EFFECT OF TRAFFIC KNOWLEDGE ON THE EFFICIENCY OF ADMISSION-CONTROL POLICIES

Ljiljana Trajković
Simon Fraser University
School of Engineering Science
Burnaby, BC V5A 1S6
Canada
Tel.: (604) 291-3998
FAX: (604) 291-4951
ljilja@cs.sfu.ca

Arnie Neidhardt
Bellcore
331 Newman Springs Road
Red Bank, NJ 07701
Tel.: (908) 758-5549
FAX: (908) 758-4371
arnie@bellcore.com

Abstract

We investigate the importance of understanding traffic characteristics for admission-control policies in packet networks. We compare the network utilization achieved with admission policies based on a partial knowledge of admitted traffic against the utilization that could be achieved with complete knowledge of traffic characteristics. Our quantitative study demonstrates that for realistic traffic traces the level of traffic knowledge dramatically affects admission control and improves network utilization. Indeed, the knowledge of only a few elementary traffic characteristics can produce substantial improvement. Yet, at least for one trace, this improvement based on elementary traffic knowledge is modest in comparison with the improvement that could be produced with additional traffic knowledge and if traffic characterization on more than one time scale were available.

1 Introduction

Researchers have adopted various approaches to designing connection admission control (CAC) algorithms. They include algorithms based on the notion of effective bandwidth [5], [10], [14], a decision-theoretic approach [8], and service-provisioning algorithms based on price-adjusting schemes [16]. Certain schemes were designed for bursty traffic and in-call re-negotiation [1], [12], [21]. Recently proposed algorithms explored statistical multiplexing of voice and data [18] and of video sources [6], the use of a neural-networks paradigm for online prediction [2], and real-time traffic measurements [13]. One such optimal re-negotiating scheme is based on large-deviations theory [20] and takes into account traffic structure on multiple time scales [9].

A CAC policy in a packet network should, in principle, depend on the quality of service (QoS) that needs to be provided to admitted users [4]. Under this ideal condition, a decision to admit a new connection is based on a prediction of the new QoS that would result from the additional traffic. One of the important obstacles to an accurate QoS prediction is the lack of understanding of the traffic characteristics. Without accurate predictions of the resulting QoS, a network may adopt an overly conservative admission policy that, while assuring certain agreed-upon QoS, lowers network utilization. If more detailed forecasts of network traffic were available, then QoS predictions could be sharpened and network utilization improved.

We want to distinguish here two obstacles to the creation of a CAC policy that ensures agreed-upon QoS, while allowing the network to be used efficiently:

- forecasting the relevant traffic features (traffic characterization), and
- deriving QoS from characteristics of the admitted traffic.

In this article we address the advantage of including traffic-information details in forecasting, but we do not solve the QoS-derivation problem. Indeed, the investigation of forecasting is hindered by our inability to solve the QoS-derivation problem.

If we could solve the QoS-derivation problem, this investigation could have been simplified. This solution would enable us to determine whether an adequate QoS would result if n traffic streams, whose traffic-generation processes were known, were multiplexed. At the time of connection admission, the traffic characteristics can never be known completely. Nevertheless, some traffic features could be understood, such as, for example, that p and m will bound the peak and mean of the traffic rate. Given that the true traffic-generation process is unknown in detail but known to have certain features, a solution of the QoS-derivation problem then determines a corresponding CAC policy: admit n connections only in case that the resulting service quality is adequate for all choices of n hypothetical traffic-generation processes having the known features. (In other words, assume the true traffic is the worst case still consistent with the known traffic characteristics.) If the traffic knowledge also has a Bayesian component, i.e., the true traffic-generation process is drawn at random from a "known" distribution of processes, then the corresponding CAC policy would admit n connections only in case the choice of n random processes drawn from this "known" distribution yields an adequate service quality "often enough." Therefore, the service standard must mention probabilities or expectations calculated based on the "known" distribution. A solution of the QoS-derivation problem would, therefore, determine a relationship between the traffic knowledge that an admission controller might have, and the CAC policy that would be used. Naturally, as traffic characterization becomes more complete, the corresponding CAC policy will admit more connections. Our interest is in identifying traffic features that are effective in raising the number of admitted connections. In other words, if an admission controller knows only these traffic features it would still admit almost as many connections as in the case of complete traffic information.

Unfortunately, the study must proceed without a solution to the QoS-derivation problem, i.e., without a complete and precise correspondence between levels of traffic knowledge and CAC policies. The study is possible, however, because we do have an approximate correspondence, though it is naturally incomplete and imprecise. Thus, we can explore how various levels of traffic knowledge correspond to differences in numbers of admitted connections. Because this correspondence lacks the case of complete traffic information, we cannot determine the important difference between the number of connections admitted based on limited information and the number that could be admitted based on complete information. To help estimate this difference, we use the following techniques:

- In addition to the conservative CAC policies that appear in our partial correspondence (between levels of traffic knowledge and CAC policies), we examine several heuristically motivated nonconservative CAC policies to generate rough estimates of what the conservative policies are missing in a broad sample of cases.
- We use simulations to explore interesting cases more thoroughly.

In this study we highlight the importance of traffic knowledge on more than one time scale. Here, by traffic knowledge on a time scale s, we mean knowledge of the marginal distribution of the arrivals in a time interval of length s. In the context of a single time scale, complete traffic knowledge refers to knowing the marginal distribution in detail, while incomplete knowledge refers to knowing only a few features of this distribution. In freer contexts that have not restricted attention to a particular time scale, traffic knowledge refers to all the joint distributions of arrivals in various intervals of various lengths. Complete traffic knowledge then implies knowledge of all these joint distributions. As noted earlier, the partial correspondence between traffic knowledge and a CAC policy misses the case of complete traffic information. Nevertheless, this correspondence includes the case of complete information on a single time scale, and thus indicates a CAC policy for this case. The difference in the number of connections that this policy admits (a relatively easy calculation) and the number of connections that the ideal policy would admit based on complete knowledge of joint distributions (the difficult task approximated with heuristic CAC policies or studied via simulations) can be attributed to the additional traffic knowledge. Thus, the benefit of knowing traffic on more than one time scale is in the possible improvement of a CAC policy over the one based on a single-time-scale marginal distribution. This study investigates the magnitude of this improvement.

We demonstrated in this study that, for realistic traffic traces, knowledge often dramatically affects admission control. Indeed, the knowledge of only a few elementary traffic characteristics always raised utilization substantially. For one trace, however, this improved utilization based on elementary knowledge is still modest, at least in comparison with the utilization that complete knowledge would produce. To approach this ideal utilization traffic characterization on more than one time scale must be available. We provide the details in the rest of this article. In Section 2 we first discuss in more detail the notion of "knowledge on a time scale" and we list the levels of traffic knowledge that we explored. In Section 3 we specify the corresponding CAC policies, thereby defining the correspondence between the available traffic knowledge and a CAC policy. We also explain the heuristically motivated nonconservative CAC policies that we used to estimate how much better an ideal policy could do compared with the conservative policies. In Section 4 we describe the two realistic traffic traces with which we explored the variation of admission control with traffic knowledge. We used the network efficiency as a criterion in comparing various CAC policies: if a CAC policy admits at most n traffic streams to be multiplexed on a network bottleneck of capacity C, and if m is the mean rate at which a single stream generates traffic (whether the admission controller "knows" m or not), then $\rho = nm/C$ is the maximum bottleneck efficiency. In Section 5 we report these efficiencies for various policies in a variety of circumstances. One particular set of circumstances seemed to offer the opportunity of demonstrating a dramatic improvement in efficiency by taking advantage of traffic knowledge on more than one time scale. We performed simulations to explore this case more carefully. The simulations confirming this improvement are described in Section 6. In Section 7 we discuss the implications and detailed results of the simulations. We conclude this article by Section 8 where we highlight major observations made in this study.

2 Traffic information a CAC policy might "know"

Traffic may be described in many ways, but a natural approach is to describe the random amounts of traffic generated in various intervals [3], [7], [15], [22]. No engineer implementing a CAC policy is likely to know in detail all the joint distributions of these random amounts, but some of the less complex statistics are likely to be available. For instance, there may be one well studied time scale $s > 0$, for which one might know reasonably well the characteristics of the traffic generated in a given interval of length s. With more ambition, one may attempt to learn about the distributions corresponding to more than one time scale s, or even about joint distributions. One subject of this investigation is whether such ambition pays off, i.e., this report explores how much additional efficiency can be achieved if one knows more about traffic than its features on only a single time scale.

A traffic process is identified by the family of random variables $A([s,t))$, representing the amount of traffic arriving in the interval $[s,t)$. An ideal traffic characterization would identify the joint distribution of these random variables. A more practical characterization would obtain some useful knowledge of the joint distribution without identifying the distribution in detail. For the purposes of this study, we identify the traffic-generation knowledge, that an admission controller might have, with a set of possibilities for the joint distribution of the random arrivals $A([s,t))$: the set of distributions that are consistent with what the controller "knows." Complete knowledge would limit this set to a single joint distribution.[1] We assume that the traffic-generation processes are stationary, i.e., the joint distribution of the original process $A([s,t))$ is the same as that of the translated process $A_r([s,t)) = A([s+r,t+r))$. In particular, the amount of arrivals in any interval of length s has the same distribution as $A(s) = A([0,s))$. Thus, for this study, traffic knowledge is a description of the joint distribution of all the random variables in $\{A(s)|0 \leq s < \infty\}$. In contrast, traffic knowledge on a single time scale s refers to the distribution of just one of these random variables.

When traffic knowledge is available on only one scale, we imply that no additional information about traffic is available. If an admission controller knows traffic features only on the time scale s, then the controller cannot know whether arrivals are independent from one interval of length s to the next; whether the arrivals in various intervals are positively or negatively correlated; the typical duration of the bursts; whether the traffic is generated by a Markov-modulated generator; or whether the traffic is self-similar. If an admission controller is conservative, it must admit connections only if service quality can be assured regardless of how "badly" the traffic in various intervals "conspires" against the bottleneck. (We will assume, however, that the arrivals from various streams are independent. Therefore, the CAC policies we consider would be vulnerable to conspiracies among different users.) In particular, the controller cannot make any default assumption (beyond stationarity) about how the one random variable $A(s)$ fits into a process. It can draw strictly logical inferences, of course. For instance, if an admission controller knows that the mean value of $A(s)$ is bounded by m, then it may know that the mean of $A(2s)$ is bounded by $2m$. For our study, however, if an admission controller has traffic knowledge only on one time scale s, then the controller does not depend on any traffic features on other scales that cannot be derived by logic alone. Specifically, the controller cannot assume independence in various intervals of the same stream. It cannot assume that arrivals are generated in bursts whose successive durations are independent; neither can it assume that the arrival process is self-similar. The reader might be surprised that any statistical-multiplexing gains are possible in these circumstances. Nevertheless, we will demonstrate that gains are possible.

One question that arises if a traffic process is to be described by just a single marginal distribution of $A(s)$ is the choice of the most useful time scale s. For a potential bottleneck of a queue served at a fixed rate, one particularly relevant time scale s is the half-buffer work time. This is the time that the queue server requires to serve the amount of traffic that could be stored in half

[1] In general, complete traffic knowledge would also include the description of any reactions of the traffic-generation process to feedback from congestion in the network. In this study, however, we only explore traffic that a user would generate for an uncongested network and we ignore congestion reactions.

of the allocated buffer. The relevance of this scale is that a conservative admission policy can be constructed based only on traffic knowledge at this scale. The construction is possible, as will be described in gory detail later, mostly because if the controller knows that excess arrivals on the half-buffer time scale almost never occur, then it basically also knows that any excesses on smaller time scales cannot cause an overflow. With the same approach over a larger time scale, one can still achieve a conservative policy, but the policy will operate with the following handicap: the controller must be assured that the traffic generated in the longer time interval is still within the same small value of a half-buffer's worth. Conversely, if one tries the same approach over a smaller time scale, one finds a conservative policy with a subtler handicap: the policy would demand that generated traffic on the small scale nearly always fits within the bottleneck's capacity, which prevents one from taking almost any advantage of the fact that traffic bursts do not always persist. These considerations indicate that the half-buffer work time is the most relevant scale for a conservative admission policy that relies on traffic knowledge on only one time scale.

We now list various levels of traffic knowledge, emphasizing the levels that we explored and indicating corresponding CAC policies and performances.

