TCP Performance Over End-to-End Rate Control and Stochastic Available Capacity

Sanjay Shakkottai, Anurag Kumar, *Senior Member, IEEE*, Aditya Karnik, and Ajit Anvekar

***Abstract*—Motivated by TCP over end-to-end ABR, we study the performance of adaptive window congestion control, when it operates over an explicit feedback rate-control mechanism, in a situation in which the bandwidth available to the elastic traffic is stochastically time varying. It is assumed that the sender and receiver of the adaptive window protocol are colocated with the rate-control endpoints. The objective of the study is to understand if the interaction of the rate-control loop and the window-control loop is beneficial for end-to-end throughput, and how the parameters of the problem (propagation delay, bottleneck buffers, and rate of variation of the available bottleneck bandwidth) affect the performance.**

The available bottleneck bandwidth is modeled as a two-state Markov chain. We develop an analysis that explicitly models the bottleneck buffers, the delayed explicit rate feedback, and TCP's adaptive window mechanism. The analysis, however, applies only when the variations in the available bandwidth occur over periods larger than the round-trip delay. For fast variations of the bottleneck bandwidth, we provide results from a simulation on a TCP testbed that uses Linux TCP code, and a simulation/emulation of the network model inside the Linux kernel.

We find that, over end-to-end ABR, the performance of TCP improves significantly if the network bottleneck bandwidth variations are slow as compared to the round-trip propagation delay. Further, we find that TCP over ABR is relatively insensitive to bottleneck buffer size. These results are for a short-term average link capacity feedback at the ABR level (INSTCAP). We use the testbed to study EFFCAP feedback, which is motivated by the notion of the effective capacity of the bottleneck link. We find that EFFCAP feedback is adaptive to the rate of bandwidth variations at the bottleneck link, and thus yields good performance (as compared to INSTCAP) over a wide range of the rate of bottleneck bandwidth variation. Finally, we study if TCP over ABR, with EFFCAP feedback, provides throughput fairness even if the connections have different round-trip propagation delays.

***Index Terms*—Congestion control, TCP over ABR, TCP performance.**

I. Introduction

In THIS paper, we report the results of an analytical and simulation study of the interactions between an end-to-end adaptive window based protocol (such as TCP), and an explicit rate-based protocol (such as ABR) for congestion control in a packet network. It is assumed that the sender and receiver of the adaptive window control protocol are colocated with the rate-control end points, as shown in Fig. 1.

Manuscript received March 5, 1999; revised July 7, 2000; approved by IEEE/ACM Transactions on Networking Editor T. V. Lakshman. This work was supported by a grant from Nortel Networks.

S. Shakkottai was with the Department of Electrical Communication Engineering, Indian Institute of Science (IISc), Bangalore 560 012, India. He is now with the Coordinated Sciences Lab, University of Illinois, Urbana-Champaign, IL 61801 USA (e-mail: shakkott@uiuc.edu).

A. Kumar and A. Karnik are with the Department of Electrical Communication Engineering, Indian Institute of Science (IISc), Bangalore 560 012, India (e-mail: anurag@ece.iisc.ernet.in; karnik@ece.iisc.ernet.in).

A. Anvekar was with the Department of Electrical Communication Engineering, Indian Institute of Science (IISc), Bangalore 560 012, India. He is now with IISc Simputer Consortium, Bangalore 560 012, India.

Publisher Item Identifier S 1063-6692(01)06846-7.

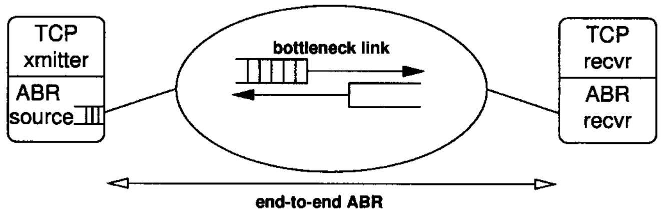


Fig. 1. The TCP endpoints are colocated with the ABR endpoints. We call this scenario TCP over end-to-end ABR.

TCP is, by far, the dominant end-to-end transport protocol for elastic traffic in the Internet today. TCP uses an adaptive window mechanism for flow control, congestion control, and bandwidth sharing. The normal behavior of all TCP senders is to gradually increase their transmit windows upon receiving acknowledgments (ACKs), thereby increasing their sending rates. This continues until some link gets congested as a consequence of which there is packet loss. Implicit loss indications then cause senders to reduce their windows. Thus, the TCP transmit window, and hence the TCP transmission rate, has an oscillatory behavior that can lead to low link utilization. Further, owing to the ACK-based self-clocking mechanism, fairness between sessions is also an issue.

The available bit rate (ABR) service in asynchronous transfer mode (ATM) networks is primarily meant for transporting best-effort data traffic. Connections that use the ABR service (so-called ABR sessions) share the network bandwidth left over after serving constant bit rate (CBR), e.g., circuit emulation, and variable bit rate (VBR), e.g., variable rate-compressed video traffic. This available bandwidth varies with the requirements of the ongoing CBR/VBR sessions. The switches carrying ABR sessions continually calculate a fair rate for each session at each output port, and use resource management (RM) cells to explicitly feed this rate back to the session sources (see [3]). This explicit rate feedback causes the ABR sources to reduce or increase their cell transmission rates depending on the availability of bandwidth in the network.

