly at the rate at which it

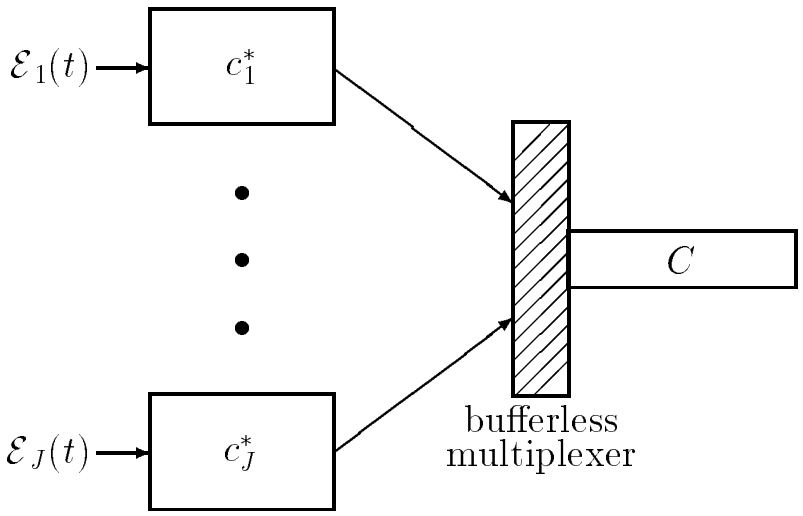


Figure 6.1: The traffic of the jth connections is characterized by the regulator function $\mathcal{E}_j(t)$. The traffic is passed through a smoother with rate c_j^* and then multiplexed onto a link with capacity C.

enters the buffer. For the fluid model and QoS criterion of this chapter we shall show that more complex smoothers consisting of cascaded leaky buckets do not improve performance.

Using the theory developed in [6], it can be shown that the maximum delay at the smoother is

$$\max_{t\geq 0}\left\{\frac{\mathcal{E}_j(t)}{c_j^*}-t\right\}. \tag{6.3}$$

Also, because the bufferless multiplexer and link introduce no delays, traffic from the jth connection that flows through the node without loss has the maximum delay of the smoother. We set the smoother rate to

$$c_j^*=\min\left\{c_j\geq 0:\max_{t\geq 0}\left\{\frac{\mathcal{E}_j(t)}{c_j}-t\right\}\leq d_j\right\}\ , \tag{6.4}$$

so that the traffic that passes through the node (i.e., traffic which does not overflow at the link) is not be delayed by more than d_j. It is straightforward to show from (6.4) that the smoother rate can be expressed as

$$c_j^*=\max_{t\geq 0}\frac{\mathcal{E}_j(t)}{d_j+t}. \tag{6.5}$$

6.3 Analysis of the Single Link

We focus in this chapter on a single link with J connections. Connection j has a regulator constraint function $\mathcal{E}_j(t)$ and QoS parameters d_j and ϵ_j. Now regard the jth arrival process as a stochastic process. Let $(A_j(t),\ t \geq 0)$ denote the jth arrival process, and let $(A_j(t,\omega),\ t \geq 0)$ denote a realization of the stochastic process. Also let $\mathbf{A}(t) = (A_1(t), \ldots, A_J(t))$, and let $(\mathbf{A}(t),\ t \geq 0)$ be the associated vector stochastic arrival process. We say that the vector arrival process $(\mathbf{A}(t),\ t \geq 0)$ is *feasible* if (i) the component arrival processes $(A_j(t),\ t \geq 0)$, $j = 1, \ldots, J$, are independent, and (ii) for each $j = 1, \ldots, J$, each realization $(A_j(t,\omega),\ t \geq 0)$ satisfies the regulator constraint

$$A_j(t+\tau,\omega) - A_j(\tau,\omega) \leq \mathcal{E}_j(t) \text{ for all } \tau \geq 0, \quad t \geq 0. \tag{6.6}$$

Denote $\mathcal{A}$ for the set of all feasible vector arrival processes $(\mathbf{A}(t),\ t \geq 0)$.

Our first goal is to develop a straightforward procedure to determine whether the QoS requirements are met for all possible feasible stochastic arrival processes. For a fixed feasible vector arrival process $(\mathbf{A}(t),\ t \geq 0)$, let $U_j(t)$ be the rate at which traffic from the jth connection leaves the associated smoother at time t, and let U_j be the corresponding steady–state random variable. Consider multiplexing $(\mathbf{A}(t),\ t \geq 0)$ traffic streams onto a bufferless multiplexer of rate C. The long–run average fraction of traffic lost by connection j is

$$P_{\text{loss}}^{\text{info}}(j) = \frac{E\left[(\sum_{k=1}^{J} U_k - C)^+ \frac{U_j}{\sum_{k=1}^{J} U_k}\right]}{E[U_j]}. \tag{6.7}$$

In the definition of $P_{\text{loss}}^{\text{info}}(j)$ we make the natural assumption that traffic loss at the bufferless multiplexer is split between the sources in a manner proportional to the rate at which the sources send traffic into the multiplexer. Note that $P_{\text{loss}}^{\text{info}}(j)$ keeps track of loss for each individual connection.

Although $P_{\text{loss}}^{\text{info}}(j)$ is an appealing performance measure, we have found it to be mathematically unwieldy. Instead of $P_{\text{loss}}^{\text{info}}(j)$ we shall work with a bound on $P_{\text{loss}}^{\text{info}}(j)$ which is more tractable and which preserves the essential characteristics of the original performance measure. Noting that the term in the expectation of the numerator of equation (6.7) is

non–zero only when $\sum_{k=1}^{J} U_k > C$, we obtain:

$$P_{\text{loss}}^{\text{info}}(j) \leq \frac{E\left[(\sum_{k=1}^{J} U_k - C)^+ U_j\right]}{C \cdot E[U_j]} := P_{\text{loss}}(j). \tag{6.8}$$

In most practical circumstances the QoS requirement specifies traffic loss to be miniscule, on the order of $\epsilon_j = 10^{-6}$ or less. Thus we expect the bound to be very tight: during the rare event when $\sum_{j=1}^{J} U_j$ exceeds C, we expect $\sum_{j=1}^{J} U_j$ to be very close to C. Henceforth, we focus on the bound $P_{\text{loss}}(j)$, and we refer to $P_{\text{loss}}(j)$ as the *loss probability for the jth connection* . In Section 6.4 we provide numerical results which show that $P_{\text{loss}}(j)$ is very nearly equal to the actual loss probability $P_{\text{loss}}^{\text{info}}(j)$.

By taking the supremum over all the feasible vector stochastic processes, we obtain the following worst–case loss probability of the jth connection:

$$\phi_j^* = \sup_{\mathcal{A}} \frac{E\left[(\sum_{k=1}^{J} U_k - C)^+ U_j\right]}{C \cdot E[U_j]} \tag{6.9}$$

If $\phi_j^* \leq \epsilon_j$ for all $j = 1, \ldots, J$, then the QoS requirements are guaranteed to be met for all feasible vector arrival processes, that is, for all independent arrival processes whose sample paths satisfy the regulator constraints. In our strategy, at connection admission we determine whether $\phi_j^* \leq \epsilon_j$ for all $j = 1, \ldots, J$ will continue to hold when adding the new connection. If not, the connection is rejected. Thus, we need to develop an efficient method to compute the bounds $\phi_1^*, \ldots, \phi_J^*$. As a first step in computing these bounds, we need to explicitly determine the random variables $U_1, \ldots, U_J$ that attain the supremum in (6.9).

Lemma 1 *Let $U_1^*, \ldots, U_J^*$ be independent random variables, with U_j^* having distribution*

$$U_j^* = \begin{cases} c_j^* & \text{with probability } \frac{\rho_j}{c_j^*} \\ 0 & \text{with probability } 1 - \frac{\rho_j}{c_j^*} \end{cases}.$$

There exists a feasible vector arrival process which produces the steady–state rate variables $U_1^*, \ldots, U_J^*$ at the smoother outputs.

Proof. The proof is by construction. For each $j = 1, \ldots, J$, let

$$t_j = \frac{\sigma_j^2}{\rho_j^1 - \rho_j^2}$$

and

$$T_j = \frac{\rho_j^1 \sigma_j^2}{(\rho_j^1 - \rho_j^2)\rho_j}.$$

Also let $\theta_1, \ldots, \theta_J$ be independent random variables with θ_j uniformly distributed over $[0, T_j]$. Let $b_j(t)$ be a deterministic periodic function with period T_j such that

$$b_j(t) = \begin{cases} \rho_j^1 & 0 \le t < t_j \\ 0 & t_j \le t \le T_j \ . \end{cases}$$

Define the jth arrival stochastic process as

$$A_j(t) = \int_0^t b_j(s + \theta_j) ds.$$

Thus each component arrival process $(A_j(t),\ t \ge 0)$ is generated by a periodic on–off source; the jth process has peak rate ρ_j^1 and average rate ρ_j. By sending each component process $(A_j(t),\ t \ge 0)$ into its respective smoother, we obtain an on–off process whose peak rate is c_j^* and whose average rate is ρ_j. Also, the component processes are independent; thus the vector arrival process produces the steady–state random variables $U_1^*, \ldots, U_J^*$ at the smoother outputs.

It remains to show that each realization of $(A_j(t),\ t \ge 0)$ satisfies the regulator constraint (6.6). It follows immediately from the definition of $b_j(t)$ that

$$\int_0^t b_j(s) ds \le \mathcal{E}_j(t) \text{ for all } 0 \le t \le T_j. \tag{6.10}$$

We can, in fact, show that

$$\int_0^t b_j(s) ds \le \mathcal{E}_j(t) \text{ for all } t \ge 0. \tag{6.11}$$

To see this consider any arbitrary $t = nT_j + s$, where n is some non–negative integer and $0 \le s \le T_j$. We have

$$\begin{aligned} \int_0^t b_j(s) ds &= \int_0^{T_j} b_j(s) ds + \ldots + \int_{(n-1)T_j}^{nT_j} b_j(s) ds + \int_{nT_j}^{nT_j + s} b_j(s) ds \\ &\le nT_j \rho_j + \mathcal{E}_j(s) \\ &\le [\mathcal{E}_j(nT_j + s) - \mathcal{E}_j(s)] + \mathcal{E}_j(s) \\ &= \mathcal{E}_j(t) \ . \end{aligned}$$

The first inequality follows from (6.10) and from the fact that the average rate of $b_j(t)$ over any period of length T_j is ρ_j. The second inequality follows because the slope of $\mathcal{E}_j(t)$ is never less than ρ_j. This establishes (6.11). Finally because $b_j(t)$ is non–increasing over each of its periods, we have

$$\int_{\tau}^{t+\tau} b_j(s)ds \leq \int_0^t b_j(s)ds \text{ for all } \tau \geq 0, \quad t \geq 0. \tag{6.12}$$

Combining (6.11) and (6.12) proves that each realization of $(A_j(t),\ t \geq 0)$ satisfies the regulator constraint (6.6).

■

We now show that the random variables $U_1^*, \ldots, U_J^*$ attain the supremum in (6.9). This result will lead to a simple procedure for calculating the worst–case loss probabilities $\phi_1^*, \ldots, \phi_J^*$. To this end, we will need to make use of a concept from stochastic ordering. A random variable X is said to be smaller than a random variable Y in the sense of the *increasing convex stochastic (ics) ordering*, written as $X \leq_{icx} Y$, if $E[h(X)] \leq E[h(Y)]$ for all increasing, convex functions $h(\cdot)$.