I. For systems in which the access *link speed* is known, with no knowledge of the traffic that the link will carry, the (*physical*) *peak-rate allocation* admission policy is the only viable option. Naturally, this scheme often achieves very little efficiency.

II. If a user's *peak traffic rate* on the half-buffer time scale is known, the (*traffic*) *peak-rate allocation* policy is justifiable. This approach yields an improved efficiency, even for links serving only one connection (i.e., links whose capacity is less than the user's physical peak rate can now serve the connection). Nevertheless, with such a policy no further gains from statistical multiplexing are possible.

III. Knowing a user's *mean traffic rate* in addition to the *peak*, would lead to some multiplexing gains and would even allow link utilization to approach unity at very large capacities.

IV. Additional knowledge of the *variance* of a user's traffic on the half-buffer time scale would allow further gains. One quantitative indication of these gains is that, asymptotically, the idle time approaches zero at the same rate as in the next case.

V. Knowing the *complete marginal distribution* of a user's traffic on the single time scale of a half-buffer work time would allow further multiplexing gains, especially at the more practical link capacities below the asymptotic regime.

VI. Knowing the *traffic distribution on more than one time scale* should lead to further increases in utilization.

VII. *Complete traffic information* would allow the use of an ideal policy. The "only" restriction would be the computational complexity in implementing the policy.

The improvements along the progression (Cases I–VII) are to be expected from the simple observation that greater knowledge provides the possibility for a greater efficiency. For Cases III–V, the possibilities for nearly ideal efficiency at high multiplexing levels are not quite obvious, but are, nevertheless, still provable, as will be discussed in Section 3.2.

To evaluate these cases quantitatively at intermediate levels of multiplexing, we have examined analytically the efficiencies of these CAC policies for two genuine traffic traces. These analytical calculations are possible because Cases I–V correspond to specific conservative policies that take at least some advantage of the available traffic information. Cases VI and VII, however, do not relate to specifiable general policies that clearly take advantage of all the available information. Thus, analytical calculations could explore the benefits of various details of traffic knowledge within a single time scale (Cases I–V), but not beyond. To continue the exploration, i.e., to estimate the additional efficiency that could be attained with complete traffic knowledge, we first devised some nonconservative CAC policies, with the idea that they would act as if they were taking advantage of extra knowledge. With these nonconservative policies, we made preliminary calculations of efficiencies in a variety of situations. For one of these cases we also performed simulations.

3 Catalog of CAC policies

A network admission-control policy should take traffic knowledge into account, but it must also account for network capacities and the QoS standard to be met. For this investigation of how traffic knowledge affects network efficiency, we examined only the case when the admission control protects a single bottleneck link (indeed a single direction of the link) with a given service rate and a given buffer size. We choose the service standard to be that for any admitted connection,

$$P(\text{traffic lost during a service-monitoring interval } D) \leq \epsilon,$$

where the lost traffic refers to the given connection. Since the issue here is congestion, the only traffic losses that are relevant are those from queue overflows. During a queue overflow one should not expect many streams to escape unscathed and the service standard above should not be much more permissive than[2]

$$P(\text{queue overflow during a service-monitoring interval } D) \leq \epsilon.$$

The link parameters are its speed C cells/second and its buffer size $B \approx 2Ch$, where h is approximately the half-buffer work time. We considered two genuine traffic traces. Thus, by introducing a parameter *TrafficDescription* to distinguish the two traces, we index any of the cases we considered by five parameters: *TrafficDescription*, C, D, ϵ, and h. In any particular situation, a general admission policy will limit the admission to $Nmax$ simultaneous homogeneous sessions, where the policy's determination of $Nmax$ will be affected by C, D, ϵ, and h, and the details of *TrafficDescription* that the policy is "prepared to understand." In other words, a general policy will "calculate" $Nmax$ based on some of the features available from *TrafficDescription*. For this investigation of how traffic knowledge affects efficiency, we examined various general policies that differed in how much detail of the traffic description was used to calculate $Nmax$. To distinguish the different policies, we introduced another parameter *CACname*. Thus, a particular calculation is distinguished by six parameters: the five parameters associated with the traffic situation, and the parameter *CACname* that identifies the procedure for determining $Nmax$ from the other parameters.

We compared ten CAC policies reflecting various degrees of traffic knowledge. They are named in Table 1 together with an indication of distinguishing features. The first five CAC policies correspond to the first five levels of available traffic information listed in Section 2. Each of these CAC policies (Cases 1–5) is conservative and takes at least some advantage of the available traffic

[2]This is a simplification of service characterization because in an actual network effects of congestion events might not be equitably distributed over sessions.

Connection Admission Control policies		
Case	*CACname*	*Traffic knowledge or its treatment*
1	*HIPPI*	physical link peak rate
2	*peak*	traffic peak rate
3	*PM*	traffic peak and mean rate
4	*PMV*	traffic peak, mean, and the variance
5	*true*	the true marginal distribution
6	*normLT*	a normal approximation (to a Laplace Transform)
7	*normPB*	a normal approximation (to a probability)
8	*optsb*	optimistic, using a small buffer size
9	*opt/2*	optimistic, with a factor of 2
10	*vopt*	very optimistic

Table 1: Ten CAC policies used in comparison, all relying on traffic knowledge on the half-buffer time scale.

information. The remaining CAC policies are variations intended to estimate the behavior of a CAC policy with complete traffic information.

More precisely, the admission policies we considered all correspond to variations on the following calculation:

$$\begin{aligned}
&\Pr(\ \exists t \in [0, D),\ \text{the queue overflows } B \text{ at time } t) \\
&\leq \Pr(\bigcup_{i=1}^{L}\{A([p_i, p_{i-1})) > ch\}) && (1)\\
&\leq L\ \Pr(A(h) > ch) && (2)\\
&\leq L\ \mathrm{E}e^{r(A(h)-ch)} && (3)\\
&= L\ (\mathrm{E}e^{r(A_1(h)-ch/n)})^n && (4)\\
&\leq L\ (\mathrm{E}e^{r(\bar{A}_1(h)-ch/n)})^n && (5)\\
&\leq \epsilon, && (6)
\end{aligned}$$

where:

ϵ is the tolerable overflow probability given by the service criterion.

D is the length of a service-monitoring interval given by the service criterion.

B is the known buffer size in cells.

C is the known link service rate.

n is the known number of sessions contributing traffic.

X, c, h, and r are arbitrary positive numbers that an admission controller may select in an attempt to satisfy the inequalities above. In particular:

X is the length of a prolog $[-X, 0)$ preceding the service-monitoring interval $[0, D)$.

c is slightly smaller than C.

h is the time scale on which features of the traffic process affect the calculation. We choose it to be slightly smaller than the half-buffer work time.

$L = \lceil (X + D)/h \rceil$ is the number of intervals of length h required to cover $[-X, D)$, the service-monitoring interval together with the prolog.

$p_i = D - ih$ for each integer i, yielding points in a sequence that provides one way of identifying a partition into intervals of length h. The intervals $[p_i, p_{i-1})$ for $1 \leq i \leq L$ form a partition of $[p_L, D) \supset [-X, D)$.

$A([p_i, p_{i-1}))$ is the aggregate traffic arriving during $[p_i, p_{i-1})$ from all n streams, which is only partially known to the admission controller.

$A(h)$ is the similarly unknown aggregate traffic arriving during $[0, h)$.

$A_k(h)$ is the unknown traffic from stream k. (Hence, $A_1(h)$ is the traffic from stream 1.)

$\bar{A}_1(h)$ is a random variable whose distribution can be selected by the admission controller. This selection should reflect traffic knowledge in the sense that the controller cannot rely on the comparison (5) between the true distribution of $A_1(h)$ and the controller's selection of $\bar{A}_1(h)$ without some knowledge of the true distribution. The admission controller's traffic knowledge affects the calculation only through this choice of $\bar{A}_1(h)$ and the comparison (5). Hence, h is the only time scale at which traffic knowledge affects this calculation.

The comparisons (1)–(6) outline a proof of good service, under the general assumptions that various streams generate traffic independently and that each stream is stationary. The comparisons prove that a CAC policy is conservative, provided that it admits n sessions only when X, c, h, r, and $\bar{A}_1(h)$ can be chosen to satisfy (1)–(6). The CAC policies we study are constructed by specifying certain choices. Some of our choices are arbitrary, but we always ensure that (1)–(5) hold for the conservative CAC policies. The remaining inequality (6) then becomes the test that n has to pass for the CAC policy to admit n connections. In other words, (6) determines $Nmax$ for the CAC policy.

We now show how the comparisons (1)–(5) can be ensured. The biggest task is associated with the first inequality (1) that states that the queue-overflow event is contained in the union of excess-arrival events. We will show that this containment holds, provided that $c < C$, $2ch \leq B$, and $X(C - c) \geq B$. Let us suppose that the containment fails, i.e., that there is an outcome in the queue-overflow event that is not in the union of L excess-arrival events. In other words, suppose that there is a traffic pattern that can arrive and cause the queue to overflow during the interval $[0, D)$ while meeting the constraints

$$A([p_i, p_{i-1})) \leq ch \tag{7}$$

for $1 \leq i \leq L$. Our proof will be by deriving a contradiction from supposition (7).

Before giving a detailed proof, we introduce the main ideas of the proof. The constraints (7) imply that the arrival rate is mostly confined to the link's capacity. Therefore, the queue should not even be able to grow, let alone overflow. Of course, this idea is too simple, for the constraints refer only to the average rates over the few "distinguished" intervals $[p_i, p_{i-1})$. (We use the term distinguished for intervals named in the list of constraints (7).) In particular, while the constraints imply that no more than ch traffic cells can arrive in any one distinguished interval, the constraints do not prevent this amount from arriving as a burst at a single instant. Whenever such an instantaneous burst arrives, the queue will grow by the size of the burst, simply because the link has no time to serve it. The constraints imply that such bursts can never be larger than ch, and that there can be at most one burst of this size in any distinguished interval. Since $ch \leq B/2$, each burst can cause a queue growth of no more than half a buffer. If one such burst arrives at the end of one distinguished interval, and if another burst arrives at the beginning of the next interval (a pattern allowed by the constraints), then the queue could grow by $2ch \leq B$, an amount that does not quite produce an overflow. While this threatening pattern might suggest that overflows are possible, a more careful observation reveals that this pattern is already the "worst" that can occur. Specifically, for queue growth greater than $2ch$, the growth must extend over a period that involves more than two distinguished intervals (the only way to include more than $2ch$ arrivals). Yet, over any of the intermediate intervals, the link is working at its full capacity C while the average arrival rate is no greater than c, hence the intermediate intervals are *not* "helping" to produce an overflow. These observations indicate that the constraints are sufficient to rule out overflows from bursts that start to arrive when the queue is empty. The purpose of the prolog is then to ensure that the queue does empty at some instant before the service-monitoring interval begins. (It is irrelevant that a nonzero queue can reform to start the service-monitoring interval and that overflows can happen during the prolog.)