Even if the wide-area packet transport technology is ATM based, since the ABR service does not guarantee end-to-end reliable transport of data, the applications in the end-systems use TCP as the end-to-end transport protocol. Moreover, with the evolution of gigabit ethernet, ATM has become primarily a wide-area networking technology. Hence, ATM endpoints

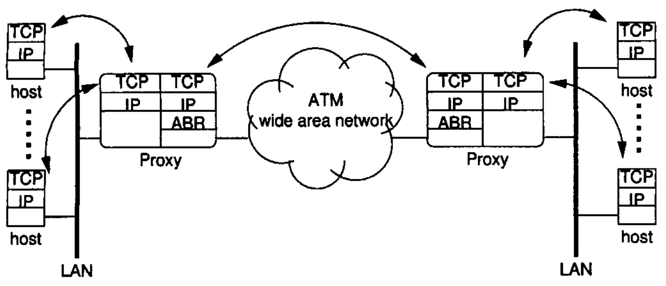


Fig. 2. TCP/IP hosts (attached to LANs) communicating over a wide area network via proxies. There is a single, long-lived, "proxy-to-proxy" TCP connection over ATM/ABR; the proxies are ATM/ABR endpoints. Each TCP session between a pair of end systems is carried over two "local" TCP/IP connections over the LANS (between the end-systems and the their respective proxies), and over the single TCP/IP/ABR connection over the ATM WAN.

would typically be in edge devices (such as edge routers or proxies) rather than in clients or servers.

A situation that our work applies to is depicted in Fig. 2. In Fig. 2, a proxy at a customer's site has an ATM network interface card that attaches it to the ATM WAN, and an ethernet card on the LAN side. The situation depicted could represent an enterprise or a web services provider that is managing (e.g., backing up, synchronizing) the data on its web servers across two sites, or an Internet brokerage that has its brokers at one site and servers at another. One persistent TCP connection can be set up over the ATM WAN between the proxies at the two sites, and this connection can be shared by all the transactions between the sites. Over the local networks, there are short-lived TCP connections between the web servers or clients and their respective proxies. In this framework, our results in this paper would apply to the "proxy-to-proxy" (edge-to-edge) TCP over ABR connection. Note that if this is the dominant mechanism for transporting elastic traffic over the ATM network, then the ATM WAN carries mostly *long-lived* ABR connections, making the end-to-end feedback based ABR approach viable. Further, the long-lived TCP connection (between the proxies) can maintain window state from transfer to transfer thus avoiding slow start for each short transfer. In addition, each proxy can effectively pace the local connections by using ack pacing, or explicit rate feedback into the TCP senders in the hosts on the LAN. The latter approach has been investigated further in [13]. Most importantly, from the point of view of this paper, this network architecture justifies studying a single long-lived TCP connection (or a small number of such TCP connections) over a long-lived wide area ATM/ABR virtual circuit(s).

One of the concerns in an integrated network is that best-effort elastic traffic shares the network bandwidth with CBR/VBR sessions. Thus, the bandwidth available to elastic traffic is time varying and stochastic. Effective rate-control mechanisms for ABR can be designed even with stochastic variations in bottleneck bandwidth (see [2]). TCP has an adaptive window-control mechanism where the window size oscillates periodically, even when the network capacity does not change. *The question that we wish to answer is that if TCP operates over a rate-control mechanism such as ABR, whether the interaction is beneficial or not, and how the interaction can be improved.*

Many simulation studies have been carried out to study the interaction between the TCP and ATM/ABR control loops. A study of the buffering requirements for zero cell loss for TCP over ABR is reported in [9]. Using simulations, it is shown that the buffer capacity required at the switch is proportional to the maximum round-trip time (RTT) of all the virtual circuits (VCs) through the link, and is independent of the number of sources (or VCs). The proportionality factor depends on the switch algorithm. In further work, the authors in [10] introduce various patterns of VBR background traffic. The VBR background traffic introduces variations in the ABR capacity and the TCP traffic introduces variations in the ABR demand.

In [6], the authors study the effect of ATM/ABR control on the throughput and fairness of running large unidirectional file transfer applications on TCP-Tahoe and TCP-Reno with a single bottleneck link with a static service rate. The authors in [16] study the performance of TCP over ATM with multiple connections, but with a static bottleneck link. The paper reports a simulation study of the relative performance of the ATM ABR and unspecified bit rate (UBR) service categories in transporting TCP/IP flows through an edge-to-edge ATM (i.e., the host nodes are not ATM endpoints) network. Their summary conclusion is that there does not seem to be strong evidence that for TCP/IP workloads the greater complexity of ABR pays off in better TCP throughputs. Their results are, however, for edge-to-edge ABR; they do not comment on TCP over end-to-end ABR which is what we study in this paper.

All the studies above are primarily simulation studies. There are also a few related analytical studies. In [11], the authors study the interaction of TCP and ABR control loops with a focus on the interaction between the rate increase behavior of the ABR source and the ramp-up time of the congestion window during TCP slow start. They conclude that the ramp-up time of the TCP window can be significantly prolonged over ABR when the RTT is small. However, in our study, as noted earlier, we are primarily interested in WANs with large RTTs, and we focus on the long-term throughput of TCP with and without rate control. In [4], the authors study TCP over a fading wireless link, which is modeled as a Markov chain. The analysis consists of modeling the arrival process into the buffer of the link as a Bernoulli process, thus neglecting TCP window dynamics. This, as they note, is different from the arrival stream generated by TCP.