Theorem 7 *For each $j = 1, 2, \ldots, J$, worst–case loss probability for the jth connection is*

$$\phi_j^* = \frac{E\left[(\sum_{k=1}^J U_k^* - C)^+ U_j^*\right]}{C \cdot E[U_j^*]}$$

Proof. Let $\mathcal{U}$ be the set of all random vectors $(U_1, \ldots U_J)$ such that

1. U_j, $j = 1, 2, \ldots, J$ are independent.

2. $0 \leq E[U_j] \leq \rho_j$ and $0 \leq U_j \leq c_j^*$ for all $j = 1, 2, , \ldots, J$.

All feasible vector arrival processes in $\mathcal{A}$ give steady–state rate variables that belong to $\mathcal{U}$. Let $(U_1, \ldots, U_J)$ be a random vector in $\mathcal{U}$. Let $U = U_1 + \cdots + U_J$ and $U^* = U_1^* + \cdots + U_J^*$. By Lemma 1 it suffices to show that

$$\frac{E[(U - C)^+ U_j]}{CE[U_j]} \leq \frac{E[(U^* - C)^+ U_j^*]}{CE[U_j^*]}. \tag{6.13}$$

Fix i, with $1 \leq i \leq J$, and consider the random vector $(\hat{U}_1, \ldots, \hat{U}_J)$ such that $\hat{U}_i = U_i^*$ and $\hat{U}_j = U_j$ for $j \neq i$. Note that $(\hat{U}_1, \ldots, \hat{U}_J) \in \mathcal{U}$. We first show that for each fixed j,

$$\frac{E[(U - C)^+ U_j]}{CE[U_j]} \leq \frac{E[(\hat{U} - C)^+ \hat{U}_j]}{CE[\hat{U}_j]}. \tag{6.14}$$

Consider the case $i \neq j$. Let $V = U - U_i - U_j$. Let $dF_V(\cdot)$ and $dF_{U_j}(\cdot)$ be the distribution functions for V and U_j. Noting that U_i, U_j and V are independent, we have

$$\begin{aligned} E[(U-C)^+U_j] &= E[(U_i+V+U_j-C)^+U_j] \\ &= \int_0^\infty \int_0^\infty E[(U_i+v+u-C)^+u]dF_V(v)dF_{U_j}(u) \end{aligned}$$

The function $f(x) = (x+v+u-C)^+u$ within the expectation is an increasing, convex function in x for each fixed v and u. Thus, because $U_i \leq_{icx} \hat{U}_i$ (e.g., see Proposition 1.5.1 in [71]), we have

$$E[(U_i+v+u-C)^+u] \leq E[(\hat{U}_i+v+u-C)^+u]$$

for all v and u. Combining the above two equations gives

$$E[(U-C)^+U_j] \leq E[(\hat{U}-C)^+\hat{U}_j],$$

which, when combined with $E[\hat{U}_j] = E[U_j]$, gives (6.14).

Now consider the case $i = j$. Let $W = U - U_i$. Using $U_i \leq c_i^*$, the independence of W and U_i, and the independence of W and $\hat{U}_i$, we obtain

$$\begin{aligned} \frac{E[(U-C)^+U_i]}{CE[U_i]} &= \frac{E[(W+U_i-C)^+U_i]}{CE[U_i]} \\ &\leq \frac{E[(W+c_i^*-C)^+]}{C}\frac{E[U_i]}{E[U_i]} \\ &= \frac{E[(W+c_i^*-C)^+]}{C}\frac{E[\hat{U}_i]}{E[\hat{U}_i]} \\ &= \frac{E[(W+c_i^*-C)^+\hat{U}_i]}{CE[\hat{U}_i]}. \end{aligned}$$

Also

$$\begin{aligned} E[(\hat{U}-C)^+\hat{U}_i] &= E[(W+\hat{U}_i-C)^+\hat{U}_i] \\ &= E[(W+c_i^*-C)^+\hat{U}_i]. \end{aligned}$$

Combining the above two equations gives (6.14) for $i = j$.

Thus (6.14) holds for all $i = 1, \ldots, J$. Therefore, starting with the original vector $(U_1, \ldots, U_J) \in \mathcal{U}$ we can replace U_1 with U_1^* and obtain a new vector in $\mathcal{U}$ such that (6.14) holds. Rename this new vector as $(U_1, \ldots, U_J)$. We can repeat the procedure, this

time replacing U_2 with U_2^*, and again obtaining a new vector in $\mathcal{U}$ such that (6.14) holds. Performing this procedure for all $i = 1, \ldots, J$ gives (6.13). ■

Using the fact that U_j^* is a Bernoulli random variable, we obtain from Theorem 7 the following expression for the bound of $P_{\text{loss}}(j)$:

$$\phi_j^* = \frac{E\left[(\sum_{k \neq j} U_k^* + c_j^* - C)^+\right]}{C} \tag{6.15}$$

We can compute these bounds directly by convolving the distributions of the independent random variables. An efficient approximate convolution algorithm is presented in [39]. We can also obtain an accurate approximation for the right–hand side of (6.15) by applying large deviation theory to the expectation in the numerator: To this end let

$$\mu_{U_k^*}(s) := \ln E[e^{sU_k^*}].$$

Note that $\mu_{U_k^*}(s)$ is the logarithm of the moment generating function for U_k^*. We define

$$U^* = \sum_{k \neq j} U_k^*.$$

Note that

$$\mu_{U^*}(s) = \sum_{k \neq j} \mu_{U_k^*}(s)$$

by the independence of the U_k^*'s. The large deviation (LD) approximation gives the following approximation for ϕ_j^* [62]

$$\frac{1}{Cs^{\star 2}\sqrt{2\pi \mu''_{U^*}(s^\star)}} e^{-s^\star(C - c_j^*) + \mu_{U^*}(s^\star)},$$

where $s^\star$ is the unique solution to

$$\mu'_{U^*}(s^\star) = C - c_j^*.$$

The LD approximation is known to be very accurate [62, 25, 13, 14, 57] and is also computationally very efficient. We use the LD approximation for the numerical studies in this chapter.

In summary, (6.15) is a simple expression for the worst–case loss probability ϕ_j^*; this simple expression involves the independent Bernoulli random variables $U_1^*, \ldots, U_J^*$, whose

distributions we know explicitly. The LD approximation for (6.15) is highly accurate and is easily calculated. For admission control, we advocate using the LD approximation to calculate ϕ_j^* and then verifying the QoS requirement, i.e., verifying in real–time whether $\phi_j^* \leq \epsilon_j$ for all $j = 1, \ldots, J$.

We note that Doshi [12] studies the arrival processes that maximize an aggregate loss ratio and an individual loss ratio. He discovers a number of anomalies of these loss criteria. With our bound $P_{\text{loss}}(j)$ (6.8) the loss is maximized by the Bernoulli random variables $U_1^*, \ldots, U_J^*$. Doshi focuses on the simple regulator and does not consider the smoothing of traffic.

6.3.1 The Optimal Smoother

Up to this point we have assumed that the smoother for each connection j consists of a single buffer that limits the peak rate of the smoother output to c_j^*. In this subsection we study more general smoothers, namely, smoothers that consist of a cascade of leaky buckets. The smoother for connection j, defined by a function $S_j(t)$, constrains the amount of traffic that can enter the network over any time interval. Specifically, if $B_j(t)$ is the amount of traffic leaving smoother j over the interval $[0, t]$, then $B_j(t)$ is required to satisfy

$$B_j(t+\tau) - B_j(\tau) \leq S_j(t) \quad \text{for all } t \geq 0,\ \tau \geq 0.$$

We assume throughout this section that the smoother functions are of the form

$$S_j(t) = \min_{1 \leq k \leq M_j} \{s_j^k + r_j^k t\} \tag{6.16}$$

with $r_j^1 > r_j^2 > \cdots > r_j^{M_j}$ and $0 = s_j^1 < s_j^2 < \cdots < s_j^{M_j}$. These piecewise linear, concave smoother functions can be easily implemented by a cascade of leaky buckets. The single–buffer smoother defined in Section 6.2 is a special case with $M_j = 1$, $s_j^1 = 0$ and $r_j^1 = c_j^*$.

We say that a set of smoothers $(S_1(t), ..., S_J(t))$ is *feasible* if the maximum delay incurred at smoother j is $\leq d_j$ for all $j = 1, \ldots, J$. By definition the set of smoothers $(c_1^* t, \ldots, c_J^* t)$ studied earlier is feasible. Now fix a feasible set of smoothers $(S_1(t), \ldots, S_J(t))$, and let the regulated traffic from the J connections pass through these smoothers. Let

$$\phi_j = \sup_{\mathcal{A}} \frac{E\left[(\sum_{k=1}^J U_k - C)^+ U_j\right]}{C \cdot E[U_j]} \tag{6.17}$$

be the associated worst–case loss probability. Recall that ϕ_j^* is the same worst–case loss probability but with the traffic passing through the set of smoothers $(c_1^*t, \ldots, c_J^*t)$. The proof of the following result is provided in Appendix A.

Theorem 8 *$\phi_j^* \leq \phi_j$ for all $j = 1, \ldots, J$. Thus the single–buffer smoothers with $c_j = c_j^*$ minimize the worst–case loss probability over all feasible sets of smoothers.*

It follows from Theorem 8 that the more complex smoothers consisting of cascaded leaky buckets do not increase the connection carrying capacity of the network. Thus without loss of performance, we may use the simple smoothers of the form $(c_1t, \ldots, c_Jt)$. Furthermore, Theorem 8 verifies the intuition that in order to maximize the admission region the smoother rates are as small as the delay constraints permit, that is, $c_j = c_j^*$ for $j = 1, \ldots, J$.

6.3.2 A Heuristic for Finding a Leaky Bucket Characterization of Prerecorded Sources

In this subsection we discuss how to obtain a good characterization $\mathcal{E}_j(t)$ of a source for a given restriction L_j on the number of leaky buckets. For any given characterization $\mathcal{E}_j(t)$ we use at the network edge a single–buffer smoother with rate c_j^* given by (6.5). Our goal is to find a characterization $\mathcal{E}_j(t)$ that has at most L_j slopes (i.e., L_j cascaded leaky buckets) and attempts to minimize both ρ_j and c_j^*. From Theorem 8 we know that minimizing ρ_j and c_j^* minimizes the worst–case loss probabilities, and thereby maximizes the connection–carrying capacity of the network.

We develop the heuristic for determining the characterization $\mathcal{E}_j(t)$ in the context of prerecorded sources. These sources include full–length movies, music video clips and educational material for video–on–demand (VoD) and other multimedia applications. It is well known how to compute the empirical envelope for prerecorded sources [31, 75, 40]. The empirical envelope gives the tightest bound on the amount of traffic that can emanate from a prerecorded source over any time interval. The empirical envelope is however not necessarily concave, and therefore we may not be able to be characterize it by a cascade of leaky buckets. However, applying the algorithms of Knightly *et al.* [75] or Grahams

Scan [5], we can compute the concave hull of the empirical envelope. The concave hull for connection–j traffic, denoted by $\mathcal{H}_j(t)$, takes the form

$$\mathcal{H}_j(t) = \min_{1 \le i \le K_j} \{\sigma_j^i + \rho_j^i t\}. \tag{6.18}$$

Here, K_j denotes the number of piecewise linear segments in the concave hull. Without loss of generality we may assume $\sigma_j^1 < \sigma_j^2 < \cdots < \sigma_j^{K_j}$ and $\rho_j^1 > \rho_j^2 > \cdots > \rho_j^{K_j}$.