The detailed argument of our proof by contradiction (to the supposition of a queue overflow during $[0, D)$ while the constraints (7) are met) begins with a preliminary consideration of the possibility that the link could be busy throughout the prolog $[-X, 0)$. More precisely, given that the constraints (7) deal only with the distinguished intervals ending at the partitioning instants p_i, the interest here is in a union of distinguished intervals that covers $[-X, 0)$. Explicitly, let p_z be the smallest non-negative instant among the partitioning instants p_i. (Hence, $p_z = 0$ if D is an integral multiple of h.) Then $[p_L, p_z) \supset [-X, 0)$. We are interested in whether the link can be busy throughout $[p_L, p_z)$. Note that $A([p_L, p_z)) \leq c(p_z - p_L)$, because we supposed that the constraints (7) hold. Note further that $p_z - p_L \geq X$. If the link is busy throughout $[p_L, p_z)$, then it must complete $C(p_z - p_L)$ amount of traffic work during this interval, which exceeds the arrivals in the same time by at least $(C - c)(p_z - p_L) \geq (C - c)X \geq B$. As with any interval in which the amount of completed work exceeds the amount of arrivals, the queue of unfinished work must decrease by at least the excess of work. (If losses occur during the same interval, then the decrease is even greater.) In the case of the interval $[p_L, p_z)$, the implication is that if the link is busy throughout the interval, then the queue must decrease by the buffer size B. In other words, regardless of whether the link is busy throughout $[p_L, p_z)$, the queue must empty at some instant in $[p_L, p_z]$.[3]

After these preliminary considerations we now consider an instant $t \in [0, D)$ when an overflow occurs. Note first that no instantaneous burst at time t may be larger than ch, because t belongs to one of the intervals $[p_i, p_{i-1})$ and we suppose that the constraints (7) hold. Hence, the queue size $Q(t)$, immediately before any possible instantaneous arrival at t, must exceed $B - ch > 0$. Consider now the busy period (when $Q > 0$) that contains t. Specifically, consider whether it can extend back to p_L. If $t \geq p_z$, then we have already shown in the previous paragraph that the busy period cannot extend back to p_L. Accordingly, consider the complementary case $t < p_z$. Note that because of the choice of p_z, $0 \leq t < p_z < h$. Now, if the busy period extends back to p_L, then the

[3]We assume that a traffic burst arriving at a single instant t influences the queue size Q only at times after t. With this interpretation, the conclusion of the argument just given can be expressed formally as $\exists u \in (p_L, p_z], Q(u) = 0$, instead of $\exists u \in (p_L, p_z], Q(u-) = 0$. Moreover, this implies that $Q(u) + A([u, v))$ is the total amount of traffic with which the system must cope in an interval (u, v).

link will have been busy throughout $[p_L, t)$, completing work in the amount of $C(t - p_L)$. Once again, to the extent that this amount exceeds the arrivals $A([p_L, t))$ in the same interval, the queue must decrease. Yet, as has already been noted, $Q(t) > B - ch$. Thus, if the busy period extended back to p_L, then

$$\begin{aligned}
B - ch &< Q(t) \\
&\leq Q(p_L) + A([p_L, t)) - C(t - p_L) \\
&\leq B + A([p_L, p_z)) - C(t - p_L) \\
&\leq B + c(p_z - p_L) - C(t - p_L) \\
&= B + c(p_z - t) + c(t - p_L) - C(t - p_L) \\
&< B + ch - (C - c)(t - p_L) \\
&\leq B + ch - (C - c)X \\
&\leq ch,
\end{aligned}$$

which contradicts the condition $2ch \leq B$. Therefore, the busy period cannot extend back to p_L, but must begin at some instant $t_1 \in (p_L, t)$.

The next step is to consider the traffic arriving in $[t_1, t]$. More precisely, in order to include the case of continuous arrivals in which no positive amount of traffic overflows precisely at time t, we consider times s slightly greater than t, but still in the same busy period. Let $S([u, v))$ be the amount of traffic served in the interval $[u, v)$, and let $\delta([u, v))$ be the amount of traffic lost in overflows during the same period. Then,

$$Q(u) + A([u, v)) = S([u, v)) + Q(v) + \delta([u, v)). \tag{8}$$

For the cases $[u, v) = [t_1, s)$, in which the link is busy throughout the interval and the queue starts empty, $Q(t_1) = 0$ and $S([t_1, s)) = C(s - t_1)$, Equation (8) becomes

$$\delta([t_1, s)) = A([t_1, s)) - C(s - t_1) - Q(s). \tag{9}$$

The fact that an overflow occurs at t means that for any open interval (u, v) containing t, $\delta([u, v))$ is strictly positive. In particular, taking any time $v_0 \in (t, D)$ such that v_0 is still in the same busy period as t,

$$\delta_0 = \delta([t_1, v_0)) > 0. \tag{10}$$

To deal more particularly with a time that is very close to an overflow instant, let v_1 be the last time in $[t_1, v_0)$ at which overflow occurs. Then $v_1 \geq t$. If v_1 is the arrival time of an instantaneous burst, then this burst would be contributing to the overflow δ_0, but would be missing from $\delta([t_1, v_1))$. Nevertheless, for all times $s > v_1$, $\delta([t_1, s)) \geq \delta_0$, and $Q(s) \geq B - C(s - v_1)$. Let $s \in (v_1, D)$ be in the same busy period as t and satisfy $C(s - v_1) \leq \delta_0/2$. Then, $Q(s) \geq B - \delta_0/2$ and equation (9) implies

$$\begin{aligned}
A([t_1, s)) - C(s - t_1) - B &= A([t_1, s)) - C(s - t_1) - (B - \frac{\delta_0}{2}) - \frac{\delta_0}{2} \\
&\geq A([t_1, s)) - C(s - t_1) - Q(s) - \frac{\delta_0}{2} \\
&= \delta([t_1, s)) - \frac{\delta_0}{2} \\
&\geq \delta_0 - \frac{\delta_0}{2} \\
&> 0.
\end{aligned}$$

Thus,

$$A([t_1, s)) > C(s - t_1) + B. \tag{11}$$

The next step is to show that for any times $y < s$ in $[p_L, D)$, $A([y, s)) \leq C(s - y) + B$. This step formalizes the idea that the constraints (7) should prevent the queue from growing by more than $2ch$. Let i and j be the indices for which $y \in [p_{i+1}, p_i)$ and $s \in [p_j, p_{j-1})$. Then $p_i \leq p_j$, except in the case that y and s belong to the same distinguished interval. In this exceptional case the inequality holds trivially: $A([y, s)) \leq ch < C(s - y) + B$. In the remaining cases when $p_i \leq p_j$,

$$\begin{aligned} A([y, s)) &\leq A([p_{i+1}, p_i)) + A([p_i, p_j)) + A([p_j, p_{j-1})) \\ &\leq c(p_{j-1} - p_{i+1}) \\ &\leq c((s - y) + 2h) \\ &\leq C(s - y) + 2ch \\ &\leq C(s - y) + B. \end{aligned}$$

Thus, in either case, $A([y, s)) \leq C(s - y) + B$. In particular, for $y = t_1$, the conclusion is that

$$A([t_1, s)) \leq C(s - t_1) + B, \tag{12}$$

which contradicts (11). This contradiction shows the falsity of the supposition of a possible traffic pattern that overflows the queue during $[0, D)$ while meeting the constraints (7). Thus, we demonstrated the validity of the first inequality (1) in the calculation (1)–(6).

The remaining relations (2)–(4) require less justification and hold for any choice of appropriate quantities. Inequality (2) follows from the assumed stationarity of A — specifically, that $\Pr(A(h) > ch)$ equals the probability of each of the excess-arrival events $\{A([p_i, p_{i-1})) > ch\}$. Inequality (3) is true if $r \geq 0$. Equality (4), expressing an exponential moment for the aggregate process as the product of moments for the individual streams, holds because of the independence of the various streams. For homogeneous cases, this product of moments over n streams can be written as an n-th power.

The comparison (5) requires some care in selecting $\bar{A}_1(h)$. We use various selections corresponding to various CAC policies, as described below. Yet, for all our conservative policies, we ensure that (5) holds. Since inequality (5) is obtained by replacing the random amount of arrivals $A_1(h)$ by another random variable $\bar{A}_1(h)$, the inequality is true if the relevant exponential moments of the new random variable are greater than or equal to those of the old one. The reader should anticipate that we select our new random variables with the care needed to ensure greater exponential moments. The tricky part of the selection, however, is that with only partial traffic knowledge we must exercise this care without knowing precisely what the old exponential moments were.

To summarize the discussion above, we have explained the extent to which the various inequalities (1)–(6) should be expected to hold. An elaborate argument showed that (1) always holds if X, c, and h satisfy a few elementary constraints: $c < C$, $2ch \leq B$, and $(C - c)X \geq B$. The comparisons (2)–(4) always hold if $r \geq 0$. The inequality (5) may always be satisfied, but the selection of $\bar{A}_1(h)$ must account for the limited traffic information available. Accordingly, in our selections various choices of $\bar{A}_1(h)$ will correspond to various CAC policies. Finally, recall that the inequality (6) is not expected to hold in all cases, i.e., for all values of n. Instead, it determines $Nmax$ as the largest value of n for which the inequality holds.

In general, one could use (1)–(6) with any suitable choices of X, c, h, r, and $\bar{A}_1(h)$ to design a conservative CAC policy. Various choices would correspond to various conservative policies with different efficiencies. Hence, it helps to use some care when making the choices. Indeed, within this framework, the most efficient policy would include an exhaustive search of all these choices. Instead, however, we employ an optimizing search only for the parameter r, while investing little effort in the remaining choices. (We also employ optimizing selections for some of the other choices, such as $\bar{A}_1(h)$, but these selections are readily constructed without a need for a search.) The little effort we did invest went into the following observations. We observed that the resulting CAC policy could be efficient only if c was close to C. In view of the relation $(C-c)X \geq B$ required for (1) to hold, c can be close to C only if X is very large. Once X is large, however, the resulting CAC policy is not very sensitive to the precise value of X. Based on these observations, together with the expectation that D would usually be far larger than h, we made the rather arbitrary choice of $X = D$. The choice $c = C - B/X$ then follows. We have also discussed earlier the choice $h = B/2c$. Then, in conformance with other earlier discussions, for a CAC policy that relies on only partial knowledge of a marginal traffic distribution, the random variable $\bar{A}_1(h)$ in (5) should have the "worst" possible distribution consistent with the partial knowledge of the traffic. The identification of this pessimal distribution for a given traffic-knowledge level will be described in conjunction with the distinct features of various CAC policies. We describe the technique we employed to seek the optimal value for r in Section 3.1.

As we have just described, the CAC policies in this study are distinguished by the choice of $\bar{A}_1(h)$. In case the admission controller "knows" the marginal distribution of the traffic-generation process on the time scale h, then the CAC policy can rely on calculations using the true distribution of $A_1(h)$, i.e., the choice $\bar{A}_1(h) = A_1(h)$ is permitted. Thus, the policy *CACname* = *true* is obtained from (1)–(6) by choosing $\bar{A}_1(h) = A_1(h)$.

Other conservative policies (Cases 1–4), based on less information than that available to *CACname* = *true*, cannot employ the choice $\bar{A}_1(h) = A_1(h)$, because they do not "know" the distribution of $A_1(h)$. Instead, for these cases, only a few features of the distribution are known, such as bounds on the peak and the mean. Accordingly, the corresponding CAC policy must employ a choice of $\bar{A}_1(h)$ that has the "worst" distribution consistent with the known features. With respect to the substitution (5), the "worst" distribution means the one with the largest value for $Ee^{r\bar{A}_1(h)}$. In general, for any $r \geq 0$ and for a set of constraints on the distribution of a random variable, if the constraints restrict the distribution to a compact set of distributions supported on a common compact interval, then there is a random variable Y whose distribution satisfies the given constraints and maximizes the expression Ee^{rY}. The sets of constraints in Cases 1–4 shown in Table 1 correspond to such compact sets of distributions. Hence, there is always a "worst" choice of $\bar{A}_1(h)$ that a CAC policy can employ.

- If the only constraint is an upper bound p on $A_1(h)$ (Cases *CACname* = *HIPPI* and *CACname* = *peak*), then the maximizing random variable is the constant $Y = p$.

- If the only constraints are bounds p and m on the peak and the mean, i.e., $p \geq A_1(h)$ and $m \geq$ the average value of $A_1(h)$ (Case *CACname* = *PM*), then the maximizing random variable Y assumes the value p with probability m/p and assumes the value 0 otherwise. Hence, Y is an on-off random variable.

- If the only constraints are bounds on the peak p, the mean m, and the variance v (Case *CACname* = *PMV*), then the maximizing random variable Y again assumes only two values: p and a lower value that yields the mean m and the variance v. Hence, Y is a high-low random variable.

In all these cases the maximizing distributions are independent of r and are easily identified in terms of the available traffic information. Accordingly, the conservative admission policies (the ones relying on less information than the complete marginal distribution, i.e., Cases 1–4) choose $\bar{A}_1(h)$ to be the maximizing distribution.

Strictly speaking, the policies we constructed also take into account the fact that if a traffic stream is limited to a peak rate p, then no queue overflows can occur without admitting more than C/p connections. This consideration has almost no influence, except in the case of the peak-rate-based CAC policies *CACname = HIPPI* and *CACname = peak*. Even for these policies, this consideration raises $Nmax$ only by the ratio of C/c from the value that would be allowed by calculations (1)–(6). This difference is usually negligible, unless the number of connections is small and an integer falls in the interval $(c/p, C/p)$, in which case one more connection is allowed and this connection is a significant contribution to efficiency.