In this paper, we make the following contributions.

1) We develop an analytical model for a TCP connection over explicit rate ABR when there is a single bottleneck link with time varying available capacity. In the analytical model, we assume that the explicit-rate feedback is based on the *short-term average available capacity*; we think of this as *instantaneous capacity* feedback, and we call the approach *INSTCAP* feedback. We explicitly model TCP's adaptive window dynamics, the bottleneck buffer process, stochastic variations of the bottleneck rate, and ABR rate feedback with delay. Since we model the buffer process at the bottleneck link, unlike the approach in [17], our analysis does not need the loss probability as an externally provided parameter.
2) We use a testbed to validate the analytical results. This testbed implements a hybrid simulation comprising Linux TCP code, and a network emulation/simulation implemented in the loopback device driver code in the Linux

kernel. While the analysis has been done only for slow bottleneck rate variations, as compared to the RTT, the simulations study a wide range of bottleneck rate variations. In spite of the fact that many of our conclusions are based on simulations, there is important value in the analysis that we have provided. Simulations are often used to verify analyses, but the reverse can also be useful. A detailed simulation of a protocol as complex as TCP, or a modification of TCP code, can often lead to erroneous implementations. If an approximate analysis is available for even some situations, it can help to validate the simulation code. In fact, when doing another related piece of work, reported in [13], a serious error in a simulation was discovered only because the simulation failed to match an analysis.

3) Then, with the loss sensitivity of TCP in mind, we develop an explicit rate feedback that is based on a notion of *effective service capacity* of the bottleneck link (derived from large deviations analysis of the bottleneck queue process). We call this *EFFCAP* feedback. EFFCAP is more effective in preventing loss at the bottleneck buffers. Since the resulting model is hard to analyze, the results for EFFCAP feedback are all obtained from the hybrid simulator mentioned above. Our results show that different types of bottleneck bandwidth feedbacks are needed for slowly varying bottleneck rate, rapidly varying bottleneck rate and the intermediate regime. *EFFCAP feedback adapts itself to the rate of bottleneck rate variation.* We then develop guidelines for choosing two parameters that arise in the on-line calculations of EFFCAP. Notions of effective service capacity of time-varying links, in the context of congestion control, have also been introduced and used in [4] and [2].
4) Finally, we study the performance of two TCP connections that pass through the same bottleneck link, but have different round-trip propagation delays. Our objective here is to determine whether TCP over ABR is fairer than TCP alone, and under what circumstances. In this study, we only use EFFCAP feedback.

The paper is organized as follows. In Section II, we describe the network model under study. In Section III, we develop the analysis of TCP over ABR with INSTCAP feedback, and of TCP alone. In Section IV, we develop the EFFCAP algorithm; TCP over ABR with EFFCAP feedback is only amenable to simulation. In Section V, we present analysis results for INSTCAP feedback, and simulation results for INSTCAP and EFFCAP. The performance of INSTCAP and EFFCAP feedbacks are compared. In Section VI, we study the choice of two parameters that arise in EFFCAP feedback. In Section VII, we provide simulation results for two TCP connections over ABR with EFFCAP feedback. Finally, in Section VIII, we summarize the observations from our work.

II. The Network Model

Consider a system consisting of a TCP connection between a source and destination node connected by a network with a large propagation delay, as shown in Fig. 1. We assume that only

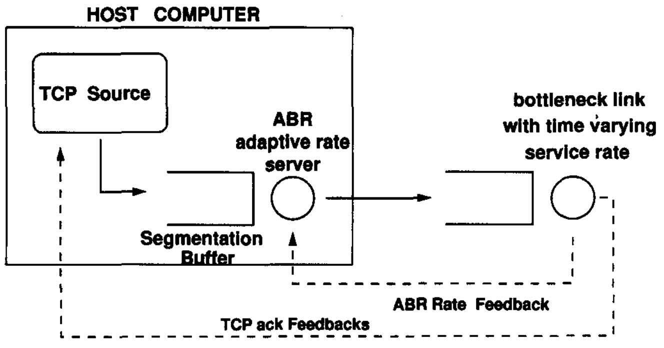


Fig. 3. The segmentation buffer of the system under study is in the host NIC card and extends into the host's main memory. The rate feedback from the bottleneck link is delayed by one round-trip delay.

one link (called the *bottleneck link*) causes significant queuing delays in this connection, the delays owing to the other links being fixed (i.e., only fixed propagation delays are introduced by the other links). A more detailed model of this is shown in Fig. 3. The TCP packets are converted into ATM cells and are forwarded to the ABR segmentation buffer. This buffer is in the network interface card (NIC) and extends into the main memory of the computer. Hence, we can look upon this as an infinite buffer. The segmentation buffer server (also called the ABR source) gets rate feedback from the network. The ABR source service rate adapts to this rate feedback. When we study TCP alone, this segmentation buffer is absent from the model.