The number of segments in the concave hull can be rather large. The *"Silence of The Lambs"* video segment used in our numerical experiments, for instance, has a concave hull consisting of 39 segments. This implies that 39 leaky bucket pairs are required to police the tightest concave characterization of the *"Silence of The Lambs"* video segment. Our goal is to find a more succinct characterization of prerecorded sources in order to simplify call admission control and traffic policing.

Suppose that a source is allowed to use L_j ($L_j < K_j$) leaky buckets to characterize its traffic. We now present a heuristic for the following problem: Given a source's concave hull $\mathcal{H}_j(t) = \min_{1 \le i \le K_j}\{\sigma_j^i + \rho_j^i t\}$ and the delay limit d_j, find L_j leaky buckets (out of the K_j leaky bucket pairs in the concave hull) that maximize the admission region.

We illustrate our heuristic for the case $L_j = 2$. For $L_j = 2$ the traffic constraint function takes the form

$$\mathcal{E}_j(t) = \min\{\sigma_j^{a_j} + \rho_j^{a_j} t,\ \sigma_j^{b_j} + \rho_j^{b_j} t\} \quad \text{with} \quad 1 \le a_j, b_j \le K_j, \tag{6.19}$$

where the indices a_j and b_j are yet to be specified. Our strategy is to first choose the leaky bucket that has the tightest bound on the average rate (i.e., minimize ρ_j), and then choose another leaky bucket which minimizes the smoother rate c_j^*. Let r_j^{ave} denote the average rate of the prerecorded source. We found in our numerical experiments that some of the leaky bucket pairs in the concave hull (particularly those with high indices) may have slopes $< r_j^{\text{ave}}$. We set $b_j = \max\{i : \rho_j^i \ge r_j^{\text{ave}},\ 1 \le i \le K_j\}$. In words, we use the highest indexed leaky bucket with a slope larger than r_j^{ave} to specify the sources' average rate.

In order to find the leaky bucket indexed by a_j we consider all leaky buckets $(\sigma_j^i,\ \rho_j^i)$ with $1 \le i < b_j$. We compute the smoother rates obtained by combining each of the leaky buckets $(\sigma_j^i,\ \rho_j^i),\ 1 \le i < b_j$ with the leaky bucket $(\sigma_j^{b_j}, \rho_j^{b_j})$ and select the index i that

gives the smallest smoother rate — and thus the largest admission region. More formally, let $c_j^{*i},\ 1 \le i < b_j$, denote the minimal smoother rate for traffic with regulator function $\mathcal{E}_j(t) = \min\{\sigma_j^i + \rho_j^i t,\ \sigma_j^{b_j} + \rho_j^{b_j} t\}$ and delay bound d_j. By (6.5) we have

$$c_j^{*i} \quad = \quad \max_{t \ge 0} \frac{\min\{\sigma_j^i + \rho_j^i t,\ \sigma_j^{b_j} + \rho_j^{b_j} t\}}{d_j + t}.$$

We can obtain a more explicit expression for c_j^{*i}. Since

$$\min\{\sigma_j^i + \rho_j^i t,\ \sigma_j^{b_j} + \rho_j^{b_j} t\} = \begin{cases} \sigma_j^i + \rho_j^i t & \text{for } 0 \le t \le t_i \\ \sigma_j^{b_j} + \rho_j^{b_j} t & \text{for } t \ge t_i \end{cases}$$

with $t_i = (\sigma_j^{b_j} - \sigma_j^i)/(\rho_j^i - \rho_j^{b_j})$, we have

$$c_j^{*i} \quad = \quad \max\left[\max_{0 \le t \le t_i} \frac{\sigma_j^i + \rho_j^i t}{d_j + t},\ \max_{t \ge t_i} \frac{\sigma_j^{b_j} + \rho_j^{b_j} t}{d_j + t} \right].$$

The expressions inside the max[·] can be further simplified. It can be shown that

$$\max_{0 \le t \le t_i} \frac{\sigma_j^i + \rho_j^i t}{d_j + t} = \begin{cases} \frac{\sigma_j^i}{d_j} & \text{if } d_j \le \frac{\sigma_j^i}{\rho_j^i} \\ \frac{\sigma_j^i + \rho_j^i t_i}{d_j + t_i}, & \text{if } d_j \ge \frac{\sigma_j^i}{\rho_j^i} \end{cases}$$

and

$$\max_{t \ge t_i} \frac{\sigma_j^{b_j} + \rho_j^{b_j} t}{d_j + t} = \begin{cases} \frac{\sigma_j^i + \rho_j^i t_i}{d_j + t_i}, & \text{if } d_j \le \frac{\sigma_j^{b_j}}{\rho_j^{b_j}} \\ \frac{\sigma_j^{b_j}}{d_j} & \text{if } d_j \ge \frac{\sigma_j^{b_j}}{\rho_j^{b_j}}. \end{cases}$$

We set the smoother rate to $\min_{1 \le i < b_j} c_j^{*i}$ and set a_j to the index that attains this minimum.

We now briefly discuss how to find the optimal regulator function consisting of 3 or more leaky buckets. First, note that there are $\begin{pmatrix} b_j - 1 \\ L_j - 1 \end{pmatrix}$ combinations of leaky bucket pairs to consider. This can be computationally prohibitive. The heuristic can be sped up by considering only regulator functions consisting of $L_j - 1$ consecutive leaky buckets of the concave hull and the leaky bucket $(\sigma_j^{b_j}\ \rho_j^{b_j})$. In the case $L_j = 3$, for instance, we compute the minimal smoother rates only for the regulator functions $\mathcal{E}_j(t) = \min\{\sigma_j^i + \rho_j^i t,\ \sigma_j^{i+1} + \rho_j^{i+1} t,\ \sigma_j^{b_j} + \rho_j^{b_j} t\}$ with $1 \le i < b_j - 1$. This speed–up of the heuristic can produce a suboptimal regulator function. Our numerical experiments (see Section 6.4), however, indicate that it works surprisingly well.

6.3.3 Interaction between Application and Network

In this subsection we discuss how the responsibilities of smoothing, call admission control and traffic policing can be shared by the application and the network. Call admission control is the responsibility of the network. Before accepting a new connection, the network has to ensure that the QoS requirements continue to hold for all established connections and the new connection. Policing is also a network responsibility. The network edge has to police all established connections in order to ensure that all connections comply with their respective regulator function advertised at connection establishment. While call admission control and traffic policing are responsibilities of the network, smoothing can be performed by either the application or the network. We refer to the case where the application performs the smoothing and sends the smoothed traffic to the network edge as *application smoothing*. The case where the application sends its unsmoothed traffic to the network edge and the network edge performs the smoothing is referred to as *network smoothing*.

With application smoothing the application internally smoothes its traffic. Based on the regulator function of its traffic and the maximum delay it can tolerate, the application finds the minimum smoother rate by applying (6.5). Since the smoothing is done by the application, there is no need to reduce the number of leaky buckets used to characterize the traffic by applying the heuristic outlined in Section 6.3.2. Instead, the concave hull of a prerecorded source is used directly for dimensioning its smoother. The application advertises the regulator function $\mathcal{E}_j(t) = \min\{c_j^* t,\ \sigma_j^{L_j} + \rho_j^{L_j} t\}$ and the delay bound $d_j = 0$ to the network. We remark that this dual leaky bucket regulator function has been adopted by the ATM Forum [19] and is being proposed for the Internet [69]. The network does not have to be aware of the smoothing done by the application. The network edge dimensions its own smoother based on $\mathcal{E}_j(t)$ and $d_j = 0$. Since $d_j = 0$ the networks' smoother degenerates to a server with rate c_j^* preceded by a buffer of size zero.

With network smoothing the application advertises its regulator function and maximum tolerable delay to the network. Prerecorded sources apply the heuristic of Section 6.3.2 when the network restricts the number of leaky buckets to a number smaller than the number of segments in the concave hull. The network edge dimensions the smoother based

d_{lambs} sec.	a_{lambs}	$\sigma_{\text{lambs}}^{a_{\text{lambs}}}$ kByte	$\rho_{\text{lambs}}^{a_{\text{lambs}}}$ kbit/sec	c_{lambs}^{*} kbit/sec	$t_{a_{\text{lambs}}}$ sec.	t_{c^*} sec.
0	1	0	3474.8	3474.8	7.735	7.735
0.042	2	13.3	939.3	2535.5	34.435	10.858
0.125	2	13.3	939.3	939.0	34.435	34.596
0.250	4	23.5	802.2	801.9	42.256	42.594
0.500	8	43.5	711.0	710.8	49.610	50.318
1.000	10	69.9	676.9	674.7	52.768	54.216

Table 6.1: Parameters of the optimal leaky bucket characterization with 2 leaky buckets as a function of the delay bound for the lambs trace. The average rate is characterized by the 34th leaky bucket, i.e., $b_{\text{lambs}} = 34$, with parameters $\sigma_{\text{lambs}}^{b_{\text{lambs}}} = 3,157.8$ kByte and $\rho_{\text{lambs}}^{b_{\text{lambs}}} = 208.8$ kbit/sec for all delay bounds.

on the regulator function and delay bound supplied by the application. Call admission control is based on the assumption of worst–case on–off traffic at the smoother output. The network edge polices the applications' traffic before it enters the smoother and drops violating traffic.

6.4 Numerical Experiments

In this section we evaluate the smoothing/bufferless multiplexing scheme proposed in this chapter using traces from MPEG–1 encoded movies (see Table 3.1). In all experiments we consider a single bufferless multilexer which feeds into a 45 Mbps link. Let x_n, $n = 1, \ldots, N$, denote the size of the nth frame in bits. We convert the discrete frame size trace to a fluid flow by transmitting the nth frame at rate $x_n F$ over the interval $[n - 1/F, n/F]$.

We first evaluate the heuristic of Section 6.3.2. We compute the empirical envelope and the concave hull of each trace using the algorithms of Knightly *et al.* [75]. Based on the concave hull of each video we compute the minimal smoother rate c_j^*. We also apply the heuristic of Section 6.3.2 to the concave hull in order to find the optimal leaky bucket characterization with 2 and more leaky buckets. (We apply the speed–up described in Section 6.3.2 for the leaky bucket characterizations with 3 or more leaky buckets.)