The remaining policies (Cases 6–10) are not conservative because not all inequalities (1)–(6) are preserved. In policies *CACname = normLT* and *CACname = normPB*, $\bar{A}_1(h)$ was chosen to have the normal distribution with the same mean and variance as the actual traffic distribution $A_1(h)$. The reason for introducing exponential moments in the original calculations (1)–(6) was to avoid the difficulty of performing n-fold convolutions to determine the tail probability $\Pr(A(h) > ch)$ of a sum $A(h)$ of independent random variables. By employing normal approximations, however, the difficulty vanishes because the corresponding approximation to $A(h)$ is a normal random variable whose mean and variance are n times the individual mean and variance. Thus, with the normal approximation, the exponential moments can be eliminated. The policy *CACname = normLT* uses these moments, while the policy *CACname = normPB* does not. Policy *CACname = normPB* has greater efficiency than *CACname = normLT*. (Note that for $r > 0$, the exponential moment Ee^{rY} of a positive random variable Y is the Laplace transform (LT) of its distribution evaluated with the negative argument $-r$. Hence, we named it *CACname = normLT*.)

Nonconservative policies (Cases 8–10) also employ normal approximations, but they are more optimistic. Note that to create a buffer overflow in time t, starting with an empty queue, the number of arrivals must exceed $Ct + B$. In particular, to create an overflow in the time interval h, the arrivals must exceed $Ch + B \approx 3Ch$. If one expects overflows to occur only from sudden bursts, one might be tempted to replace ch by $3Ch$ in (1). This replacement gives the policy *CACname = opt/2*. If one also realizes that this assumption of overflows occurring only suddenly implies that the queue behavior before time 0 is irrelevant, then one would also set $X = 0$ in defining L for (1). This setting gives the very optimistic policy *CACname = vopt*. Note that the variation of X from D to 0 changes the factor $(X + D)/h$ by a factor of 2 Hence, the factor 2 in the name *CACname = opt/2*. If one expects typical overflows to require more than a half-buffer work time to be created, one would be more concerned with the events when $A(h)$ exceeds Ch without exceeding $Ch + B$. As an intermediate approach, *CACname = optsb* compares ϵ with $(D/h)\Pr(A(h) > Ch + b)$, where b is roughly $B/10$, and is precisely the buffer size for which the policy *CACname = true* would take into consideration the time scale $h/10$. Thus, *CACname = optsb* has the flavor of one of the very optimistic policies applied to the smaller buffer of size b.

3.1 Procedure for finding optimal r

We have defined the CAC policies in terms of calculations (1)–(6), the prescribed choices for X, c, h, $\bar{A}_1(h)$, and an optimizing choice for r. As one of the practical considerations of implementing a CAC policy, we discuss here the procedure for finding the optimal r. For this study, in which the interesting quantity of a CAC policy is the number of admitted connections $Nmax$, we must search for r without knowing n ahead of time. (We assume that we are dealing only with homogeneous connections.) In contrast, a CAC policy would make an admission decision while knowing n, but it would have to deal with a heterogeneous mix of connections, and, therefore, the n^{th} power in (5) would be replaced by an n-fold product. Nevertheless, the search for the optimal r is no more difficult conceptually, though each step becomes lengthier as n different moments are calculated separately. Incidentally, the fact that our search begins without knowing n is not a hardship, because a single value of r will be decisive for all n.

For a given CAC policy, we calculate $Nmax$ in the following manner. By definition,

$$\begin{aligned} Nmax \quad = \quad & \max\{n | n \geq 0 \ \& \ (\ np \leq C \quad \text{or} \\ & \qquad \exists r \geq 0, L(\mathrm{E}e^{r(\bar{A}_1(h) - ch/n)})^n \leq \epsilon \ ) \ \}, \end{aligned} \tag{13}$$

where p is the traffic peak rate. This expression cannot be used directly, because it is given as a test on infinitely many pairs (n, r). To calculate $Nmax$, we first rewrite the last inequality in (13) as

$$n \leq \nu(r) \tag{14}$$

where

$$\nu(r) = \frac{chr - q}{\Lambda(r)}, \tag{15}$$

$$q = \log(\frac{L}{\epsilon}), \tag{16}$$

and

$$\Lambda(r) = \log \mathrm{E}e^{r\bar{A}_1(h)}. \tag{17}$$

This procedure is based on finding the value $r = r_1$ that maximizes $\nu(r)$. $Nmax$ is the integer part of $\nu(r_1)$, unless the peak-rate inequality $np \leq C$ in (13) is decisive.

This search for r_1 proceeds in a binary manner (each step halving the interval of possibilities). This is legitimate because the function $\nu(r)$ is unimodal (the global maximum being equal to the only local maximum), which is a consequence of the convexity of Λ. At each step, for a value r in the interval of possibilities, the procedure evaluates the sign of $\nu'(r)$ and decides whether r_1 is above or below r (e.g., a positive derivative shows that $r_1 > r$). $\nu'(r)$ can be calculated from (15) and

$$\Lambda'(r) = \frac{\mathrm{E}\bar{A}_1(h)e^{r\bar{A}_1(h)}}{\mathrm{E}e^{r\bar{A}_1(h)}}, \tag{18}$$

which, in turn, can be calculated from expectations of $\bar{A}_1(h)e^{r\bar{A}_1(h)}$ and of $e^{r\bar{A}_1(h)}$ with respect to the distribution of $\bar{A}_1(h)$.

Note that $\nu(r)$ is positive only for $r \geq r_0 = q/ch$, and, therefore, we can start the search for r_1 knowing that r_1 is in the initial interval $[r_0, \infty]$. The infinite end point is not a problem because $\nu(\infty)$ and the sign of the derivative $\nu'(r)$ for r near ∞ are both well defined, as follows. First, $\nu(\infty) = c/p$ because ph is the peak value of $\bar{A}_1(h)$. Second, although $\nu'(\infty)$ does not exist, the sign of $\nu'(r)$ for r near ∞ is the same as the sign of

$$pq + c\log(\Pr(\bar{A}_1(h) = ph)), \tag{19}$$

as can be seen by differentiating (15). As a result, it is easy to recognize when $r_1 = \infty$. More explicitly, if (19) is non-negative, then our search ends immediately with $r_1 = \infty$; otherwise, we must search the interior (r_0, ∞).

When the remaining interval of possibilities is still infinite, our procedure is usually to choose $2l$ for a "midpoint" of $[l, \infty]$. The only exception is the choice in the first interval $[r_0, \infty]$. Instead of choosing $2r_0$ as the midpoint, we choose $r_0 + \sqrt{r_0^2 + 2r_0 m/v}$ if $v > 0$, where m and v are the mean and variance of the distribution of $\bar{A}_1(h)$. (This is exactly the maximizing value r_1 if $\bar{A}_1(h)$ is Gaussian.)

In principle, the binary search can quickly determine (to machine precision) the maximizing value r_1. We stop the search as soon as we can tell that, throughout the interval of remaining possibilities, $\lfloor \nu(r) \rfloor$ could never exceed a value already found. Hence, this value is $Nmax$, and there is no need to identify r_1 with any greater precision. To recognize this event, i.e., to determine quickly a helpful upper bound on $\nu(r)$ over an interval $[r_{low}, r_{high}]$, we rely on the monotonicity and convexity of Λ. Thus, within a finite interval, we rely on Λ's monotonicity to observe that

$$\nu(r) \leq \frac{chr_{high} - q}{\Lambda(r_{low})}. \tag{20}$$

For $r_{high} = \infty$, we rely on Λ's convexity and the fact that $\Lambda(0) = 0$ to observe that

$$\nu(r) \leq \frac{chr_{low}}{\Lambda(r_{low})}. \tag{21}$$

In summary, the major task in this procedure for calculating $Nmax$ is to determine r_1. After testing the trivial possibility that $r_1 = \infty$, we begin a binary search for r_1 with $[r_0, \infty]$ as the initial interval. For each step of the search, we test a "midpoint" of the current interval, thereby determining a smaller interval for the next step. For finite intervals, the tested "midpoint" was the true midpoint. For infinite intervals, it is usually twice the lower end point. We terminate the search as soon as we find that one of the bounds (20) or (21) can assure us that $Nmax$ has been determined.

3.2 Nearly ideal efficiency at large multiplexing levels

In this section we elaborate on the earlier claims of asymptotic efficiency for Cases III–V of traffic knowledge. Note first that the CAC policies we associated with these cases may be applied to a variety of settings. In particular, they can be applied to settings with different values of the bottleneck's service rate C. In any one setting, a CAC policy will determine a maximum number n of streams of a given type that it will admit. As noted earlier, $\rho = nm/C$ may be identified as the efficiency of the policy, where m is the mean rate of an individual stream. A natural question to consider is how n and ρ change in various settings, especially in sequences of settings in which C grows while the traffic associated with individual streams remains the same. In every such sequence, as long as B grows in proportion to C (hence, h can remain constant), the number n of connections that are admissible by any of our CAC policies grows. Thus, the multiplexing level increases. More significantly, for any of our CAC policies corresponding to cases III-V, the efficiency ρ approaches 1 as $C \to \infty$. Strictly speaking, $nm/c \to 1$ while $c/C = 1 - B/CX$ remains a constant strictly less than 1. Hence, ρ rises only to this constant, which is not quite ideal efficiency. Nevertheless, the earlier claims are correct for the asymptotic efficiency of CAC policies based on partial traffic knowledge. The reason is that slight modifications to our CAC policies achieve the ideal asymptotic efficiency ($\rho \to 1$) while continuing to rely on the same traffic information. (Instead of fixing $X = D$, the modifications choose X to grow to infinity with C.) Indeed (with the slight modifications) the inefficiencies $1 - \rho_{IV}$ and $1 - \rho_V$ approach zero at the same rate, i.e., their ratio approaches 1.

These claims of asymptotic efficiency can be proved from observations concerning the expression (15) for the quantity $\nu(r)$ giving a lower bound $\lfloor \nu(r) \rfloor$ for n. Since $n \to \infty$, the discussion here can ignore the difference between $\nu(r)$ and $\lfloor \nu(r) \rfloor$. To this accuracy, then, the efficiency of our CAC policies is given by

$$\rho \approx \sup_r \frac{\nu(r)m}{C} = \sup_r \frac{mhr\frac{c}{C} - \frac{qm}{C}}{\Lambda(r)} = \sup_r \frac{mh\frac{c}{C} - \frac{qm}{rC}}{\Lambda(r)/r}. \tag{22}$$

The key observations are

$$\Lambda'(0) = \mathrm{E}\bar{A}_1(h) = mh \tag{23}$$

for the CAC policies that take advantage of knowledge of the mean, and

$$\Lambda''(0) = \mathrm{var}(\bar{A}_1(h)), \tag{24}$$

which is the variance σ^2 of $A_1(h)$ in both *CACname = true* and *CACname = PMV*. In particular, as $r \to 0$, $\Lambda(r)/r \to mh$. In finer detail for the policies that "know" σ^2, $\Lambda(r)/r - mh \sim \sigma^2 r/2$ as $r \to 0$.

If one chooses values r_C approaching zero slowly enough that Cr_C still approaches infinity, then (22) implies that

$$\rho \geq \frac{mh\frac{c}{C} - \frac{qm}{Cr_C}}{\Lambda(r_C)/r_C} \to 1 \tag{25}$$

as $C \to \infty$. (Strictly speaking, the limit for the unmodified policies is $c/C < 1$. For modified policies in which $c/C \to 1$, the limit is indeed 1, but with q also growing with C the requirement on r_C is that $q/Cr_C \to 0$, which can always be met as long as X does not approach infinity as fast as e^C.) As for the exact rate at which the inefficiency approaches zero, note that if the maximizing values of r also vary with C by approaching zero, but slowly enough that q/Cr_C also approaches zero, then the inefficiency is given by

$$1 - \rho = \frac{\Lambda(r_C)/r_C - mh\frac{c}{C} + \frac{qm}{Cr_C}}{\Lambda(r_C)/r_C} \sim \frac{mh(1-\frac{c}{C}) + \frac{\sigma^2 r_C}{2} + \frac{qm}{Cr_C}}{mh}, \tag{26}$$

provided $\bar{A}_1(h)$ has the same variance as $A_1(h)$. It turns out that the maximizing values of r behave in the required way. Thus, the inefficiencies of *CACname = PMV* and of *CACname = true* match each other when C is large. In particular, for the modified versions, the inefficiencies approach zero at the same rate.

4 Traffic traces used for comparisons

For the comparison of various CAC policies, we used two genuine video and data traffic traces.