The bottleneck link buffer represents either an ABR output buffer in an ATM switch (in case of TCP over ABR), or a router buffer (in case of TCP alone). The network carries other traffic (CBR/VBR), which causes the bottleneck link capacity (as seen by the connection of interest) to vary with time. The bottleneck link buffer is finite, which can result in packet loss due to buffer overflow when rate mismatch between the source rate and the link service rate occurs. In our model, we will assume that a portion of the link capacity is reserved for best-effort traffic, and hence is always available to the TCP connection. In the ATM/ABR case, such a reservation would be made by using the minimum cell rate (MCR) feature of ABR, and would be implemented by an appropriate link scheduling mechanism. Thus, when guaranteed service traffic is backlogged at this link, then the TCP connection gets only the bandwidth reserved for best-effort traffic, otherwise it gets the full bandwidth. *Hence, a two-state model suffices for the available link rate.*

III. TCP/ABR with INSTCAP Feedback

Fig. 4 shows a queuing model of the network scenario described in Section II. At time t, the cells in the ATM segmentation buffer at the source are transmitted at a time dependent rate S_t^{-1} which depends on the ABR rate feedback (i.e., S_t is the service time of a packet at time t). The bottleneck has a finite buffer $B_{\max}$ and has time dependent service rate R_t^{-1} *packets/s.*

A. Modeling Assumptions

In order to simplify an otherwise intractable analysis, and to focus on the basic issue of an adaptive window congestion con-

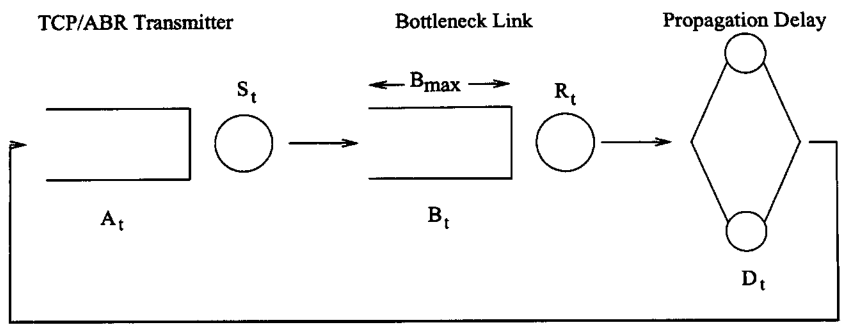


Fig. 4. Queuing model of TCP over end-to-end.

trol operating over an adaptive-rate congestion control, we make the following modeling assumptions.

1) We model a longed-lived TCP connection during the data transfer phase, hence the data packets are assumed to be of fixed length (the TCP segment size).
2) The ABR segmentation buffer can extend into the main memory of the client; hence, the segmentation buffer capacity is assumed to be infinite. There are as many packets in this buffer as the number of untransmitted packets in the TCP window. The (time-dependent) service time S_t at this buffer models the time taken to transmit an entire TCP packet worth of ATM cells. We assume that the service rate at the segmentation buffer does not change during the transmission of the cells from a single TCP packet.
3) The bottleneck link is modeled as a finite buffer queue with service rate that is Markov modulated by an independent Markov chain on two states 0 and 1; the service rate is higher in state 0. Each packet that enters the buffer has a service rate R_t^{-1} at time t, which is assumed constant over the service time of the packet.
4) If the bottleneck link buffer is full when a cell arrives at it, the cell is dropped. In addition, we assume that all cells corresponding to that TCP packet are dropped. This assumption allows us to work with full TCP packets only.[1]
5) The round-trip propagation delay Δ is modeled by an infinite server queue with service time Δ. Notice that various propagation delays in the network (the source-bottleneck link delay, bottleneck link-destination delay, and the destination-source return path delay) have been lumped into a single delay element (see Fig. 4). This can be justified from the fact that even if the source adapts itself to the change in link capacity earlier than one RTT, the effect of that change will be seen only after a RTT at the bottleneck link.
6) On receiving an ACK, the TCP sender may increase the transmit window. The TCP window evolution can be modeled in several ways (see [15], [14], [17]). In this study, we model the TCP window adjustments in the congestion avoidance phase probabilistically as follows. Every time a nonduplicate ACK arrives at the source, the window size W_t increases by one with probability (w.p.) $1/W_t$

$$W_{t+} = \begin{cases} W_t + 1, & \text{w.p. } \frac{1}{W_t} \\ W_t, & \text{otherwise.} \end{cases} \tag{1}$$

7) If a packet is lost at the bottleneck link buffer, the ACK packets for any subsequently received packets continue to carry the sequence number of the lost packet. Eventually, the source window becomes empty, timeout begins and at the expiry of the timeout, the threshold window W_t^{th} is set to half the maximum congestion window achieved after the loss, and the next slow start begins.

This model approximates the behavior of TCP without fast retransmit. We consider this simple version of TCP, as we are primarily interested in studying the interaction between rate and window control. This version is simpler to model and captures the interaction that we wish to study.

With "packets" being read as "full TCP packets," we define the following notation.

A_t	number of packets in the segmentation buffer at the host at time t;
B_t	number of packets in the bottleneck link buffer at time t;
D_t	number of packets in the propagation queue at time t;
R_t	service time of a packet at the bottleneck link; $R_t \in \{r_0, r_1\}$. We take $r_0 = 1$ and $r_1 > r_0$. Thus, **all times are normalized to the bottleneck link packet service time at the higher service rate;**
S_t	service time of a packet at the ABR source.