The heuristic of Section 6.3.2 produced the optimal leaky bucket characterizations given in Table 6.1 for the lambs trace. The table gives the index a_{lambs} and the parameters of the

leaky bucket $(\sigma_{\text{lambs}}^{a_{\text{lambs}}},\ \rho_{\text{lambs}}^{a_{\text{lambs}}})$ for various delay bounds. The average rate is characterized by the 34th leaky bucket in the concave hull, i.e., $b_{\text{lambs}} = 34$, for all delay bounds. The table also gives the minimal smoother rates for the various delay bounds. The table furthermore lists the time $t_{a_{\text{lambs}}}$. Recall from Section 6.3.2 that $t_{a_{\text{lambs}}}$ is the time where the slopes of the leaky buckets $(\sigma_{\text{lambs}}^{a_{\text{lambs}}},\ \rho_{\text{lambs}}^{a_{\text{lambs}}})$ and $(\sigma_{\text{lambs}}^{b_{\text{lambs}}},\ \rho_{\text{lambs}}^{b_{\text{lambs}}})$ intersect. We also give the time t_{c^*} defined by $c_j^* t_{c^*} = \sigma_j^{b_j} + \sigma_j^{b_j} t_{c^*}$, that is, t_{c^*} is the time where the leaky buckets $(0,\ c_j^*)$ and $(\sigma_j^{b_j},\ \rho_j^{b_j})$ intersect. The traffic at the smoother output is thus characterized by the regulator function

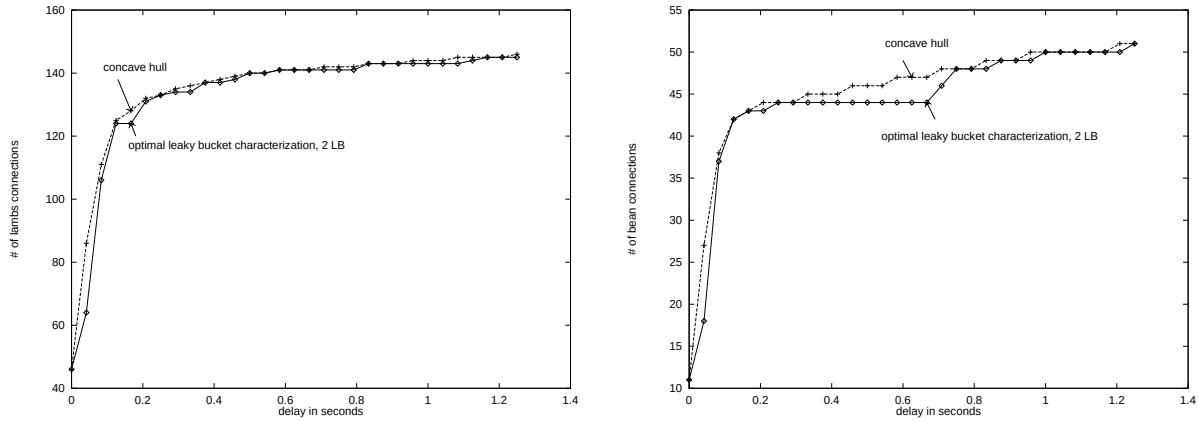
$$\mathcal{E}_j(t) = \min\{c_j^* t,\ \sigma_j^{b_j} + \rho_j^{b_j} t\} = \begin{cases} c_j^* t & \text{for } 0 \le t \le t_{c^*} \\ \sigma_j^{b_j} + \rho_j^{b_j} t & \text{for } t \ge t_{c^*} \end{cases}$$

Intuitively, t_{c^*} is the maximum burst length of the smoother output, that is, the smoother can burst at rate c_j^* for at most t_{c^*} seconds.

For a delay bound of zero, the smoother rate is set to the rate of the first leaky bucket, i.e., the peak rate of the trace. For $d_{\text{lambs}} = 0.042$ sec $(= 1/F)$ the trace is characterized by the 2nd and 34th leaky bucket of the concave hull $(a_{\text{lambs}} = 2,\ b_{\text{lambs}} = 34)$. Note that $d_{\text{lambs}} < \sigma_{\text{lambs}}^{a_{\text{lambs}}}/\rho_{\text{lambs}}^{a_{\text{lambs}}}$ in this case and $c_{\text{lambs}} = \sigma_{\text{lambs}}^{a_{\text{lambs}}}/d_{\text{lambs}}$. For $d_{\text{lambs}} \ge 0.125$ sec we have $d_{\text{lambs}} > \sigma_{\text{lambs}}^{a_{\text{lambs}}}/\rho_{\text{lambs}}^{a_{\text{lambs}}}$ and $c_{\text{lambs}}^* = (\sigma_j^{a_{\text{lambs}}} + \rho_j^{a_{\text{lambs}}} t_{a_{\text{lambs}}})/(d_j + t_{a_{\text{lambs}}})$.

Assuming worst–case on–off traffic, the smoother outputs are statistically multiplexed onto the bufferless link as discussed in the previous sections. We set $\epsilon_j = 10^{-7}$ for all connections. In Figure 6.2 we plot the number of admissible video connections as a function of the delay bound. The graph gives the number of admissible video connections when the videos are characterized by the concave hull or the optimal leaky bucket characterization with 2 leaky buckets. We observe from the plots that the optimal leaky bucket characterization with 2 leaky buckets admits almost as many video connections as the more accurate concave hull characterization. The curves for 3 or more leaky buckets coincide with the curve for the concave hull.

In the next experiment we compare the admission region of our approach with the admission region obtained with the deterministic admission control condition of Knightly *et al.* [75]. Note that the deterministic approach of Knightly *et al.* is lossless and guarantees that no bit is delayed by more than the prespecified delay limit in the multiplexer buffer.



(a) lambs (b) bean

Figure 6.2: Number of video connections as a function of the delay bound. The videos are characterized by the concave hull or the optimal leaky bucket characterization with 2 leaky buckets. The bound on the loss probability is 10^{-7}.

Our approach, on the other hand, exploits the independence of traffic emanating from the J connections. The videos are passed through simple smoothers with $c_j = c_j^*$. The smoother outputs — assuming worst–case on–off traffic — are then statistically multiplexed onto the bufferless link, as discussed in the preceding sections. We set $\epsilon_j = 10^{-7}$ for all connections. Losses this small have essentially no impact on the perceived video quality and can be easily hidden by error concealment techniques [43].

In Figure 6.3 we plot the number of admissible lambs connections as a function of the delay bound. The graph gives the number of lambs connections that are admitted with the our approach (RRR) when 2 or 3 leaky buckets (LB) are used to characterize the video trace. As we just saw in Figure 6.2 the optimal leaky bucket characterization with 3 leaky buckets admits as many connections as the concave hull, the most accurate, concave characterization of the video; using more leaky buckets does not increase the admission region. We also plot the number of lambs connections that are admitted with the approach of Knightly *et al.* (KLZ) when 2, 3, 8 or 16 leaky buckets are used to characterize the trace. We observe that for delays on the order of .5 seconds or more, the number of admissible connections significantly increases as the number of leaky buckets used to describe the trace increases. The approach of Knightly *et al.* thus greatly benefits from a more accurate characterization of the video — achieved by more leaky buckets.

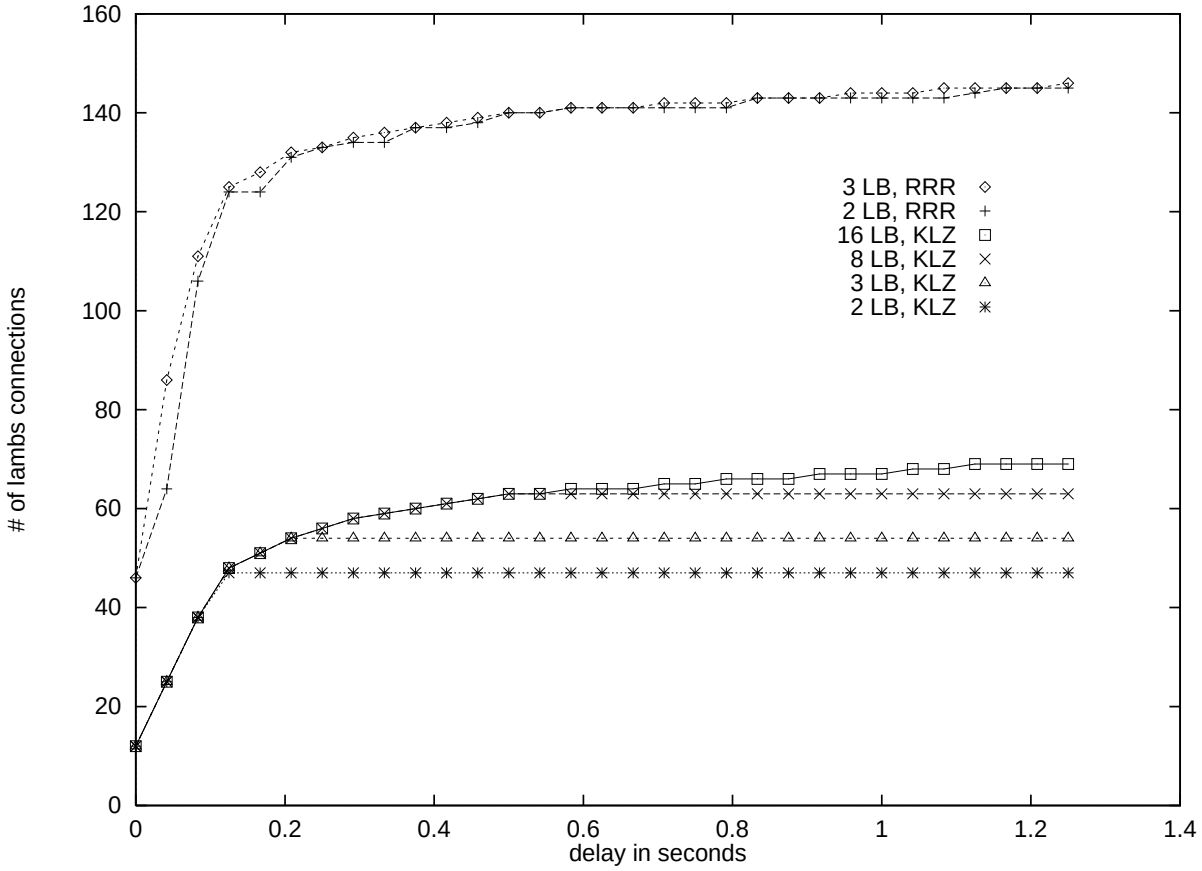


Figure 6.3: Number of lambs connections as a function of the delay bound and the number of leaky buckets (LB). Plots shown are for Knightly *et al.* (KLZ) [75] and our approach(RRR). The bound on the loss probability is 10^{-7}.

The main result of this experiment, however, is that our approach allows for more than twice the number connections than does the approach of Knightly *et al.* For example, for a delay bound of 1.1 seconds, Knightly *et al.* admit 69 connections (= 29.6 % average link utilization) with 16 leaky buckets while our approach admits 146 connections (= 62.7 % average link utilization) with 3 leaky buckets. We obtain this dramatic increase in the admission region by exploiting the independence of the sources and allowing for a small loss probability.