Video Traffic: The video-conference trace shown in Figure 1 contains the "real data" of a teleconference showing talking heads and shoulders, used in previous studies [11]. The trace lists the number of 64-octet (64-byte) cells in each of approximately 48 500 frames. It corresponds to approximately 30 minutes of a video at 24 frames/second.

Data Traffic: The data traffic trace shown in Figure 2 was acquired by measuring TCP/IP traffic running over a HIPPI network at the Pittsburgh Supercomputing Center at Monroeville, PA, used for storage and retrieval of data files [19]. The TCP Maximum Transmission Unit (MTU) was set to 64 Kbytes. The selected traffic session lasted 1.972×10^4 seconds (5.48 hours). In addition to the

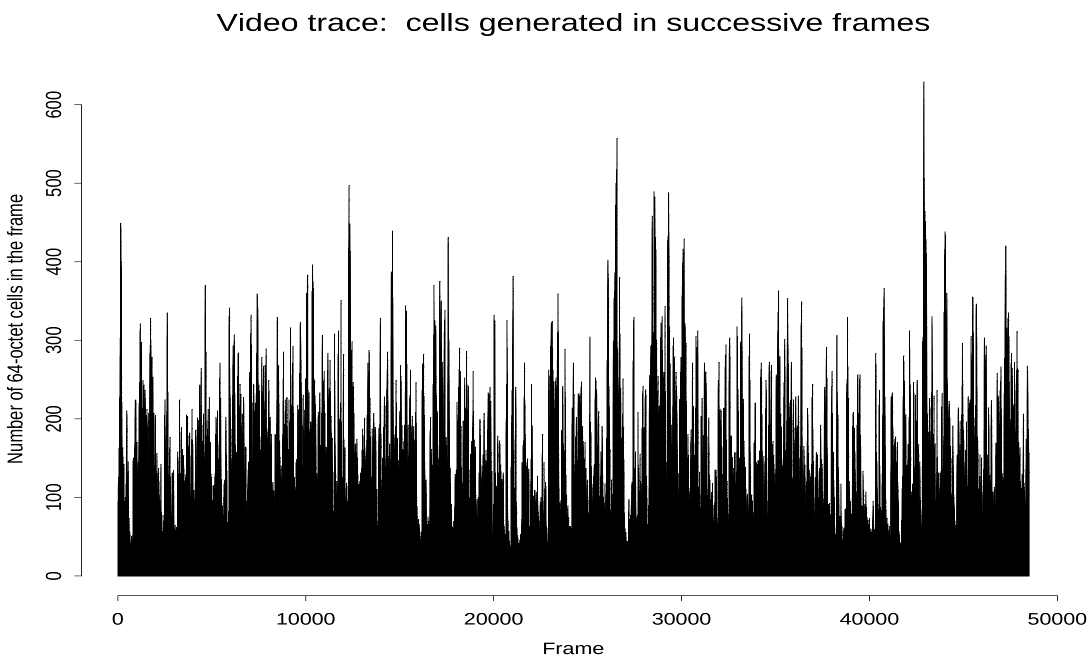

Figure 1: Traffic pattern of the video trace (time scale = 1/24 second). The trace lists the number of 64-octet (64-byte) cells in each of approximately 48 500 frames. It corresponds to approximately 30 minutes of a video at 24 frames/second.

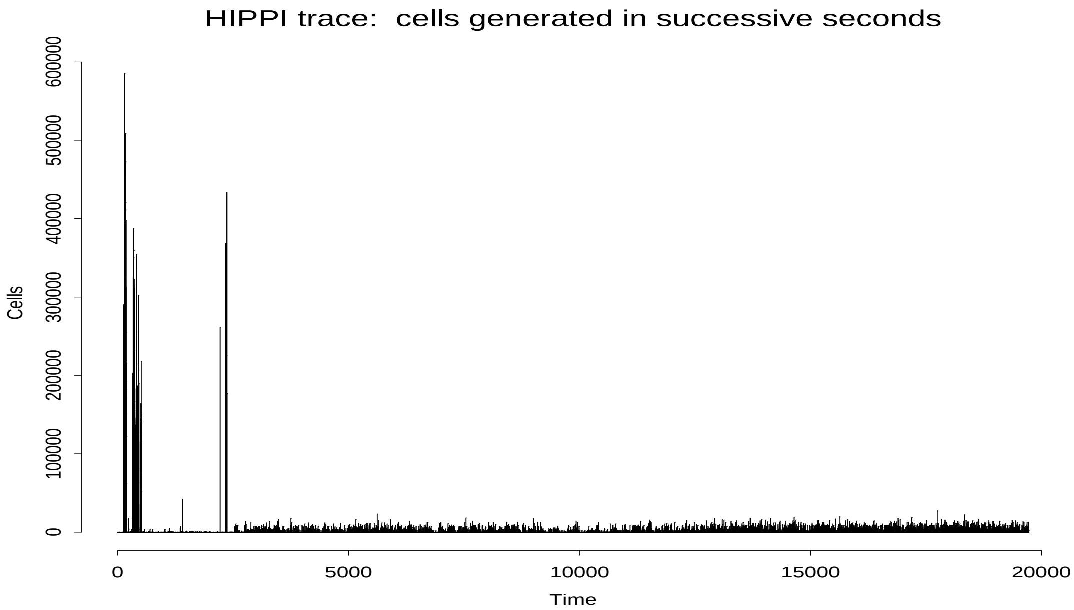

Figure 2: Traffic pattern of the TCP/IP-over-HIPPI data trace (time scale = 1 second). The TCP Maximum Transmission Unit was 64 Kbytes and the session lasted 1.972×10^4 seconds (5.48 hours). In addition to the data traffic, the trace includes small signaling packets and the acknowledgment packets. The trace shows large variability in the packet inter-arrival times and packet sizes.

data traffic, the trace also includes rather small signaling packets and the acknowledgment (ACK) packets. The trace shows large variability in the packet inter-arrival times and packet sizes.

The two traffic traces were used to explore the behavior of network bottlenecks carrying independent traffic streams. The situations we considered were always homogeneous, i.e., the different traffic streams always shared the same statistical properties. (We multiplexed various numbers of video traces and data traces separately, and we never mixed a video and a data trace.) The CAC policies we considered all anticipated independent random traffic-generation processes, but the two sample traces are clearly deterministic histories. To create a random process from a deterministic trace of length T, we extend the trace to be periodic with period T, and then select a starting time S at random from $[0, T)$. The random amount of arriving traffic during an interval $[a, b)$ is then $A_1([a, b)) = A_t([a + S, b + S))$, where A_t is the periodic extension of the deterministic trace. For n independent streams, we choose n independent starting times. Note that with this construction, the arrival process of any particular stream is a stationary random periodic process, and the arrival processes of the distinct streams are independent and identically distributed. Both the analytical calculations described in Section 5 and simulations described in Section 6 refer to the same traffic-generation processes constructed in the described manner.

The network bottlenecks, whose behavior we explored, differed from the media from which the traces were obtained. We explored ATM links that carry information in 53-byte cells that contain 5-byte cell headers (overhead) [17]. However, the video trace we used reported traffic in 64-byte cells, while the data trace reported traffic in variable-length packets. Moreover, we assumed earlier a server with continuous traffic with service rate C, as opposed to the more realistic ATM link serving discrete cells.

It was rather elementary to convert the video trace into a record of arrival times of ATM cells. With the fixed rate of 24 frames/second, each frame represented an interval of time of length 1/24 of a second, i.e., frame f represented the time interval $[\frac{f-1}{24}, \frac{f}{24}]$. If the video trace reported that frame f consisted of b (big) 64-byte cells, then the $64b$ bytes of information were imagined as being distributed among the 48-byte payloads of $\gamma(b) = \lceil 64b/48 \rceil$ successive ATM cells. (We choose a payload size of 48 bytes to conform with an earlier study [11].) These $\gamma(b)$ cells would be transmitted uniformly over the interval corresponding to frame f. More explicitly, for $1 \leq k \leq \gamma(b)$, we imagined that ATM cell k of frame f arrived at the bottleneck at time $\frac{f-1}{24} + \frac{k}{24\gamma(b)}$. This uniform manner of transmission is consistent with a conception of a video coding of pictures into cells that is designed to keep cell traffic smooth within each frame while maintaining a fixed frame-delivery rate. In this manner the raw video trace was converted into a trace of ATM-cell arrivals.

For the data trace the procedure was more involved, partly because even the raw traffic had a nontrivial structure. The raw trace specified for each HIPPI packet a time t at which the packet began to pass the trace-collection point on the HIPPI interface, and the packet size b in bytes. A large packet is composed of a sequence of HIPPI bursts, each burst limited to 1024 bytes. Given a HIPPI packet of b bytes, there are $Bu = \lceil b/1024 \rceil$ bursts in the packet, of which the first $bu = \lfloor b/1024 \rfloor$ bursts are full. If $Bu \neq bu$, then the last burst carries $\beta = b - 1024 \times bu$ bytes.

To convert the HIPPI trace to a specification of arrivals of ATM cells, we transformed each burst separately into the cells carrying an AAL4 (ATM Adaptation Layer 4) packet. Given a burst consisting of β bytes on the HIPPI interface, we first accounted for the convergence-layer overhead for carrying a HIPPI burst as an AAL4 packet. Specifically, for each byte we use 1 additional bit for a parity control, and add 4 bytes for the Length/Longitudinal Redundancy Check (LLRC). This number is then partitioned into the 44 payload bytes of successive cells. Thus, a burst of β bytes corresponds to $\lceil (\lceil 9 \times \beta/8 \rceil + 4)/44 \rceil$ cells. (Note that a full parity byte will be added for any segment of less than 8 bytes.) In particular, each full burst of 1024 bytes is converted into 27 cells.

The timing of the ATM cells was determined by assuming that the cells were created at the end of each burst. This assumption is consistent with an arrangement in which the traffic first traverses a HIPPI interface to reach a node that both converts the arriving HIPPI traffic into cells and manages the outgoing bottleneck link. The HIPPI interface operated at the speed of 800 Mbits/second, or 10^8 bytes/second. The interface was not serial and carries 4-byte words in parallel. Thus, a HIPPI packet of b bytes, that begins to cross the trace-collection point of a HIPPI interface at time t, will complete the crossing at time $t + 4\lceil b/4 \rceil \times 10^{-8}$. We assumed that this time is the creation time of the cells corresponding to the last burst of the HIPPI packet. As for the other bursts, because they are full bursts of size 1024 bytes (a multiple of 4 bytes) they completed their crossings on a uniform schedule. Explicitly, full burst k of the packet that began at time t, ends its crossing at time $t + 1024k \times 10^{-8}$. Accordingly, we assumed that 27 cells arrived simultaneously at each of these ending times of full bursts, and that the cells corresponding to the last burst all arrived at the end time of the HIPPI packet. This is how the raw data trace was converted into the specification of a trace of ATM-cell arrivals. The converted ATM-cell trace is shown in Figure 2.

Finally, we give an account of the mismatch between the calculations for a continuous server and the more realistic discrete server of ATM cells. The discreteness of a realistic server emphasizes several concerns that complicate the "simple" argument greatly, but have numerically minor effects.

- The server can be idle even though the queue has waiting cells. An obvious reason is that the first cell might arrive in the middle of an outgoing cell slot. More subtly, especially with the complexities of all the tasks that might be demanded of an ATM queue handler, the time of decision for whether an arriving cell can enter the queue or not might precede the time of its availability for transmission. (The few processor cycles required between deciding to admit the cell and actually getting it a buffer for forwarding on the link might extend across a cell-slot boundary.)

- Another subtlety is that a full buffer can block an arriving cell, even though part of the space is occupied by a cell that has already been transmitted. (There may be a few processor cycles between the end of the cell slot carrying the cell and the time when the cell's space in the buffer is marked, as being eligible for overwriting in a form that can be recognized by the process that decides whether to drop arriving cells.)

The subtle versions of these effects might be eliminated by careful hardware design. Yet, the same effects arise whenever cell slots do not all have the same duration, whether due to a clock's variation or whether due to a lower-layer protocol. For instance, if the ATM cells are carried over a SONET link, a small fraction of the cells will cross various overhead "boundaries," which has the effect of lengthening these cell slots.