8) We assume that S_t follows R_t with delay Δ, i.e., $S_t = R_{t-\Delta}$, and $S_t \in \{r_0, r_1\}$. For simplicity, we do not model the detailed ABR source behavior which additively increases the transmission rate in small increments (see [1]). We are not driving the rate feedback from variations in the bottleneck queue length, but are directly feeding back the current available rate at the bottleneck link.

Since the instantaneous rate of the bottleneck link is fed back, we call this the *instantaneous-rate feedback* scheme. (Note that, in practice, the instantaneous rate is really the average rate over a small window; that is how

[1]This is an idealization of cell discard schemes, such as partial packet discard [18] or early packet discard (EPD), designed to prevent the ATM network from wastefully carrying cells that belong to TCP packets some of whose constituent cells have been lost.

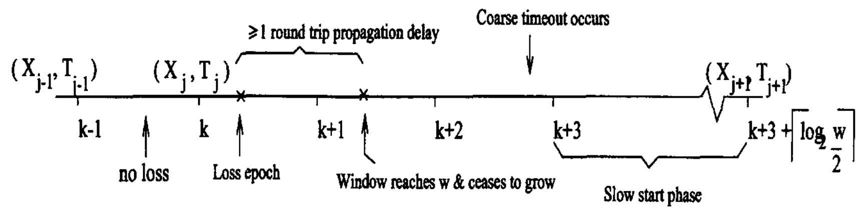


Fig. 5. The embedded process $\{(X_j, T_j), j \geq 0\}$.

instantaneous-rate feedback is modeled in our simulations to be discussed later; we will call this feedback *INSTCAP*).[2]

B. Analysis of the Queueing Model

Consider the vector process

$$\{Z_t, t \geq 0\} := \{(A_t, B_t, D_t, R_t, S_t), t \geq 0\}. \quad (2)$$

This process is hard to analyze directly. Instead, we study an embedded process, which with suitable approximations, turns out to be analytically tractable. Define $t_k := k\Delta$, $k \geq 0$. Now, consider the embedded process

$$\left\{\tilde{Z}_k, k \geq 0\right\} = \left\{Z_{t_k}, k \geq 0\right\} \quad (3)$$

with $\tilde{Z}_0 = (1, 0, 0, r_0, r_0)$. We will use the obvious notation $\tilde{Z}_k = (A_k, B_k, D_k, R_k, S_k)$.

For mathematical tractability, we will make the following additional assumptions.

1) We assume that the rate modulating Markov chain is embedded at the epochs $(t_0, t_1, \ldots)$, i.e., the bottleneck link rate changes only at multiples of Δ. Thus, this analysis will not apply to cases where the link rate changes more frequently than once per Δ. For these cases, we will use simulations.
2) We assume that packet transmissions do not straddle the embedded epochs.
3) We assume that there is no loss in the slow-start phase of TCP. In [15], the authors show that loss will occur in the slow-start phase if $B_{\max}/[(\Delta/r_0) + 1] < 1/3$, even if no rate change occurs in the slow-start phase. For the case of TCP over ABR, as the source and bottleneck link rates match, no loss will occur in this phase as long as rate changes do not occur during slow-start. This assumption is valid for the case of TCP alone only if $B_{\max}/[(\Delta/r_0) + 1] > 1/3$; hence, with this assumption, we find that our analysis overestimates the throughput when TCP is used alone (without ABR).
4) *Timeout and Loss-Recovery Model:* Observe that packets in the propagation delay queue (see Fig. 4) at t_k will have departed from the queue by t_{k+1}. This follows as the service time is deterministic, equal to Δ, and $t_{k+1} - t_k = \Delta$. Further, any new packet arriving to the propagation delay queue during (t_k, t_{k+1}) will still be present in that queue at t_{k+1}. On the other hand, if loss occurs due to buffer overflow at the bottleneck link in (t_k, t_{k+1}), we proceed as follows. Fig. 5 shows a packet-loss epoch in the interval (t_k, t_{k+1}). This is the first loss since the last time that TCP went through a timeout and recovery. At this loss epoch, there are packets in the bottleneck buffer, and some ACKs "in flight" back to the transmitter. These ACKs and packets form an unbroken sequence, and hence will all contribute to the window increase algorithm at the transmitter (we assume that there is no ACK loss in the reverse path). The transmitter will continue transmitting until the window is exhausted and will then start a coarse timer. We assume that this timeout will occur in the interval (t_{k+2}, t_{k+3}) (see Fig. 5), and that recovery starts at the embedded epoch t_{k+3}. Thus, when the first loss (after recovery) occurs in an interval then, in our model, it takes two more intervals to start recovery.[3]

At time t_k, let $\tilde{Z}_k = (a, b, d, r, s)$. If no loss has occurred (since last recovery) until t_k the TCP window at t_k is $a + b + d$. Now, given $\tilde{Z}_k$, we can find the probability that a loss occurs during (t_k, t_{k+1}), and the distribution of the TCP window at the time that timeout starts. (This calculation depends on the fact that d ACKs will arrive at the TCP transmitter during t_k, t_{k+1}, and also on the probabilistic window evolution model during TCP congestion avoidance; the calculation is explained below.) Suppose this window is w, then the congestion avoidance threshold in the next recovery cycle will be $m := \lceil w/2 \rceil$. It will take approximately $\lceil \log_2 m \rceil$ RTTs (each of length Δ) to reach the congestion avoidance threshold. Under the assumption that no loss occurs during the slow-start phase, congestion avoidance starts at $k' = k+3+\lceil \log_2 m \rceil$, and we can determine the distribution of $\tilde{Z}_{k'}$.