In Figure 6.4 we consider multiplexing two different movies, beans and lambs, each with its own delay constraint. We again assume a 45 Mbps link. We use delay bounds of d_{lambs} = 125 ms or 1.25 seconds and d_{bean} = 125 ms or 1.25 seconds, giving four combinations. Both videos are characterized by 3 leaky buckets. We assume that both video connections have

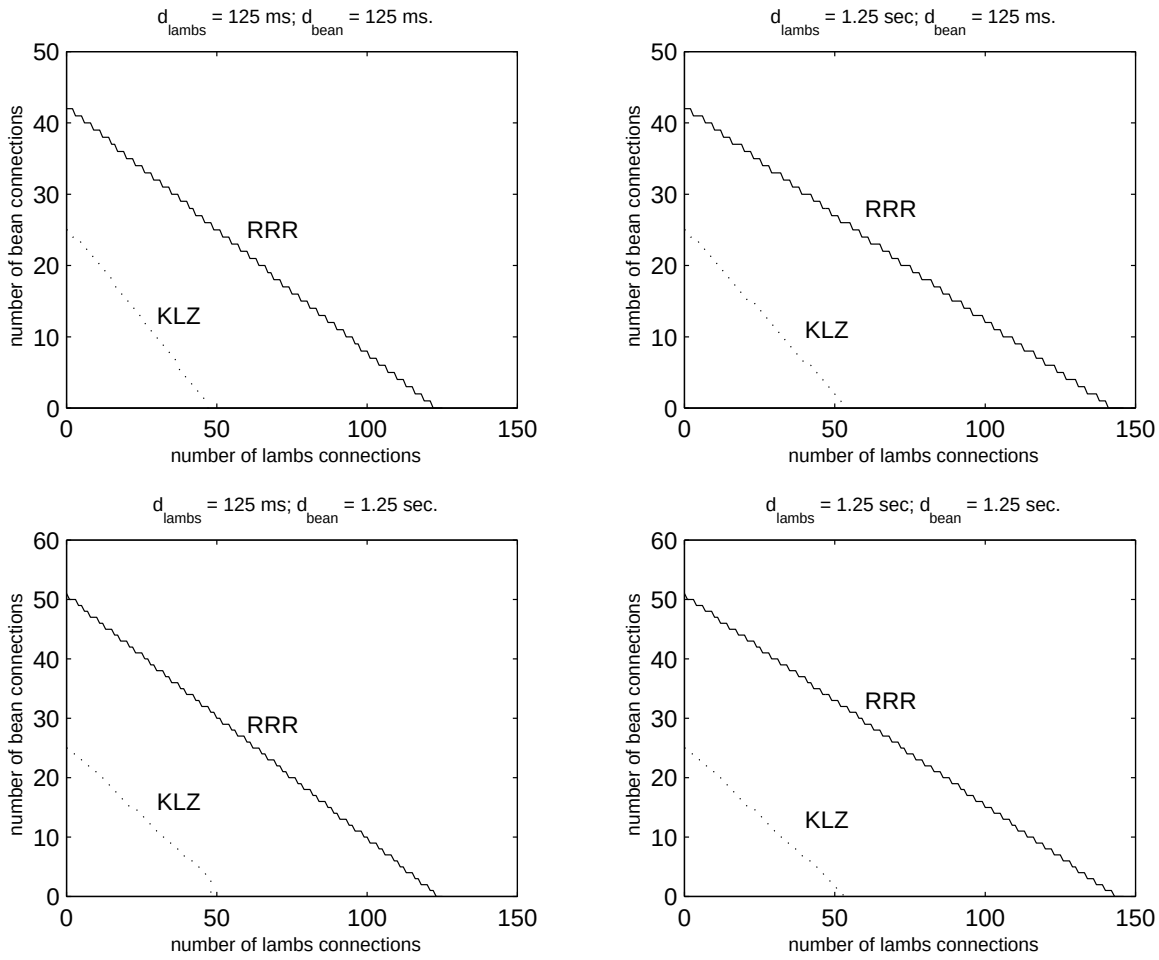
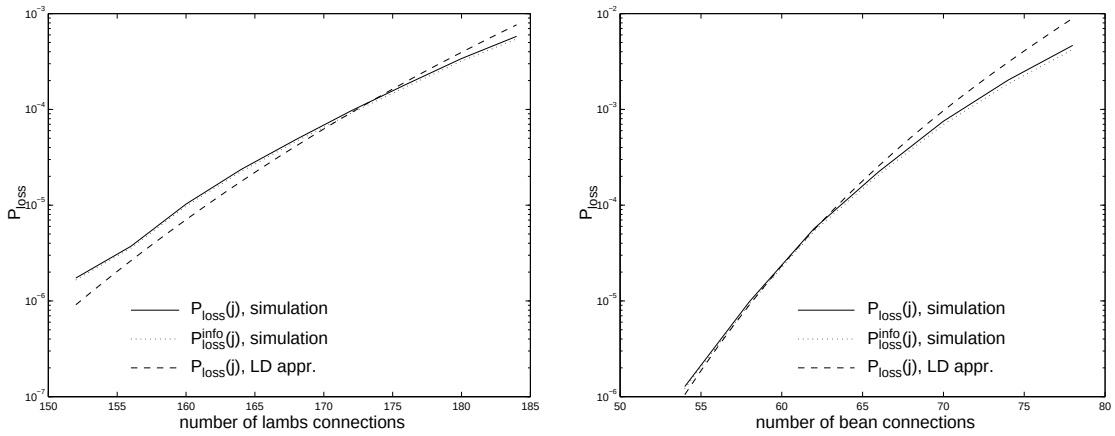


Figure 6.4: Admission region for the multiplexing of lambs and bean connections over a 45 Mbps link.

the QoS requirement that the fraction of traffic that is delayed by more than the imposed delay limit is less than 10^{-7}. For the Knightly *et al.* plot we use Earliest Deadline First (EDF) scheduling. We see that for all four cases, the admission region for our approach is dramatically larger.

In Figure 6.5 we compare the actual loss probability, $P_{\text{loss}}^{\text{info}}(j)$ given by (6.7) with our bound for loss probability, $P_{\text{loss}}(j)$, given by (6.8). We obtain $P_{\text{loss}}^{\text{info}}(j)$ and $P_{\text{loss}}(j)$ by simulation, and assume worst–case on–off traffic. We also verify the accuracy of the large deviation approximation for $P_{\text{loss}}(j)$. In Figure 6.5 we plot the loss probabilities as a function of the number of connections being multiplexed over a 45 Mbps link. We consider the scenario where the videos have a delay bound of 1 second and are characterized by



(a) lambs (b) bean

Figure 6.5: The simulation verifies that the bound on the loss probability $P_{\text{loss}}(j)$ tightly bounds the actual loss probability $P_{\text{loss}}^{\text{info}}(j)$. The plots further confirm the accuracy of the Large Deviation (LD) approximation. We use a delay bound of 1 second and characterize the videos by 3 leaky buckets. The link rate is 45 Mbps. The plots give the loss probability as a function of the number of ongoing connections.

3 leaky buckets. We observe that the bound on the loss probability $P_{\text{loss}}(j)$ (solid line) tightly bounds the actual loss probability $P_{\text{loss}}^{\text{info}}(j)$ (dotted line). We also observe that the LD approximation (dashed line) closely approximates the simulation results.

6.5 Comparison with Buffered Statistical Multiplexing

The numerical results of the previous section show that our approach allows for dramatically more connections than buffered deterministic multiplexing. In this section we briefly consider buffered multiplexing with an allowance of small loss probabilities, which we refer to as *buffered statistical multiplexing*. Consider the buffered analogy of the single–link bufferless system studied in Section 6.3. The link has capacity C and is preceded by a finite buffer of capacity B. Let the same J connections arrive to this system; specifically the J connections are independent and the jth connection is regulated by a given regulator function $\mathcal{E}_j(t)$. The traffic from the J connections passes directly into the buffered multiplexer, i.e., the traffic is not pre–smoothed before arriving at the buffer. This buffered system is illustrated in Figure 6.6. Assuming that traffic is served FIFO, the maximum delay in this

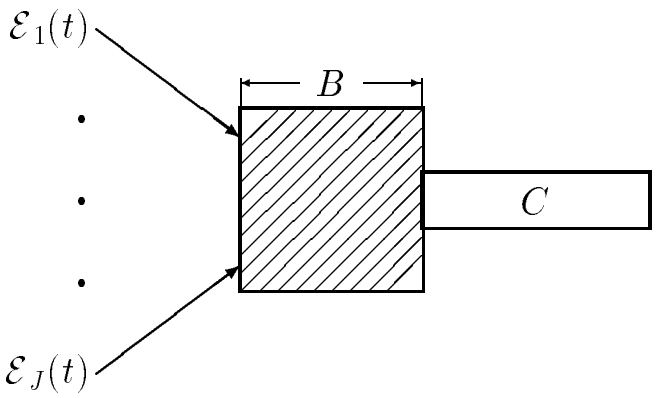


Figure 6.6: The traffic of connection j is characterized by the regulator function $\mathcal{E}_j(t)$ and fed directly, i.e. unsmoothed, into a buffered multiplexer.

system is $d = B/C$. Suppose that the buffer overflow probability is constrained to be no greater than ϵ.

It is a difficult and challenging problem to accurately characterize the admission region for a buffered multiplexer which multiplexes regulated traffic and which allows for statistical multiplexing. In the seminal paper, Elwalid *et al.* in [14] consider the buffered multiplexer for the *special case of regulators with two leaky buckets*, i.e., for $\mathcal{E}_j(t) = \min\{\rho_j^1 t,\ \sigma_j + \rho_j t\}$. (In our numerical comparisons, we extend their theory to the case of multiple cascaded leaky buckets.) In order to make the buffered multiplexer mathematically tractable they assign each connection its own virtual buffer/trunk system. Each virtual buffer/trunk system is allocated buffer $b_{0,j}$ and bandwidth $e_{0,j}$. The allocations are based on the buffer and bandwidth resources (B and C, respectively) and on the regulator parameters (ρ_j, ρ_j^1, and σ_j) for the input traffic. It turns out that the bandwidth $e_{0,j}$ is exactly the c_j^* obtained by setting $d_j = d = B/C$ in (6.4). After some analysis Elwalid *et al.* obtain the following bound on the fraction of time during which loss occurs at the buffered multiplexer:

$$P_{\text{loss}}^{\text{EMW}} = P(U_1^* + \cdots + U_J^* \leq C).$$

where $U_1^*, \ldots, U_J^*$ are exactly the same random independent random variables that occur in Theorem 7. (To calculate the associated $c_1^*, \ldots, c_J^*$, set $d_j = d = B/C$ for each connection j.)

This observation indicates that the bufferless system of this chapter has remarkably similarities with the buffered system in [14]. Specifically, for a fixed maximum delay d in the buffered system, we can design a bufferless system with pre–smoothers which has the same maximum delay and which has an admission region based on the same set of independent random variables $U_1^*, \ldots, U_J^*$. The pre–smoothers essentially implement the virtual buffer/trunk systems introduced by Elwalid *et al.* For a maximum loss probability of ϵ the admission region for the buffered multiplexer is defined by

$$P(U_1^* + \cdots + U_J^* \leq C) \leq \epsilon$$

whereas the admission region for the bufferless system is

$$\frac{E[(\sum_{k=1}^{J} U_k^* - C)^+ U_j^*]}{C \cdot E[U_j^*]} \leq \epsilon.$$

Although these admission regions are different, they are based on exactly the same independent random variables $U_1^*, \ldots, U_J^*$. The difference in these admission regions is an artifact of using two different notions of loss probability: while in this chapter we use "fraction of traffic lost", the paper [14] uses "the fraction of time during which loss occurs". If the same notions of loss were used, then the admission regions would be identical. Figure 6.7 gives the number of lambs connections that are admitted with the approach of Elwalid *et al.* (EMW) [14] and our approach (RRR) when 3 leaky buckets are used to characterize the trace. We assume a 45 Mbps link and set $\epsilon_j = 10^{-7}$ for all connections.

Thus, our bufferless system has essentially the same admission region as the buffered system in [14] for a fixed worst–case delay d and loss probability ϵ. While being no more difficult to perform call admission, we believe that the bufferless system has some important advantages over the buffered system: (i) no buffer is needed at the multiplexer (for packetized traffic, a relatively small buffer would be needed); (ii) the bufferless approach allows for a per–connection QoS requirement, whereas the buffered system imposes the same QoS requirement on all connections; and (iii), perhaps most importantly, networks are quite tractable for bufferless links, as we can reasonably approximate a connection's traffic at the output of the multiplexer as being identical to its traffic at the input to the multiplexer.