One can account for all these complications simply by introducing a small number a of cells to represent the "ambiguity". Numerically, the procedure is to replace the conditions $(C - c)X \geq B$ and $2ch \leq B$ by $(C - c)X \geq B + a$ and $2ch \leq B - a$. The intuition underlying these changes is that, in contrast to the earlier considerations of a busy interval (u, v) with $Q > 0$ in which the continuous server was effectively making room in the buffer at rate C, the more realistic server can spend some of this time in idleness and waste the last part of the time in failing to clear cells from the buffer that have already been transmitted. The difference between $C(v - u)$ and the number of cells actually cleared from the buffer is never more than a few cells. The integer a is a bound on this difference. The modified condition $(C - c)X \geq B + a$ ensures that the prolog empties the queue, in spite of the complicating effects. Similarly, the modified condition $2ch \leq B - a$ ensures that, with average arrival rates bounded by c on a sequence of non-overlapping intervals of length h, no queue overflows can occur once the queue empties. In our calculations we use $a = 2$ for the following reasons. Obviously, once the server is discretized into cell slots, a must be at least one cell, to be a bound on the difference between $C(v-u)$ and the number of cells actually cleared. We take a to be larger than 1, because we do not expect all the subtle effects to be negligible. Because we expect all of them to be small, we choose $a = 2$.

5 Analytical calculations of efficiencies of CAC policies

The analytical calculations required knowledge of the distribution of $A_1(s)$, the random amount of traffic arriving on a single stream in a time interval of length s. For a trace of length T, the definition of the arrival distribution on the time scale s is that for any value a, $\Pr(A_1(s) \leq a)$ is T^{-1} times the Lebesgue measure of the set of times t such that the arrivals in $[t, t+s)$ did not exceed a. For these calculations, however, instead of computing the distribution of $A_1(s)$ exactly, we partitioned the total interval T into T/s intervals of length s and we pretended that $\Pr(A_1(s) = a)$ was the fraction of these intervals in which the amount a of traffic arrived. In other words, we used a discrete approximation to the true distribution. (The true distribution of the *number* of cells arriving in an interval is also discrete, but it is much finer than the coarse approximation we employed.)

We have examined efficiencies (occupancies) ρ of a link when it is "filled" with sessions admitted according to various CAC policies. The variables identifying each traffic situation were the type *TrafficDescription* of traffic to be carried, the link's speed C and its buffer size as given through h, and the service standard to be met, given by parameters D and ϵ. As described earlier, in various cases identified by these parameters, a general policy $CACname$ will determine a maximum number $Nmax$ of simultaneously admitted sessions. When the link is "filled" with sessions according to this policy, the link will be operating at the efficiency

$$\rho = Nmax * meanrate(\mathit{TrafficDescription})/C.$$

The results of several efficiency calculations are shown in Figure 3 and Figure 4 for the video-conference and the HIPPI traces, respectively. Each curve corresponds to a separate CAC policy and it shows the efficiency ρ vs. the link speed C, while $CACname$ and the remaining traffic parameters are held fixed. The values of the fixed traffic parameters are indicated in the graph titles. The link speeds we explored are indicated on the horizontal axis, where the values of buffer size B (in cells) and link speed C (in cells/second) shown in the right bottom corner are for the OC-48 case. Because we began this investigation with the HIPPI trace, the graphs include a tick mark labeled "HIPPI" at the HIPPI speed. The link speeds are listed in Table 2. The horizontal line corresponding to efficiency $\rho = 1$ is marked with tick marks at various points $(C, 1)$ labeled

$$n = \left\lfloor \frac{C}{mean(TrafficDescription)} \right\rfloor$$

indicating how many sessions would keep the link (almost) fully occupied.

Because we were most interested in the effect of traffic knowledge on one time scale, each graph displays the results of a large number of cases, all with the same choice of h. Of course, this also implies that the calculations are partly done in the reverse order: instead of determining h from B, C, and D, as would be natural in setting up the CAC policy for a given link, h is fixed first,

Name	Speed (cells/second)
DS3	96,000
OC-1	116,830
OC-3	353,208
OC-6	707,774
OC-12	1,416,910
HIPPI	2,636,719
OC-24	2,835,170
OC-48	5,671,700

Table 2: Link speeds used in the study.

and then B is determined. As noted earlier, the necessary conditions between B and h for the conservative CAC policies (to be justified as conservative through the calculation (1)–(6)), are that $(C-c)X \geq B+a$ and $2ch \leq B-a$. With $X = D$ and $a = 2$, if one examines all the pairs (B, c) consistent with these conditions, one finds the pair with the greatest value of c to have B equal to

$$\frac{2Ch+4}{1+\frac{2h}{D}} - 2. \tag{27}$$

Accordingly, we set the buffer size B to be the ceiling of this expression. As advertised, $B \approx 2Ch$.

We explored numerous cases, but the two graphs shown in Figure 3 and Figure 4 seem to be the most interesting. The values $D = 180$ seconds and $\epsilon = 0.01$ were inspired by a nearly traditional standard of 1% (0.01) blocking for ordinary phone calls that last for an average of 3 minutes (180 seconds). The service standard we are exploring here with the analytical calculations and simulations is that only 1% of 3-minute segments of admitted connections should be disrupted by cell losses. We also used other values for D and ϵ without seeing much of an effect.

For a video-conference trace, we found that nearly all the multiplexing gains could be achieved if we knew the traffic peak, the mean, and the variance on a single time scale (Case IV degree of traffic knowledge employed by policy $CACname = PMV$). Moreover, the graph indicates that reasonable efficiency can be achieved with only a 2-millisecond buffer. Note that if knowledge of the variance is neglected, i.e., if the policy $CACname = PM$ is used, the efficiency is significantly lowered.

For data traffic, however, even with large 2-second buffers the conservative CAC policies all yield rather small efficiencies well below both the ideal efficiency of 100% and the efficiencies produced by the nonconservative CAC policies. Furthermore, the efficiencies of the nonconservative CAC policies vary over a wide range. In other words, these analytical estimates are not very suggestive of the efficiency of the ideal CAC policy that would account for all the features of the data traffic that could ever be described. Therefore, to estimate the efficiency of this ideal CAC policy we performed simulations.

6 Simulations of service quality

We used simulations to estimate service quality for an ideal policy and to explore a specific case considered by the analytical calculations. The case that drew our attention was the OC-24 link shown in Figure 4. It concerns the decision of whether to admit a number of traffic streams with

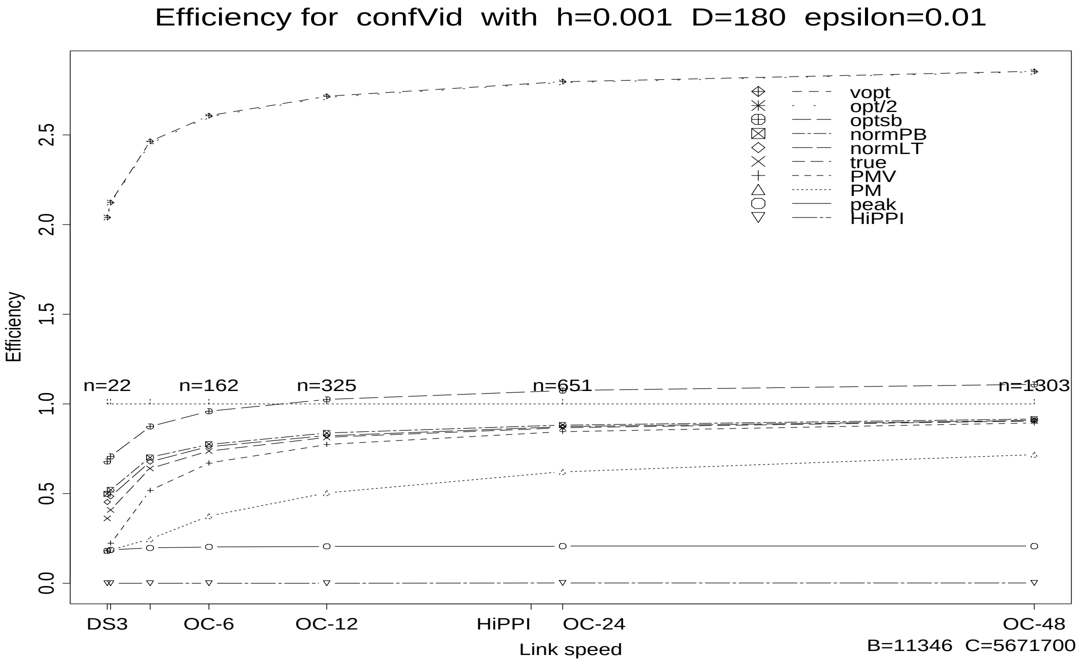

Figure 3: Efficiency vs. link speed for video-conferencing traffic. Each curve corresponds to a separate CAC policy. Nearly all the multiplexing gains could be achieved by knowing the traffic peak, the mean, and the variance on a single time scale.

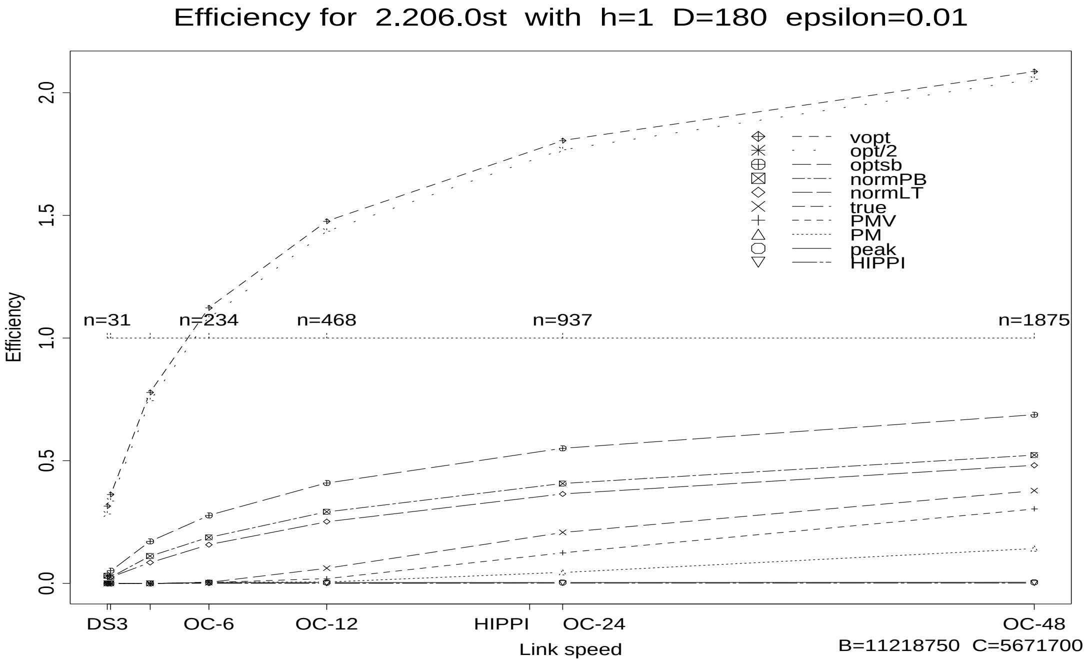

Figure 4: Efficiency vs. link speed for HIPPI traffic. Each curve corresponds to a separate CAC policy. The conservative CAC policies all yield rather small efficiencies well below both the ideal efficiency of 100% and the efficiencies produced by the nonconservative CAC policies.

the characteristics of the HIPPI trace, when the link has a buffer of roughly 2 seconds and the QoS target is $D = 180$ and $\epsilon = 0.01$. This case drew our attention for the following reasons:

- The policy *CACname* = *true* allowed 195 simultaneous connections, yielding an efficiency of 21%. This was high enough to show that significant efficiency is achievable with a rather simple characterization of the traffic, and was low enough to suggest that a much greater efficiency was conceivable.
- The nonconservative policies *CACname* = *normLT*, *normPB*, and *optsb* produced a rather wide scatter of efficiencies, all higher than produced by the policy *CACname* = *true*, suggesting that the ideal policy would produce a similarly impressive efficiency.

We did not conduct a thorough test to identify precisely the number of connections that the ideal policy would admit. We first discuss the tests that should have been performed if we were precise: Consider a number n of connections that one might admit, and let p_n be the probability that a given connection loses traffic during a given 3-minute interval. By definition of the QoS target, the ideal policy would admit n connections only in case $p_n \leq 1\%$. In any single simulation run l with n arriving streams competing for a simulated link, one can prescribe a sequence of 3-minute service-monitoring intervals and observe the fraction $\hat{p}_n(l)$ of pairs (k, I) for which stream k loses cells during interval I. The traffic was derived from a single trace 5.48 hours long. In a single run we monitored more than 100 adjacent 3-minute intervals. Of course, the results from adjacent intervals were highly dependent, and the results from different streams were often identical. Hence, an accurate estimate of p_n from observations of $\hat{p}_n(l)$ would require a very large number of simulation runs.