With the above description in mind, define

$$T_0 = t_0 = 0 \quad \text{and} \quad X_0 = \tilde{Z}_0 = (1, 0, 0, r_0, r_0). \quad (4)$$

[2]Notice that with ABR alone (i.e., ABR is not below TCP), if the average bottleneck link rate is fed back to the source, and the source sends at this rate, then we have an "unstable" open queuing model. With TCP over ABR, however, the model in Fig. 4 is a closed queuing network, in which the number of "customers" is bounded by the maximum TCP window. Hence, even if the ABR source rate is equal to the average service rate at the bottleneck, the system will be stable. Also, with INSTCAP rate feedback, the rate feedback will either be r_0^{-1} or $r_1^{-1} (< r_0^{-1})$. If the source sends at r_0^{-1}, then eventually there will be a loss, and since TCP is over ABR the system will be "reset." See [2] for an approach for explicit rate-based congestion control (without TCP) based on the effective service capacity concept, where the source directly adapts to an available rate estimate. The rate estimate is chosen, however, to put a certain constraint on the queue behavior if the source was to send at that rate.

[3]TCP samples some RTTs of transmitted packets, and uses an exponentially weighted moving average for estimating an average RTT and a measure of variation of RTT. The retransmission time out (RTO) is then obtained from these two statistics. It is not possible to capture this mechanism in a simple Markov model. Hence, in our simulations, we modified TCP code so that the RTO was fixed at two times the RTT.

For $k \geq 1$

$$T_k = \begin{cases} T_{k-1} + \Delta, & \text{if no loss occurs in } (T_{k-1}, T_{k-1} + \Delta) \\ T_{k-1} + \left(3 + \left\lceil \log_2 \frac{w}{2} \right\rceil\right) \Delta, & \text{if loss occurs in } (T_{k-1}, T_{k-1} + \Delta) \text{ and the loss window is } w \end{cases} \quad (5)$$

and $X_k = \tilde{Z}_{T_k}$. For a particular realization of X_k, we will write $X_k = x$ where $x = (a, b, d, r, s)$. Define

$$p(x) = \Pr\{\text{loss occurs during } (T_k, T_k + \Delta) | X_k = x\} \quad (6)$$

and

$$U_k = \frac{(T_{k+1} - T_k)}{\Delta} \qquad \text{for } k \geq 0. \quad (7)$$

Given $X_k = x$, we have

$$U_k = 1 \text{ w.p. } 1 - p(x). \quad (8)$$

We now proceed to analyze the evolution of $\{X_k, k \geq 0\}$.

The bottleneck link-modulating process, as mentioned earlier, is a two-state Markov chain embedded at $\{t_k, k \geq 0\}$ taking values in $\{r_0, r_1\}$. Let p_{01}, p_{10} be the transition probabilities of the Markov chain. Notice that $S_k = R_{k-1}$, hence (R_k, S_k) is also a discrete-time Markov chain (DTMC). Let Q be the transition probability matrix for (R_k, S_k). Then, $Q^n(i_1 j_1; i_2 j_2) = \Pr\{R_{k+n} = i_2, S_{k+n} = j_2 | R_k = i_1, S_k = j_1\}$, where $i_1, j_1, i_2, j_2 \in \{r_0, r_1\}$.

As explained above, given $X_k = (A_k, B_k, D_k, R_k, S_k)$, the TCP congestion window is

$$W_k = A_k + B_k + D_k. \quad (9)$$

For particular $X_k = (a, b, c, r, s)$, X_{k+1} can be determined using the probabilistic model of window evolution during the congestion avoidance phase. Consider the evolution of A_k, the segmentation buffer queue process. If no loss occurs in (T_k, T_{k+1})

$$A_{k+1} = \left(a + d + N_k - \frac{\Delta}{s}\right)^+ \quad (10)$$

where N_k is the increment in the TCP window in the interval, and is characterized as follows. During (T_k, T_{k+1}), for each ACK arriving at the source (say, at time t), the window size increases by one with probability $1/W_t$. However, we further assume that the window size increases by one with probability $1/W_k$ (where $W_k = a + b + d$), i.e., the probability does not change after every arrival but, instead, we use the window at T_k. Then, with this assumption, due to d arrivals to the source queue, the window size increases by the random amount N_k. We see that for d ACKs, the maximum increase in window size is d. Let us define $\tilde{N}_k$ such that $\tilde{N}_k \sim \mathit{Binomial}(d, [1/(a+b+d)])$. Then , $N_k = \min(\tilde{N}_k, W_{\max} - (a + b + d))$. We can similarly get recursive relations for B_{k+1} and D_{k+1} [19].