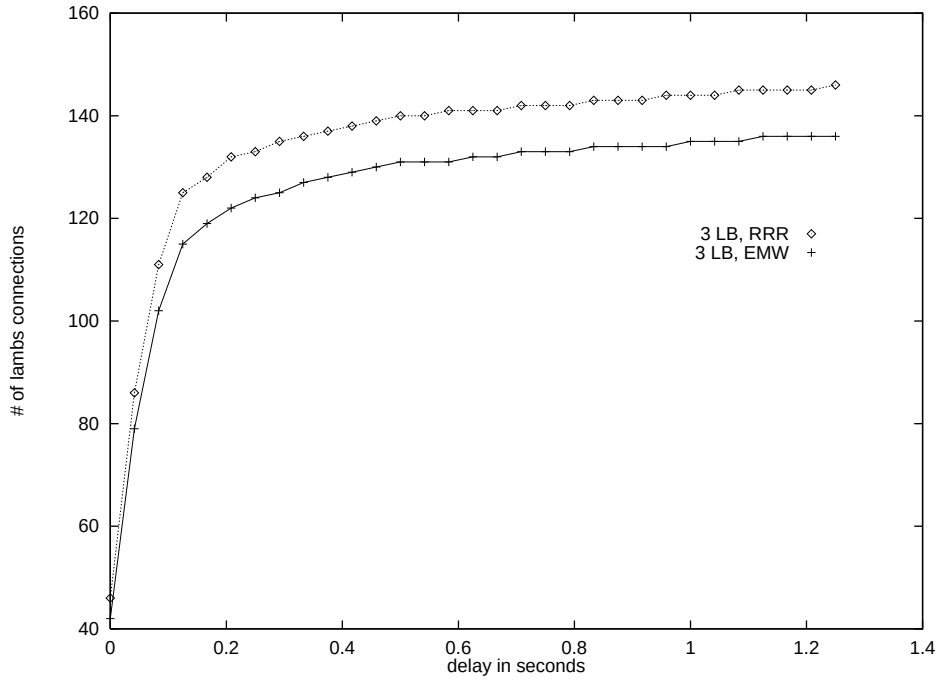


Figure 6.7: Number of lambs connections as a function of the delay bound. The lambs video is described by 3 leaky buckets. Plots shown are for Elwalid *et al.* (EMW) [14] and our approach(RRR). The difference in the number of admissible connections is due to the different notions of loss probability. Elwalid *et al.* use "fraction of time during which loss occurs" while we use "fraction of traffic lost".

On the other hand, the buffered system does have some advantages over the bufferless system. First, although both systems have the same worst–case delay, the buffered system will have a lower average delay. Second, the admission region of [14] can be increased using the techniques in [42] and [54] (at the expense of a much more complicated admission procedure). Because multimedia applications are typically designed for a delay bound, and because the aforementioned increase in admission region is typically small, we feel that the advantages of the bufferless approach outweigh the advantages of the buffered approach.

6.6 Conclusion

In this chapter we have considered traffic management for multimedia networking applications which permit a small amount of loss and some bounded delay. We have argued that it is preferable to smooth the traffic at the ingress and to perform bufferless statistical multiplexing within the node than to use shared–buffer multiplexing. For our scheme we have

determined the worst–case traffic and have outlined an admission control procedure based on the worst–case traffic. We have also explicitly characterized the optimal smoother.

As pointed out in Section 6.3.3 the smoothing can be performed by either the network (at the network edge) or by the applications themselves. If the applications perform the smoothing, then an application should smooth the traffic as much as permitted by the delay constraint, and the network should offer a service to the application which guarantees queueing–free delays (delays only due to propagation and nodal processing) and allows the application to specify a maximum tolerable loss rate. The network node should perform statistical multiplexing in order to maximize its connection–carrying capacity. To guarantee QoS, admission control should suppose that the traffic is adversarial to the extent permitted by the regulators and smoothers.

Chapter 7

Conclusion

In this thesis we have established a framework for the management of prerecorded traffic in broadband networks. In Chapters 2 through 4 we have shown how to exploit the special properties of prerecorded traffic — predictability and prefetchability — in order to allow for efficient transport. Our protocols afford a connection probabilistic QoS guarantees while efficiently utilizing network resources.

Even though we have defined our prefetching protocols for discrete–time video connections that display a new frame roughly every 40 msec our protocols can easily be adapted for continuous–time traffic such as audio traffic. One option is to employ a packet–based prefetching protocol as described in Section 3.6. With this packet–based protocol the amount of prefetched traffic in a clients' buffer in measured in the number of prefetched packets.

Alternatively, we may divide time into fixed–length slots of, say, 40 msec and prefetch the continuous traffic slot by slot. With this scheme the amount of prefetched traffic in a clients' buffer is measured in the number of prefetched traffic slots and the frame–based prefetching protocol defined in Section 3.3 is modified in the obvious manner.

There are several avenues for future research. It is important to evaluate the prefetching protocols with a greater variety of multimedia traffic traces. These experiments require traces that give an accurate account of the bandwidth requirements of long video sequences. Unfortunately, there are currently only few such traces publicly available. Furthermore, it would be useful to extend the decentralized prefetch protocols such that they can tolerate

longer feedback delays. This would allow the servers to be distributed deeper into the network. Its is also worthwhile to develop a hybrid scheme that combines centralized and decentralized prefetching. Such a hybrid scheme is useful when each video server feeds not one but many clients and a number of servers share a common link.

In Chapters 5 and 6 we have studied traffic management schemes for regulated traffic. Regulated traffic is not necessarily prerecorded. The concepts studied here apply also to live sources as long as they can supply a regulator function. Specifying a regulator function for a live source is, however, a challenging problem. Finding a tight regulator function of a prerecorded source, on the other hand, is a straightforward task. The concept of regulated traffic is very popular within the standards bodies of the ATM [19] and Internet [69] communities and has therefore received a great deal of attention. Our research indicates that it is possible to efficiently transmit prerecorded multimedia traffic within the regulated traffic framework. We show that smoothers at the network edge and statistical multiplexing with small loss probabilities are key elements of such a transmission scheme. The research in Chapter 6 is restricted to a single node. We are currently extending the smoothing/bufferless multiplexing scheme to a more general multi–hop network.

The call admission rule in Chapter 6 assumes that the smoothers feed adversarial on–off traffic into the network. The advantage of this approach is that it allows us to guarantee the QoS negotiated at call set–up time as long as all sources transmit within the limits of the leaky bucket constraints that they declared at call set–up. These leaky bucket constraints are easy to police. We are therefore able to prevent traffic that violates the leaky bucket constraints from entering the network and are thus able to guarantee the agreed upon QoS. The disadvantage of this "adversarial" approach is that it can be overly conservative and lead to low network utilizations. An alternative approach is to base call admission on the actual traffic at the smoother outputs. This approach works as follows. We design the smoother of connection j such that no bit is delayed by more than d_j. The design described in Section 6.2 achieves exactly that. Next, we simulate passing the prerecorded video through the smoother and find the frame size trace at the smoother output. We then apply the methodology developed in Chapter 2 and base the call admission decision on the smoothed frame size trace at the smoother output. We refer to this approach as the *non–adversarial* approach. The non–adversarial approach takes advantage of the predictability

of prerecorded traffic. It is less conservative than the adversarial approach and gives higher link utilizations. We have conducted some preliminary numerical investigations of the non–adversarial approach that are very promising. One disadvantage of the non–adversarial approach is that a source has to specify its traffic with a very verbose frame size trace. It is also not possible to police a source in a straightforward manner. Preventing a malicious source from deteriorating everyones QoS is an unsolved challenge.

Appendix A

Proof of Theorem 8

The purpose of this appendix is to provide a proof for Theorem 8. But first we need to establish two lemmas.

Lemma 2 *A necessary condition for* $(S_1(t), \ldots, S_J(t))$ *to be feasible is* $r_j^1 \geq c_j^*$ *for all* $j = 1, \ldots, J$.

Proof. From [7, 8, 23] the maximum delay at smoother j is

$$\tilde{d}_j = \max_{t \geq 0} \{ \max_{1 \leq k \leq M_j} \frac{\mathcal{E}_j(t) - s_j^k}{r_j^k} - t \}. \tag{8.1}$$

Suppose $r_j^1 < c_j^*$ for some $j = 1, \ldots, J$. Because $s_j^k \geq 0$ and $r_j^k \leq r_j^1$ for all k, it follows from (8.1) that

$$\tilde{d}_j \geq \max_{t \geq 0} \{ \frac{\mathcal{E}_j(t)}{r_j^1} - t \}. \tag{8.2}$$

And because, by assumption, $r_j^1 < c_j^*$, it follows from (8.2) that

$$\tilde{d}_j > \max_{t \geq 0} \{ \frac{\mathcal{E}_j(t)}{c_i^*} - t \} = d_j,$$

where the last equality follows from (6.4). ■

Lemma 3 *There exists a stochastic vector arrival process in* $\mathcal{A}$ *that produces the steady-state rate variables* $\tilde{U}_1, \ldots, \tilde{U}_J$ *with* $\tilde{U}_j$ *having distribution*

$$\tilde{U}_j = \begin{cases} \min(r_j^1, \rho_j^1) & \textit{with probability } \frac{\rho_j}{\min(r_j^1, \rho_j^1)} \\ 0 & \textit{with probability } 1 - \frac{\rho_j}{\min(r_j^1, \rho_j^1)} \end{cases}$$

at the smoother outputs.

	$\rho_j^1 \geq r_j^1$		$\rho_j^1 < r_j^1$	
	$\mathcal{E}_j(t_j) \geq S_j(\delta_j)$	$\mathcal{E}_j(t_j) < S_j(\delta_j)$	$\mathcal{E}_j(t_j) \geq S_j(\delta_j)$	$\mathcal{E}_j(t_j) < S_j(\delta_j)$
T_j	$S_j(\delta_j)/\rho_j$	$\mathcal{E}_j(t_j)/\rho_j$	$S_j(\delta_j)/\rho_j$	$\mathcal{E}_j(t_j)/\rho_j$
t_{on_j}	$S_j(\delta_j)/\rho_j^1$	t_j	$S_j(\delta_j)/\rho_j^1$	t_j
τ_{on_j}	δ_j	$\mathcal{E}_j(t_j)/r_j^1$	$S_j(\delta_j)/\rho_j^1$	t_j

Table 8.1: On–times and periods of $\tilde{b}_j(t)$ and $\tilde{o}_j(t)$.

Proof. For each $j = 1, \ldots, J$, let $t_j = \sigma_j^2/(\rho_j^1 - \rho_j^2)$ and $\delta_j = s_j^2/(r_j^1 - r_j^2)$. At $t = t_j$ the slope of $\mathcal{E}_j(t)$ changes form ρ_j^1 to $\rho_j^2 < \rho_j^1$. Consequently, $\mathcal{E}_j(t_j) = \rho_j^1 t_j$ is the maximum size burst that can be transmitted at rate ρ_j^1, provided successive maximum size bursts are spaced at least $\mathcal{E}_j(t_j)/\rho_j - t_j$ apart. Similarly, at $t = \delta_j$ the slope of $S_j(t)$ changes form r_j^1 to $r_j^2 < r_j^1$. Consequently, $S_j(\delta_j) = r_j^1 \delta_j$ is the maximum size burst the smoother can pass at rate r_j^1, provided successive maximum size bursts are spaced at least $S_j(\delta_j)/r_j^{M_j} - \delta_j$ apart.