If we wanted to identify $Nmax$ exactly for the ideal policy, we should have searched through all the numbers $n \in (195, 937)$ until we found a pair $(n, n+1)$ with $p_n \leq 1\% < p_{n+1}$. Moreover, in verifying this criterion, we should have made a large enough number of runs so that the observations of the various $\hat{p}_n(l)$ and $\hat{p}_{n+1}(l)$ would determine p_n and p_{n+1} precisely enough to confirm the comparison $p_n \leq 1\% < p_{n+1}$ with confidence. Instead, we tried only a few values of n, in hope that we would find a value for which an average of observations $\hat{p}_n(l)$ over a number of runs l would be close to 1%. We began our first trials by considering the OC-24 case shown in Figure 4. We selected a prominent point corresponding to *CACname* = *optsb* that allowed 517 connections to be admitted, with the resulting efficiency 55%. (The point perhaps seemed prominent because it was close to the midpoint between *CACname* = *true* and 100% efficiency.)

The simulation runs for the case $n = 517$ were sufficient for our modest purposes. We found the average over 50 runs of the values $\hat{p}_{517}(l)$ to be 0.94%. Moreover, although the roughly 5000 pairs (k, I) considered in each run were highly dependent, they were not identical and supplied somewhat independent information, especially from separated intervals I. More precisely, instead of $\hat{p}_{517}$ over 50 runs having a sample variance near 0.99/5000 that would be expected for 50 independent indicators of 1% events, the sample variance was closer to 0.99/150000, as if each run were supplying about 30 independent estimates of p_{517}. Therefore, we terminated simulations, having concluded that the ideal policy would admit a number of connections that is much closer to 517 than to 195.

One reason to terminate simulations was that each run lasted a long time, in spite of the two short cuts that we took to speed up the simulation: we changed the nature of both the queue server and the arriving traffic. We treated the traffic as a fluid instead of a sequence of discrete cells (though traffic was still measured in "cells"). For the queue server, instead of keeping track of each cell slot, we simply forwarded fluid traffic at the rate $C = 2,835,170$ cells/second of an OC-24 link as long as the queue size was positive. (While the queue size was zero, we forwarded traffic as it arrived, as long as the arrival rate remained below C.) For the arriving traffic, we did not identify every instant at which a HIPPI burst ends and a corresponding burst of cells arrives instantaneously at the queue. Instead, we assumed that a c-cell HIPPI packet would arrive as a fluid at the HIPPI rate of $27/1024 \times 10^8$ cells/second. The interval over which the fluid from a packet would arrive was assumed to commence at the time recorded in the raw trace for the packet's first bit. Finally, whenever the buffer filled and fluid continued to arrive faster than the service rate, the excess was "spilled" as lost cells. Furthermore, the losses were assigned equally among the sessions with packets in progress.

Because of the short cuts, the system we simulated was not the original system of interest. Nevertheless, the simulation results are still very relevant, especially for the estimation of the total lost traffic. First, in converting the queue server from a discrete to a continuous version, a total error of at most one cell will be made in each busy period. Next, in converting arrivals from instantaneous bursts, the maximum total error is $517 \times 27 = 13,959$ cells in any busy period. These errors are very small in comparison with the total buffer size of 5,608,031 cells. Although we could anticipate that the total measures of lost traffic is quite accurate, the justification is weaker for the accuracy of the attributions of losses to individual streams. Nevertheless, to the extent that a loss episode typically lasts much longer than the time for generating a single HIPPI burst, it is natural to expect that all active sessions will experience losses during such an episode.

7 Discussion of the simulation results

As anticipated, the simulation results indicated that an ideal CAC policy could allow much greater efficiency than any of the provably conservative policies. Specifically, for data streams with the traffic characteristics of the HIPPI trace, while $CACname = true$ would restrict an OC-24 link with a 2-second buffer to admitting only 195 independent streams (21% efficiency), the simulations indicated that the link could handle 517 streams (55% efficiency). Strictly speaking, the simulations merely showed that the maximum number of streams consistent with good service is close to 517, but this distinction is unimportant for the rest of the discussion. After contrasting these observations for the data traffic with the results for the video traffic, in this section we then discusses reasons for the inefficiency of policy $CACname = true$. The main reason is identified to be the failure to account for traffic information on multiple time scales. Then, with the intention to help the construction of an efficient CAC policy, in the rest of the section we attempt to identify the most relevant time scales (beyond the half-buffer work time already incorporated into policy $CACname = true$). The relevant time scales, of course, are those at which traffic knowledge would have to be incorporated if a conservative CAC policy is to be efficient.

The results for the data trace differ dramatically from those for the video trace. The calculations with $CACname = true$ show that a 74% occupancy can be achieved with video conferencing merely by using an OC-6 link to multiplex 120 sessions with a buffer of only 1,418 cells to handle bursts. In contrast, to achieve noticeable multiplexing gains with the data traffic, the higher-capacity OC-24 link was required (to allow more sessions in the statistical mix), and a buffer of 5,608,031 cells ($\approx$ 2 seconds of work) was required to absorb the bursts. Even then, the occupancy achieved in multiplexing 517 sessions was only 55%. For clarity, these observations are repeated in Table 3. Note that these differences reflect the nature of the traffic: the data traffic is tremendously bursty and even an ideal CAC policy cannot handle it as efficiently as the video traffic.

Another difference between the two traces concerns whether the attainable efficiency can be realized with a simple CAC policy. The 74% occupancy with the video trace is achieved by the

Traffic trace	Link speed	Buffer size	Nmax	Efficiency (%)
Video	OC-6	1,418	120	74
HIPPI	OC-24	5,608,031	517	55

Table 3: Efficiencies achieved through simulations for the video and HIPPI traces.

policy *CACname = true*. Indeed, the simpler policy *CACname = PMV* achieves 67%, which is not much smaller. In contrast, with the data traffic, while the ideal policy would achieve the modest occupancy of 55%, the most efficient of the conservative policies *CACname = true* realizes the strikingly smaller occupancy of 21%. Evidently, traffic knowledge on a single time scale can be used rather efficiently for the video trace with the policy *CACname = true*. Yet, for the data trace, the same policy *CACname = true* fails to be efficient.

In spite of this comparison, one should not overlook the fact that the conservative policies *CACname = true*, *PMV*, and *PM* provide dramatic increases in efficiency over peak-rate policies. Specifically, in contrast to the traffic-peak-rate efficiency of 0.4%, they all yield a gain of more than an order of magnitude: *CACname = PM* yields 4.5%, *CACname = PMV* yields 12.5%, and *CACname = true* yields 21%. These yields look even better against the efficiency 0.1% of the physical-peak-rate policy. Clearly, the efficiency of *CACname = PMV* of more than 10% is already within an order of magnitude of the ideal policy. Nevertheless, with simulations suggesting a significantly greater ideal efficiency of about 55%, the pursuit of a better policy seems worthwhile. The *PMV-true* comparison indicates that beyond the peak, mean, and variance, additional traffic details at the half-buffer scale can contribute substantially to efficiency. Yet, as noted earlier, this contribution still falls short of the ideal. Assuming this shortfall is due to the lack of traffic knowledge, and not to its inefficient incorporation within policy *CACname = true*, the conclusion is that an efficient policy must incorporate additional traffic knowledge if it is to handle traffic such as the data trace. In other words, when dealing with bursty traffic, it is imperative to understand traffic on more than one time scale.

Before investigating other relevant time scales, it is appropriate to investigate the deficiencies in the argument above, i.e., to consider the possibility that *CACname = true* uses its traffic knowledge inefficiently. Recall that the simulations showed that policy *CACname = true* is inefficient: it could justify only 195 data streams, while the QoS target was met when 517 data streams were multiplexed. Obviously, with interpolations in the calculation (1)–(6), the reason for the greater efficiency obtained by simulations is that at least one of the calculations is too inefficient. Hence, the question is "Which one?" At first sight, it might be that the responsible inequalities are (3)–(5) that calculate a simple bound on a tail probability. In general, the replacement of $A_1(h)$ by $\bar{A}_1(h)$ in (5) could introduce some inefficiency, but not for the case *CACname = true* in which $\bar{A}_1(h) = A_1(h)$. The only other inequality in the simple bound (3)–(5) is (3). Yet, the results for the video trace suggest that this inequality does not introduce a significant inefficiency. (The suggestion is weak because the same results also suggest that policy *CACname = true* would be as efficient as the ideal policy.) For an independent estimate of the inefficiency introduced by the inequality (3), one should compare the points in Figure 4 for the OC-24 case with the two policies *CACname = normLT* and *CACname = normPB*. The fact that these two points are very close supports the idea that the inequality (3) is not responsible for much inefficiency. (Ultimately, this question concerns a property of the distribution of $A(h)$, a 517-fold convolution of the distribution of $A_1(h)$. The evidence presented so far only implies that two other distributions agree well with the inequality.) These considerations lead to the conclusion that the responsibility is with inequalities

(1) and (2), i.e., that our conservative CAC policies ignore other expressions for queue-overflow events in terms of joint patterns of arrivals, and that they ignore the possibility of massive overlaps of all the excess-arrival events appearing in (1). In other words, the conclusion is that significantly greater efficiency can be obtained only by taking advantage of traffic information on more than one time scale.

The extra information, that can help a policy be efficient, concerns cancelations of traffic bursts. Thus, recall the types of events $\{A(h) > ch\}$ entering the calculations defining $CACname = true$. The fact that $CACname = true$ allows only 195 data streams to be multiplexed, means that when 517 streams share the link, many intervals of length h occur in which arrivals exceed link capacity. Specifically, they occur often enough to "frighten" the policy $CACname = true$. In spite of these intervals with frighteningly many arrivals, the service standard is met. Evidently, on most of these occasions, in which arrivals exceed link capacity on one interval of length h, the queue does not overflow because arrivals in neighboring intervals are below link capacity. In other words, bursts of excess arrivals in some intervals are cancelled by the lack of arrivals in neighboring intervals. Realization of an efficient admission-control policy, therefore, requires quantitative knowledge of these cancelations between arrivals in neighboring intervals.

The required information is partly at odds with intuition of how traffic arrivals correlate across neighboring intervals. Thus, it is natural to expect that the arrivals in one interval would be positively correlated with the arrivals in the next interval. Stated roughly, if the arrival rate is high now, it is likely to be high for some time to come. Yet, this type of information is not what is required to develop an efficient CAC policy. Of course, the two types of information are compatible: neighboring intervals can have positively correlated arrivals, with a high arrival rate likely to be sustained. Nevertheless, in the same process, whenever the arrival rate becomes so high as to exceed the link capacity in one interval, and is likely still to be rather high in the next interval, the likely high values will not be high enough to continue to exceed link capacity. The information required by an efficient CAC policy is this detailed type of information that cancelations occur in spite of positive correlations.

7.1 Relevant time scales

Having concluded that significantly greater efficiency can be obtained only with traffic information on more than one time scale, the question now becomes, "What other time scales are most relevant?" Identifying relevant time scales is difficult and we could only determine some rather loose bounds. The first time scale, naturally, is the time needed to serve half a buffer (≈ 1 second in the case of multiplexed data traffic that we considered). For other time scales, we could only determine that it was necessary to understand the traffic on some time scales between 10 seconds and 6 minutes, including at least one time scale in the range of 10–20 seconds. In other words, beyond basic traffic knowledge on the time scale of 1 second, efficiency would be aided significantly by additional traffic knowledge on a time scale near 15 seconds and on other scales. Efficiency would not be aided significantly by knowledge on time scales larger than 6 minutes. In particular, it is important to understand traffic over a range covering more than an order of magnitude (from 1 second to more than 10 seconds), and very likely close to two orders of magnitude (from 1 second to 6 minutes = 360 seconds).