Let us now define

$$\alpha(x; w) = \Pr\{\text{window achieved is } w | X_k = x, \text{ loss in } (T_k, T_k + \Delta)\}. \quad (11)$$

When no loss occurs, U_k is given by (8). When loss occurs, given $X_k = x = (a, b, c, i_1, j_1)$, the next cycle begins after the recovery from loss which includes the next slow-start phase. Suppose that the window was $2m$ when loss occurred. Then, the next congestion avoidance phase will begin when the TCP window size in the slow-start phase after loss recovery reaches m. This will take $\lceil \log_2 m \rceil$ cycles. At the end of this period, the state of various queues is given by $(A_k, B_k, D_k) = (0, 0, m)$. The channel state at the start of the next cycle can be described by the transition probability matrix of the modulating Markov chain. Hence

$$U_k = 3 + \lceil \log_2 m \rceil \text{ w.p. } p(x).\alpha(x; 2m) \quad (12)$$

and

$$X_{k+1} = (0, 0, m, i_2, j_2) \text{ w.p. } p(x).\alpha(x; 2m).Q^{(3+\log_2 m)}(i_1, j_1; i_2, j_2). \quad (13)$$

From the above discussion, it is clear that given X_k, the distribution of X_{k+1} can be computed without any knowledge of its past history. Hence, $\{X_k, k \geq 0\}$ is a Markov chain. Further, given X_k, the distribution of T_{k+1} can be computed without any knowledge of past history. Hence, the process $\{(X_k, T_k), k \geq 0\}$ is a Markov Renewal Process (MRP) (See [21]). It is this MRP that is our model for TCP/ABR.

C. Computation of Throughput

Given the Markov Renewal Process $\{(X_k, T_k), k \geq 0\}$, we associate with the kth cycle (T_k, T_{k+1}) a "reward" V_k that accounts for the successful transmission of packets. Let $\pi(x)$ denote the stationary probability distribution of the Markov chain $\{X_k, k \geq 0\}$. Denote by $\gamma_{\text{TCP/ABR}}$, the throughput of TCP over ABR. Then, by the Markov renewal-reward theorem [21], we have

$$\gamma_{\text{TCP/ABR}} = \frac{E_\pi V}{E_\pi U} \quad (14)$$

where $E_\pi(\cdot)$ denotes the expectation w.r.t. the stationary distribution $\pi(x)$.

The distribution $\pi(x)$ is obtained from the transition probabilities in Section III-B. We have

$$E_\pi V = \sum_x \pi(x) V(x) \quad (15)$$

where $V(x)$ is the expected reward in a cycle that begins with $X = x$. Denote by $A(x)$, $B(x)$ and $D(x)$ the values of A, B and D in the state x. Then, in an interval $(T_k, T_k + \Delta)$ where no loss occurs, we take

$$V(x) = D(x) \text{ w.p. } 1 - p(x). \quad (16)$$

Thus, for lossless intervals, the reward is the number of ACKs returned to the source; note that this actually accounts for packets successfully received by the receiver in previous intervals.

Loss occurs only if the ABR source is sending at the high rate and the link is transmitting at the low rate. When loss occurs in $(T_k,\, T_k + \Delta)$, we need to account for the reward in the interval starting from T_k until T_{k+1} when slow-start ends. Note that at T_k, the congestion window is $A(x) + B(x) + D(x)$. The first component of the reward is $D(x)$; all the $B(x)$ buffered packets will result in ACKs, causing the left edge of the TCP window to advance. Since the link rate is half the source rate, loss will occur when $2(B_{\max} - B(x))$ packets enter the link buffer from the ABR source; these packets succeed and cause the left edge of the window to further advance. Further, we assume that the window grows by 1 in this process. Hence, following the lost packet, at most $A(x)+B(x)+D(x)+1$ packets can be sent. Thus, we bound the reward before timeout occurs by $D(x)+B(x)-2(B_{\max}-B(x))+A(x)+B(x)+D(x)+1 = A(x) + 2D(x) + 2B_{\max} + 1$. After loss and timeout, the ensuing slow-start phase successfully transfers some packets (as described earlier). Hence, an upper bound on the "reward" when loss occurs is $A(x) + 2D(x) + 2B_{\max} + 1 + S_{slowstart}(x)$, where

$$S_{slowstart}(x) = \sum_w \alpha(x;\, w)\left(2^{\log_2(w/2)} - 1\right) \tag{17}$$

the summation index w being over all window sizes. Actually, this is an optimistic reward, as some of the packets will be transmitted again in the next cycle, even though they have successfully reached the receiver. We could also have a conservative accounting, where we assume that if loss occurs, all the packets transmitted in that cycle are retransmitted in future cycles. In the numerical results, we shall compare the throughputs with these two bounds. It follows that

$$\begin{aligned} E_\pi V = \sum_x \pi(x)\Bigg((1-p(x))D(x) + p(x) \\ \cdot \Bigg(A(x) + 2D(x) + 2B_{\max} + 1 \\ + \sum_w \alpha(x;\, w)\left(2^{\log_2(w/2)} - 1\right)\Bigg)\Bigg). \end{aligned} \tag{18}$$

Similarly, we have

$$E_\pi U = \sum_x \pi(x) U(x) \tag{19}$$

where $U(x)$ is the mean cycle length when $X = x$ at the beginning of the cycle.