Let $\tilde{b}_j(t)$ be a deterministic periodic function such that

$$\tilde{b}_j(t) = \begin{cases} \rho_j^1 & 0 \leq t < t_{\text{on}_j} \\ 0 & t_{\text{on}_j} \leq t \leq T_j \ . \end{cases}$$

with on–time t_{on_j} and period T_j given in Table 8.1. Also, let $\theta_1, \ldots, \theta_J$ be independent random variables with θ_j uniformly distributed over $[0, T_j]$ and define the jth stochastic arrival process as

$$\tilde{A}_j(t) = \int_0^t \tilde{b}_j(s + \theta_j) ds.$$

Thus each component arrival process $(\tilde{A}_j(t),\ t \geq 0)$ is generated by a periodic on-off source; the jth process has peak rate ρ_j^1 and average rate ρ_j. The argument in the proof of Theorem 7 shows that the vector process $(\tilde{\mathbf{A}}(t),\ t \geq 0)$ is a feasible process in $\mathcal{A}$.

It remains to show that by sending each component process $(\tilde{A}_j(t),\ t \geq 0)$ into its respective smoother we obtain an on-off process whose peak rate is $\min(r_j^1, \rho_j^1)$ and whose average rate is ρ_j. Specifically, we now show that $\tilde{A}_j(t)$ produces $\tilde{O}_j(t) = \int_0^t \tilde{o}_j(s + \theta_j) ds$ at the smoother output where

$$\tilde{o}_j(t) = \begin{cases} \min(r_j^1, \rho_j^1) & 0 \leq t < \tau_{\text{on}_j} \\ 0 & \tau_{\text{on}_j} \leq t \leq T_j \ , \end{cases}$$

where the periods and on–times are given in Table 8.1.

First, consider the case $\rho_j^1 \geq r_j^1$ and $\mathcal{E}_j(t_j) \geq S_j(\delta_j)$. Clearly, $t_{\mathrm{on}_j} \leq t_j$ since $t_{\mathrm{on}_j} = S_j(\delta_j)/\rho_j^1$ and $t_j = \mathcal{E}_j(t_j)/\rho_j^1$ and by assumption $S_j(\delta_j) \leq \mathcal{E}_j(t_j)$. This implies that $\mathcal{E}_j(t_{\mathrm{on}_j}) = \rho_j^1 t_{\mathrm{on}_j}$. Hence

$$S_j(\tau_{\mathrm{on}_j}) = \mathcal{E}_j(t_{\mathrm{on}_j}). \tag{8.3}$$

Note furthermore that

$$t_{\mathrm{on}_j} \leq \tau_{\mathrm{on}_j} \tag{8.4}$$

since $t_{\mathrm{on}_j} = S_j(\delta_j)/\rho_j^1 = r_j^1 \delta_j/\rho_j^1$ and by assumption $r_j^1 \leq \rho_j^1$. Because of (8.3) and (8.4) and $\tau_{\mathrm{on}_j} = \delta_j$ the smoother bursts at rate r_j^1 for a duration of τ_{on_j} when fed with an input burst at rate ρ_j^1 for a duration of $t_{\mathrm{on}_j} \leq t_j$. Also, note that the smoother output has average rate $\mathcal{E}_j(t_{\mathrm{on}_j})/T_j = \rho_j \leq r_j^{M_j}$, where the last inequality follows from the stability condition.

Next, consider the case $\rho_j^1 \geq r_j^1$ and $\mathcal{E}_j(t_j) < S_j(\delta_j)$. We have

$$\tau_{\mathrm{on}_j} \leq \delta_j \tag{8.5}$$

since $\tau_{\mathrm{on}_j} = \mathcal{E}_j(t_j)/r_j^1$ and $\delta_j = S_j(\delta_j)/r_j^1$ and by assumption $S_j(\delta_j) > \mathcal{E}_j(t_j)$. Thus $S_j(\tau_{\mathrm{on}_j}) = r_j^1 \tau_{\mathrm{on}_j}$. Hence

$$S_j(\tau_{\mathrm{on}_j}) = \mathcal{E}_j(t_{\mathrm{on}_j}). \tag{8.6}$$

Also,

$$t_{\mathrm{on}_j} \leq \tau_{\mathrm{on}_j} \tag{8.7}$$

since $t_{\mathrm{on}_j} = \mathcal{E}_j(t_j)/\rho_j^1$ and $\tau_{\mathrm{on}_j} = \mathcal{E}_j(t_j)/r_j^1$ and by assumption $\rho_j^1 > r_j^1$. Because of (8.5), (8.6) and (8.7) the smoother bursts at rate r_j^1 for a duration of τ_{on_j} when fed with an input burst at rate ρ_j^1 for a duration of t_{on_j}. The average rate of the smoother output is $\mathcal{E}_j(t_{\mathrm{on}_j})/T_j = \rho_j \leq r_j^{M_j}$, where the last inequality follows from the stability condition.

Now consider the case $\rho_j^1 < r_j^1$ and $\mathcal{E}_j(t_j) \geq S_j(\delta_j)$. We have $t_{\mathrm{on}_j} \leq t_j$ since $t_{\mathrm{on}_j} = S_j(\delta_j)/\rho_j^1$ and $t_j = \mathcal{E}_j(t_j)/\rho_j^1$ and by assumption $S_j(\delta_j) \leq \mathcal{E}_j(t_j)$. This implies that $\mathcal{E}_j(t_{\mathrm{on}_j}) = \rho_j^1 t_{\mathrm{on}_j}$. Hence

$$S_j(\delta_j) = \mathcal{E}_j(t_{\mathrm{on}_j}). \tag{8.8}$$

Note furthermore that

$$\delta_j \leq t_{\mathrm{on}_j} \tag{8.9}$$

since $\delta_j = S_j(\delta_j)/r_j^1$ and $t_{\mathrm{on}_j} = S_j(\delta_j)/\rho_j^1$ and by assumption $r_j^1 > \rho_j^1$. Because of (8.8), (8.9) and $\rho_j^1 < r_j^1$ (by assumption) the smoother passes the input burst at rate ρ_j^1 for a duration of t_{on_j} unchanged. The average rate of the smoother output is $\mathcal{E}_j(t_{\mathrm{on}_j})/T_j = \rho_j \leq r_j^{M_j}$, where the last inequality follows from the stability condition.

Finally, consider the case $\rho_j^1 < r_j^1$ and $\mathcal{E}_j(t_j) < S_j(\delta_j)$. These two assumptions imply that the smoother can pass the input burst of size $\mathcal{E}_j(t_j)$ at rate ρ_j^1. The average rate of the smoother output is $\mathcal{E}_j(t_{\mathrm{on}_j})/T_j = \rho_j \leq r_j^{M_j}$, where the last inequality follows from the stability condition. ■

Proof of Theorem 8: Using Lemma 3 and mimicking the proof of Theorem 7 we obtain

$$\phi_j \quad = \quad \frac{E\left[(\sum_{k=1}^{J} \tilde{U}_k - C)^+ \tilde{U}_j\right]}{C \cdot E[\tilde{U}_j]},$$

where $\tilde{U}_1, \ldots, \tilde{U}_J$ are defined in Lemma 3. Using the fact that $\tilde{U}_j$ is a Bernoulli random variable, we obtain from the above expression

$$\begin{aligned} \phi_j &= \frac{E\left[(\sum_{k\neq j} \tilde{U}_k + \min(r_j^1, \rho_j^1) - C)^+\right]}{C} \\ &\geq \frac{E\left[(\sum_{k\neq j} \tilde{U}_k + c_j^* - C)^+\right]}{C}, \end{aligned} \tag{8.10}$$

where the last inequality follows from Lemma 2.

From (6.15) and (8.10) it remains to show that

$$E[(\sum_{k\neq j} U_k^* + c_j^* - C)^+] \leq E[(\sum_{k\neq j} \tilde{U}_k + c_j^* - C)^+]. \tag{8.11}$$

From Lemma 2 and Proposition 1.5.1 in [71]

$$U_k^* \leq_{icx} \tilde{U}_k \quad \text{for all } k = 1, \ldots, J. \tag{8.12}$$

The inequality (8.11) follows from (8.12), the independence of $U_1^*, \ldots, U_J^*$ and an argument that parallels the argument in the proof of Theorem 7. □

Bibliography

[1] P. Billingsley. *Probability and Measure*. Wiley, 1979.

[2] L. Brakmo and L. Peterson. TCP Vegas: end to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communications*, 13(8):1465–1480, October 1995.

[3] C.-S. Chang. Stability, queue length and delay of deterministic and stochastic networks. *IEEE Transactions on Automatic Control*, 39(5):913–931, May 1994.

[4] L. T. Chia., D. J. Parish, and JWR Griffiths. On the treatment of video cell loss in the transmission of motion-JPEG and JPEG images. *Computers & Graphics*, 18(1):11–19, January 1994.

[5] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.

[6] R. Cruz. A calculus for network delay, part I: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–121, January 1991.

[7] R. Cruz. A calculus for network delay, part II: Network analysis. *IEEE Transactions on Information Theory*, 37(1):122–141, January 1991.

[8] R. Cruz. Quality of service guarantees in virtual circuit switched networks. *IEEE Journal on Selected Areas in Communications*, 13(6):1048–1056, August 1995.

[9] I. Dalgic and F. A. Tobagi. Characterization of quality and traffic for various video encoding schemes and various encoder control schemes. Technical Report CSL–TR–96–701, Stanford University, Departments of Electrical Engineering and Computer Science, August 1996.

[10] S. E. Deering. *Multicast Routing in Datagram Internetwork*. PhD thesis, Stanford University, December 1991.

[11] J. Dey, S. Sen, J. Kurose, D. Towsley, and J. Salehi. Playback restart in interactive streaming video applications. In *Proceedings of IEEE Multimedia*, Ottawa, Canada, 1997.

[12] B. T. Doshi. Deterministic rule based traffic descriptors for broadband ISDN: Worst case behavior and connection acceptance control. In J. Labetoulle and J. W. Roberts, editors, *Proceedings of International Teletraffic Congress (ITC) 14*, pages 591–600. Elsevier Science B. V., 1994.

[13] A. Elwalid, D. Heyman, T. Lakshman, D. Mitra, and A. Weiss. Fundamental bounds and approximations for ATM multiplexers with application to video teleconferencing. *IEEE Journal on Selected Areas in Communications*, 13(6):1004–1016, August 1995.

[14] A. Elwalid, D. Mitra, and R. H. Wentworth. A new approach for allocating buffers and bandwidth to heterogeneous regulated traffic in an ATM node. *IEEE Journal on Selected Areas in Communications*, 13(6):1115–1127, August 1995.

[15] W. Feng, F. Jahanian, and S. Sechrest. Providing VCR functionality in a constant quality video–on–demand transportation service. In *IEEE Multimedia*, Hiroshima, Japan, June 1996.

[16] W. Feng and J. Rexford. A comparison of bandwidth smoothing techiniques for the transmission of prerecorded compressed video. In *Proceedings of IEEE Infocom*, Kobe, Japan, April 1997.

[17] ADSL Forum. ADSL forum system reference model. Technical Report TR–001. http://www.adsl.com/adsl_reference_model.html.

[18] ADSL Forum. *ADSL Tutorial: Twisted Pair Access to the Information Highway.* http://www.adsl.com/adsl_tut.html.

[19] ATM Forum. *ATM User–Network Interface Specification, Version 3.0.* Prentice–Hall, 1993.

[20] M. W. Garret. *Contributions toward Real-Time Services on Packet Networks.* PhD thesis, Columbia University, May 1993. ftp address and directory of the used video trace: bellcore.com /pub/vbr.video.trace/.