How do we know that time scales longer than 10 seconds matter? In one of the many analytic calculations of the efficiency that we performed, we found that only 410 simultaneous sessions could be justified by $CACname = normPB$ when the buffer size was increased by a factor of 10, while the link capacity remained unchanged. In this case the half-buffer work time for the larger buffer was 10 second (10 times as large). (This is the reason why this result does not appear in Figure 4.) Since this calculation for *CACname = normPB* is based on the probability of arrivals exceeding link capacity for 10 seconds, the observation that only 410 sessions can be justified means that with 517 concurrent sessions there are many 10-second intervals in which arrivals exceed link capacity. Hence, if an enhanced CAC policy were to attempt to make its decisions based only on traffic knowledge on time scales below 10 seconds, it would not be able to justify the admission of more than 410 sessions. In other words, the policy must incorporate traffic knowledge on time scales longer than 10 seconds if it is to approach the efficiency represented by the admission of 517 sessions.

How do we know that a time scale between 10 and 20 seconds is important? Even with long (> 10 seconds) intervals in which the arrival rate exceeds link capacity, overflows occur only after enough time for the excess rate to build a queue beyond the buffer size. To estimate an adequate time scale, we consider the magnitudes of the excess rates. For an estimate of these magnitudes, recall that 517 was the number suggested by *CACname = optsb*. Note that 517 is then also the number of sessions that is justified by *CACname = normPB* for a link 20% faster and a buffer 20% larger (except for the slight effect of a missing factor of 2 corresponding to the difference between *CACname = vopt* and *CACname = opt/2*). Recall further that *CACname = normPB* would be a conservative single-time-scale CAC policy, if only arrivals were Gaussian. Thus, the calculation can be interpreted roughly as stating that with 517 concurrent sessions, the 1-second intervals in which arrivals exceed 120% of link capacity are only barely rare enough to be tolerated. This conclusion suggests that the 1-second intervals, in which arrivals exceed an intermediate percentage (between 100% and 120%) of link capacity, would be too frequent to be tolerated on their own without some knowledge of isolation. In particular, the calculation suggests that many 1-second intervals have arrivals in excess of 110% of link capacity. Of course, if an arrival rate of 110% of link capacity persists for more than 20 seconds, the buffer would overflow. The fact, that the service is good, means that these overflows are mostly avoided, and that the excess arrivals mostly end within 20 seconds. A CAC policy able to predict this good service would have to "know" about the non-persistence of excess arrivals on a time scale no longer than 20 seconds. The earlier observation, concerning the frequency of 10-second periods of excess arrivals, indicates that the required knowledge must be on a time scale between 10 and 20 seconds.

How do we know that time scales beyond 6 minutes do not matter? In essence, we attempted to look at whether a typical episode of queue overflow was brief or durable. We found that typical durations were of a few seconds, and quite a few lasted longer than a minute. None seem to have lasted more than 6 minutes.

Unfortunately, the examination of the duration of queue overflow episodes had to be indirect. Exact durations were not available because two neighboring busy periods are usually part of a single larger episode of congestion events, and it is the length of the larger episode that is relevant. Nevertheless, one can estimate the length of the episodes based on reports of neighboring 3-minute intervals with losses. For instance, one aid to interpretation is that the number of lossy 3-minute intervals formed only 1% of the total number of lossy intervals. Hence, one may interpret almost any incident of neighboring lossy intervals as a single episode of long duration, with the expectation that this cavalier interpretation will misidentify pairs of distinct short episodes of loss as single long episodes in only a small number of cases (approximately 1% of the total number of lossy intervals). Similarly, if a typical lossy episode lasting for the fraction x of a minute is detected by reports generated every minute, then x is also the probability that the episode will be recorded in consecutive reports of lossy intervals. Based on considerations such as these, we found that a typical congestion episode lasted only 6 seconds, but quite a few lasted longer than a minute. Only very few episodes longer than 6 minutes were reported, and we concluded that these reports were due to coincidences of separate episodes happening close in time. The loss episodes are shown in Figure 5.

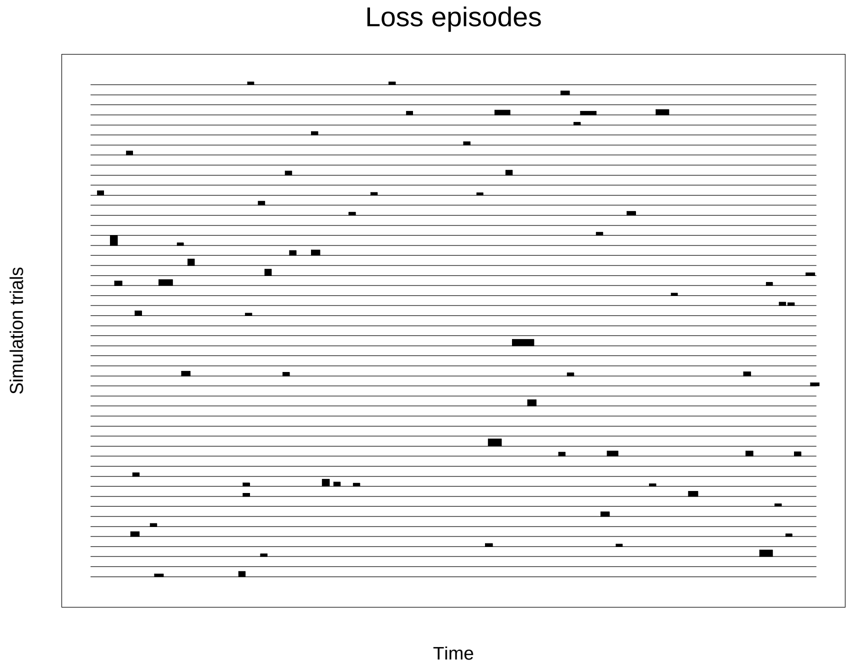

Figure 5: Loss episodes in 25 simulation trials over the period of 5.48 hours. A typical congestion episode lasted only 6 seconds, a few lasted longer than a minute, and only very few longer than 6 minutes were reported.

Another way to arrive at the 6-minute figure is to observe the traffic trace shown in Figure 2. One way of describing this traffic is in the form of a relatively long initial burst, lasting roughly from time 100 seconds until time 500 seconds, followed by a quiet period until a second short burst around time 2300 seconds, after which the traffic comes in a relatively steady drizzle. In this description, the initial burst lasts for approximately 6 minutes. The description also suggests that queue overflows mostly occur when the initial bursts of several sessions overlap. In this picture, a typical overflow cannot last longer than an overlap, and the overlap of 6-minute intervals must be a shorter interval.

Incidentally, as a separate observation, Figure 5 also suggests that the particular service standard is rather irrelevant. Thus, a user's session is either entirely lossless or not, and, in the latter case, losses will disrupt the session severely. For instance, in all the simulation runs of an OC-24 link carrying 517 data sessions, only 0.94% of the various 3-minute intervals that we examined had any losses; yet the average cell-loss rate was 2×10^{-4}, which means that on the few occasions when losses occurred, the losses were extremely large. Moreover, for an average loss episode in this case of 517 concurrent sessions, 359 sessions had losses. Thus, as long as the admission-control policy attempts to be efficient, there will be some episodes of loss and each episode will be extremely large. Regardless of the precise values that may be targeted as tolerable loss rates, the service quality provided by a policy will be determined almost entirely by the probability of whether a loss episode occurs at all.

8 Concluding remarks

We investigated the effectiveness of incorporating various traffic-information details into the design of CAC policies. We found that, compared to a peak-rate-based policy, substantial efficiency could be gained by the additional knowledge of the mean and the variance on the time scale of the half-buffer work time. While knowledge of the mean alone could provide some gain in efficiency, the additional knowledge of the variance always contributed a noticeable advantage. For some types of realistic traffic, such as the video-conference trace, the efficiency produced by this elementary knowledge of the traffic peak, mean, and variance on a single time scale, was quite satisfying. For other traffic types, such as the data trace, this simple single-time-scale information can yield some disappointing efficiencies. For such traffic traces, an efficient policy would have to employ traffic information on multiple time scales that range over more than an order of magnitude.

References

[1] P. E. Boyer and D. P. Tranchier, "A reservation principle with applications to the ATM traffic," *Computer Networks and ISDN Systems*, vol. 24, no. 4, pp. 321–334, May 1992.

[2] S. Chong, S. Q. Li, and J. Ghosh, "Predictive dynamic bandwidth allocation for efficient transport of real-time VBR video over ATM," *IEEE J. Select. Areas Commun.*, vol. 13, no. 1, pp. 12–23, January 1995.

[3] D. R. Cox, "Long-range dependence: A review," *Statistics and Appraisal, Proc. 50th Anniversary Conf.*, Iowa State Statistical Laboratory, H. A. David and H. T. David, Eds., The Iowa State University Press, 1984, pp. 55–74.

[4] J. Curose, "Open issues and challenges in providing quality of service guarantees in high-speed networks," *Computer Communication Reviews*," vol. 23, no. 1, pp. 6–15, January 1993.

[5] A. Elwalid and D. Mitra, "Effective bandwidth of general Markovian traffic sources and admission control of high-speed networks," *IEEE/ACM Transactions on Networking.* vol. 1, no. 3, pp. 329–343, June 1993.

[6] A. Elwalid, D. Heyman, T. V. Lakshman, and D. Mitra, "Fundamental bounds and approximations for ATM multiplexers with applications to video conferencing," *IEEE J. Select. Areas Commun.*, vol. 13, no. 6, pp. 1004–1016, August 1995.

[7] M. W. Garrett and W. Willinger, "Analysis, modeling and generation of self-similar VBR video traffic," *Proc. ACM SIGCOMM '94,* London, UK, August 1994, pp. 269–280.

[8] R. J. Gibbens, F. P. Kelly, and P. B. Key, "A decision-theoretic approach to call admission control in ATM networks," *IEEE J. Select. Areas Commun.*, vol. 13, no. 6, pp. 1101–1114, August 1995.

[9] M. Grossglauser, S. Keshav, and D. Tse, "RCBR: A simple and efficient service for multiple time-scale traffic," *Proc. ACM SIGCOMM '95,* Cambridge, MA, September 1995, pp. 219–230.

[10] R. Guerin, H. Ahmadi, and M. Nagshineh, "Equivalent capacity and its application to bandwidth allocation in high-speed networks," *IEEE J. Select. Areas Commun.*, vol. 9, no. 7, pp. 968–981, September 1991.

[11] D. P. Heyman, A. Tabatabai, and T. V. Lakshman, "Statistical analysis and simulation study of video teleconference traffic in ATM networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 2, no. 1, pp. 49–59, March 1992.

[12] J. Hui, "Resource allocation for broadband networks," *IEEE J. Select. Areas Commun.*, vol. 6, no. 9, pp. 1598–1608, December 1988.

[13] S. Jamin, P. Danzig, S. Shenker, and L. Zhang, "A measurement-based admission control algorithm for integrated service packet networks," *IEEE/ACM Trans. Networking*, vol 5, no. 1, pp. 56–70, February 1997.

[14] G. Kesidis, J. Walrand, and C.-C. Chang, "Effective bandwidths for multiclass Markov fluids and other ATM sources," *IEEE/ACM Trans. Networking*, vol. 1, no. 4, pp. 424–428, August 1993.

[15] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Trans. Networking*, vol. 2, no. 1, pp. 1–15, February 1994.

[16] S. H. Low and P. P. Varaiya, "A new approach to service provisioning in ATM networks," *IEEE/ACM Trans. Networking*, vol. 1 no. 5, pp. 547–553, October 1993.

[17] M. de Prycker, *Asynchronous Transfer Mode Solution to Broadband ISDN*. New York: Ellis Norwood, 1991.

[18] K. Sriram, "Methodologies for bandwidth allocation, transmission scheduling, and congestion avoidance in broadband ATM networks," *Computer Networks and ISDN Systems*, vol. 26, pp. 43–59, September 1993.

[19] D. Tolmie and J. Renwick, "HIPPI: Simplicity yields success," *IEEE Network Magazine*, vol. 7, no. 1, pp. 28–32, January 1993.

[20] D. Tse, R. G. Gallager, and J. N. Tsitsiklis, "Statistical multiplexing of multiple time-scale Markov streams," *IEEE J. Select. Areas Commun.*, vol. 13, no. 6, pp. 1028–1038, August 1995.

[21] J. S. Turner, "Managing bandwidth in ATM networks with bursty traffic," *IEEE Network Magazine*, vol. 6, no. 5, pp. 50–58, September 1992.

[22] W. Willinger, D. V. Wilson, W. E. Leland, and M. S. Taqqu, "On traffic measurements that defy traffic models (and vice versa): Self-similar traffic modeling for high-speed networks," *ConneXions*, vol. 8, no. 11, pp. 14–24, November 1994.