[21] D.J. Gemmell and J. Han. Multimedia network file servers: multichannel delay sensitive data retrieval. *Multimedia Systems*, 1:240–252, 1994.

[22] D.J. Gemmell and H.M. Vin. Multimedia storage servers: A tutorial. *IEEE Computer*, pages 40–49, 1995.

[23] L. Georgiadis, R. Guerin, V. Peris, and K.N. Sivarajan. Efficient network QoS provisioning based on per node traffic shaping. *IEEE/ACM Transactions on Networking*, 4(4):482–501, August 1996.

[24] M. Grossglauser, S. Keshav, and D. Tse. RCBR: A simple and efficient service for multiple time-scale traffic. In *ACM SIGCOMM*, 1995.

[25] I. Hsu and J. Walrand. Admission control for ATM networks. In *IMA Workshop on Stochastic Networks*, Minneapolis, Minnesota, March 1994.

[26] J. Y. Hui. Resource allocation for broadband networks. *IEEE Journal on Selected Areas in Communications*, 6(9):1598–1608, December 1988.

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

[28] A. Hung and G. Kesidis. Resource management of prerecorded VBR sources in ATM networks. Technical Report 95-05, University of Waterloo, EECE.

[29] V. Jacobson. Congestion control and avoidance. In *Proceedings of SIGCOMM '88 Symposium*, pages 314–329, August 1988.

[30] F. P. Kelly. Effective bandwidth at multiple class queues. *Queueing Systems*, 9:5–16, 1991.

[31] E. Knightly, J. Liebeherr, D. Wrege, and H. Zhang. Fundamental limits and tradeoffs for providing deterministic guarantees to VBR video traffic. In *Proceedings of IEEE Infocom '95*, Boston, MA, April 1995.

[32] E. W. Knightly. H-BIND: A new approach to providing statistical performance guarantees to VBR traffic. In *Proceedings of IEEE Infocom '96*, San Francisco, CA, April 1996.

[33] E. W. Knightly and H. Zhang. Providing end–to–end statistical performance guarantees with bounding interval dependent stochastic models. In *Proceedings of ACM Sigmetrics '94*, 1994.

[34] E. W. Knightly and H. Zhang. Traffic characterization and switch utilization using a deterministic bounding interval dependent traffic model. In *Proceedings of IEEE Infocom '95*, Boston, MA, April 1995.

[35] E. W. Knightly and H. Zhang. D-BIND: an accurate traffic model for providing QoS guarantees to VBR traffic. *IEEE/ACM Transactions on Networking*, 5(2):219–231, April 1997.

[36] K. Kvol and S. Blaabjerg. Bounds and approximations for periodic on/off queue with applications to ATM traffic control. In *Proceedings of IEEE Infocom*, pages 487–494, 1992.

[37] M. Laubach. Cable modem basics: An introduction to cable modem technology, October 1996. Presentation at the University of Pennsylvania
Slides available at ftp.com21.com /pub/laubach/.

[38] M. Laubach. To foster residential area broadband internet technology: IP datagrams keep going and going and going.... *ConneXions*, 10(2), February 1996.

[39] T. Lee, K. Lai, and S. Duann. Design of a real–time call admission controller for ATM. *IEEE/ACM Transactions on Networking*, 4(5):758–765, October 1995.

[40] J. Liebeherr and D. Wrege. Video characterization for multimedia networks with a deterministic service. In *Proceedings of IEEE Infocom '96*, San Francisco, CA, March 1996.

[41] J. Liebeherr, D. Wrege, and D. Ferrari. Exact admission control for networks with a bounded delay service. *IEEE/ACM Transactions on Networking*, 4(6):885–901, December 1996.

[42] F. LoPresti, Z. Zhang, D. Towsley, and J. Kurose. Source time scale and optimal buffer/bandwidth trade–off for regulated traffic in an ATM node. In *Proceedings of IEEE Infocom*, Kobe, Japan, April 1997.

[43] W. Luo and M. El Zarki. Analysis of error concealment schemes for MPEG-2 video transmission over ATM based networks. In *Proceedings of SPIE Visual Communications and Image Processing 1995*, Taiwan, May 1995.

[44] K. Maxwell. Asymetric digital subscriber line: Interim technology for the next forty years. *IEEE Communications Magazine*, pages 100–106, October 1996.

[45] J. M. McManus and K. W. Ross. A comparison of traffic management schemes for prerecorded video with constant quality service. Technical report, University of Pennsylvania, Department of Systems Engineering, 1996. Available at http://www.eurecom.fr/~ross.

[46] J. M. McManus and K. W. Ross. Video on demand over ATM: Constant-rate transmission and transport. *IEEE JSAC*, 14(6):1087–1098, August 1996.

[47] J.M. McManus and K.W. Ross. A dynamic programming methodology for managing prerecorded VBR sources in packet-switched networks. In *Proceedings of SPIE, Performance and Control of Network Systems*, pages 140–154, Dallas, TX, November 1997. Available at http://www.eurecom.fr/~ross.

[48] D. Mitra and J. A. Morrison. Multiple time scale regulation and worst case processes for ATM network control. In *Proceedings of 34th Conference on Decision and Control*, pages 353–358, 1995.

[49] P. Mockapetris. @HOME network overview, October 1996. Presentation at the University of Pennsylvania.

[50] P. Oechslin. Worst case arrival of leaky bucket constrained sources: The myth of the on-off source. In *Proceedings of the fifth IFIP International Workshop on Quality of Service*, pages 67–77, Columbia University, New York, May 1997.

[51] A. Parekh and R. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[52] V. Peris. *Architecture for Guaranteed Delay Service in High Speed Networks.* PhD thesis, Institute for Systems Research, University of Maryland, College Park, 1997.

[53] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C, The Art of Scientific Computing.* Cambridge University Press, Cambridge, MA, second edition, 1992.

[54] S. Rajagopal, M. Reisslein, and K. Ross. Packet multiplexers with adversarial regulated traffic. In *Proceedings of IEEE Infocom '98*, pages 347–355, San Francisco, CA, April 1998.

[55] A.L.N. Reddy and J. Wyllie. Disk scheduling in multimedia I/O system. In *Proceedings of ACM Multimedia Conference*, 1992.

[56] M.I. Reiman. Some diffusion approximations with state space collapse. In F. Baccelli and G. Fayolle, editors, *Lecture Notes in Control and Informational Sciences 60*, pages 209–240. Springer-Verlag, 1983.

[57] M. Reisslein and K. W. Ross. Call admission for prerecorded sources with packet loss. *IEEE Journal on Selected Areas in Communications*, 15(6):1167–1180, August 1997.

[58] M. Reisslein and K. W. Ross. A join–the–shortest–queue prefetching protocol for VBR video on demand. In *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, pages 63–72, Atlanta, GA, October 1997. extended version available at http://www.eurecom.fr/~ross/.

[59] M. Reisslein and K. W. Ross. Prefetching protocols for VBR video on demand. In *Proceedings of Community Networking Workshop*, Atlanta, GA, September 1997. extended version available at http://www.eurecom.fr/~ross/.

[60] M. Reisslein, K. W. Ross, and V. Verillotte. Decentralized prefetching protols for VBR video on demand. In *Proceedings of 3rd European Conference on Multimedia Applications, Services and Techniques (ECMAST)*, Berlin, Germany, May 1998. extended version available at http://www.eurecom.fr/~ross/.

[61] J. Rexford and D. Towsley. Smoothing variable–bit–rate video in an internetwork. Technical Report CMPSCI–97-33, University of Massachusetts at Amherst, Department of Computer Science, May 1997. Availble via ftp://gaia.cs.umass.edu/pub/Rex97:Tandem.ps.Z.

[62] J. Roberts, U. Mocci, and J. Virtamo (Eds.). *Broadband Network Traffic: Performance Evaluation and Design of Broadband Multiservice Networks, Final Report of Action COST 242, (Lecture Notes in Computer Science Vol. 1155)*. Springer Verlag, 1996.

[63] J. W. Roberts, B. Bensaou, and Y. Canetti. A traffic control framework for high speed data transmission. In *Proceedings of IFIP Workshop Modelling, Performance Evaluation, ATM Technology*, pages 243–262, 1993.

[64] O. Rose. Statistical properties of MPEG video traffic and their impact on traffic modelling in ATM systems. Technical Report 101, University of Wuerzburg, Insitute of Computer Science, Am Hubland, 97074 Wuerzburg, Germany, February 1995.
ftp address and directory of the used video traces:
ftp-info3.informatik.uni-wuerzburg.de /pub/MPEG/.

[65] Keith W. Ross. *Multiservice Loss Models for Broadband Telecommunication Networks*. Springer–Verlag, 1995.

[66] J. Salehi, Z. Zhang, J Kurose, and D. Towsley. Optimal smoothing of stored video and the impact on network resource requirements. *submitted to IEEE/ACM Transactions on Networking*, 1996.

[67] J. Salehi, Z.-L. Zhang, Kurose J, and D. Towsley. Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing. In *Proceedings of ACM SIGMETRICS*, May 1995. Philadelphia, PA.

[68] S. Shenker and C. Partridge. Specification of guaranteed quality of service. Technical report. Internet Draft; December 1995.

[69] S. Shenker and J. Wroclawski. Request for comments 2215: General characterization parameters for integrated service network elements. September 1997.

[70] R. Stevens. *TCP/IP Illustrated, Volume 1, The Protocols*. Addison–Wesley, 1994.

[71] D. Stoyan. *Comparison methods for queues and other stochastic models*. Wiley, 1983.

[72] D. Veeneman and R. Olshansky. ADSL for video and data services. In *IEEE International Conference on Communications*, June 1995. http://www.gte.com/Adsl/News/Docs/9506.html.

[73] H. Vin and P. Venkat Rangan. Chapter 4: Multimedia storage systems. In *Multimedia Systems and Techniques, ed. B. Furht*, 1996.

[74] T. Worster. Modelling deterministic queues: The leaky bucket as an arrival process. In J. Labetoulle and J. W. Roberts, editors, *Proceedings of International Teletraffic Congress (ITC) 14*, pages 581–590. Elsevier Science B. V., 1994.

[75] D. Wrege, E. Knightly, H. Zhang, and J. Liebeherr. Deterministic delay bounds for VBR video in packet–switching networks: Fundamental limits and tradeoffs. *IEEE/ACM Transactions on Networking*, 4(3):352–362, June 1996.

[76] P. Yu, M. S. Chen, and D. D. Kandlur. Design and analysis of a grouped sweeping scheme for multimedia storage management. In *Third International Workshop on Network and Operating System Support for Digital Audio and Video*, 1993.

[77] H. Zhang. Providing end-to-end performance guarantees using non-workconserving disciplines. *Computer Communications: Special Issue on System Support for Multimedia Computing*, 18(10), October 1995.

[78] H. Zhang and D. Ferrari. Rate-controlled service disciplines. *Journal of High Speed Networks*, 3(4):389–412, 1994.

[79] Z. Zhang, J. Kurose, J. Salehi, and D. Towsley. Smoothing, statistical multiplexing and call admission control for stored video. *IEEE Journal on Selected Areas in Communications*, 13(6):1148–1166, August 1997.

[80] Q. Zhu, Y. Wang, and L. Shaw. Coding and cell–loss recovery in DCT-Based packet video. *IEEE Transactions on Circuits and Systems for Video Technology*, 3(3):248–258, June 1993.