From the analysis in Section III-B, it follows that

$$U(x) = \begin{cases} 1, & \text{w.p. } 1 - p(x) \\ \sum_w \left(3 + \left\lceil \log_2 \frac{w}{2} \right\rceil\right) \alpha(x;\, w), & \text{otherwise.} \end{cases} \tag{20}$$

Hence

$$\begin{aligned} E_\pi U = \sum_x \pi(x)\Bigg((1 - p(x)) + p(x) \sum_w \alpha(x;\, w) \\ \cdot \left(3 + \left\lceil \log_2 \frac{w}{2} \right\rceil\right)\Bigg). \end{aligned} \tag{21}$$

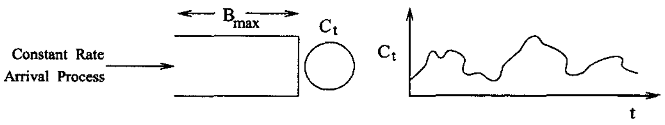


Fig. 6. Single-server queue with time-varying service capacity, being fed by a constant rate source.

D. TCP without ATM/ABR

Without the ABR rate control, the source host would transmit at the full rate of its link; we assume that this link is much faster than the bottleneck link and model it as infinitely fast. The system model is then very similar to the previous case, the only difference being that we have eliminated the segmentation buffer. The assumptions we make in this analysis, however, lead to an optimistic estimate of the throughput. The analysis is analogous to that provided above.

IV. TCP/ABR WITH EFFCAP FEEDBACK

We now develop another kind of rate feedback. To motivate this approach, consider a finite buffer single server queue with a stationary ergodic service process (see Fig. 6). Suppose that the ABR source sent packets at a constant rate. Then, we would like to find that rate which maximizes TCP throughput. Hence, let the input process to this queue be a *constant-rate deterministic arrival process*. Given the buffer size $B_{\max}$ and a desired quality of service (QoS) (say a cell-loss probability $\leq \epsilon$), we would like to know the maximum rate of the arrival process such that the QoS guarantee is met.

We look at a discrete-time approach to this problem (see [20]); in practice, the discrete-time approach is adequate, as the rate feedback is only updated at multiples of some basic measurement interval. Consider a slotted-time-queuing model where we can service C_i packets in slot i and the buffer can hold $B_{\max}$ packets. $\{C_i\}$ is a stationary and ergodic process; let EC be the mean of the process and $C_{\min}$ be the minimum number of packets that can be served per slot. A constant number of packets (denoted by γ) arrive in each slot. We would like to find $\gamma_{\max}$ such that the desired QoS (cell loss probability $\leq \epsilon$) is achieved. In [20], the following asymptotic condition is considered. If X is a random variable that represents the stationary queue length, then, with $\delta > 0$,[4]

$$\lim_{B_{\max}\to\infty} \frac{1}{B_{\max}} \log P(X > B_{\max}) < -\delta \tag{22}$$

i.e., for large $B_{\max}$, the loss probability is better then $e^{-\delta B_{\max}}$. It is shown that this performance objective is met if

$$\gamma < \frac{-1}{\delta} \lim_{n\to\infty} \frac{1}{n+1} \log E \exp\left(-\delta \sum_{i=0}^{n} C_i\right). \tag{23}$$

For the desired QoS, we need $\delta = -\log \epsilon / B_{\max}$. Let us denote the expression on the right-hand side of (23) as Γ_{eff}. Then, Γ_{eff} can be called the *effective capacity* of the server. If $\epsilon \to 1$, then $\Gamma_{\text{eff}} \to$ EC and as $\epsilon \to 0$, $\Gamma_{\text{eff}} \to C_{\min}$ which is what we intuitively expect. For all other values of ϵ, $\Gamma_{\text{eff}} \in (C_{\min},\, \text{EC})$.

[4]All logarithms are taken to the base e.

Let us apply this effective capacity approach to our problem. Let the ABR source (see Fig. 3) adapt to the effective bandwidth of the bottleneck link server. In our analysis, we have assumed a Markov-modulated bottleneck link capacity, with changes occurring at most once every Δ units of time, Δ being the round-trip propagation delay. Hence, we have a discrete-time model, with γ being the number of packet arrivals to the bottleneck link in Δ units of time and C_i being the number of packets served in that interval. We will compute the effective capacity of the bottleneck link server using (23). However, before we can do this, we still need to determine the desired QOS, i.e., ϵ, or equivalently, δ.

To find δ, we conduct the following experiment. We let the ABR source transmit at some constant rate, say μ; $\mu \in (\mathrm{EC}, C_{\min})$. For a given Markov modulating process, we find the μ that maximizes TCP throughput. We will assume that this is the effective capacity of the bottleneck link. Now, using (23), we can find the smallest δ that results in an effective capacity of this μ. If the value of δ so obtained turns out to be consistent for a wide range of Markov modulating processes, then we will use this value of δ as the QoS requirement for TCP over ABR.

The above discrete-time queuing model for TCP over ABR can be analyzed in a manner analogous to that in Section III-B. We find from the analysis that for several sets of parameters, the value of δ which maximizes TCP throughput is consistently very large (about 60–70) [19]. This is as expected, since TCP performance is very sensitive to loss.

A. Algorithm for Effective Capacity Computation

In practice, we do not know *a priori* the statistics of the modulating process. Hence, we need an on-line method of computing the effective bandwidth. In this section, we develop an algorithm for computing the effective capacity of a time varying bottleneck link carrying TCP traffic. The idea is based on (23), and the observation at the end of the previous section that δ is very large.

We take the measurement interval to be s time units; s is also the update interval of the rate feedback. We shall approximate the expression for effective bandwidth in (23) by replacing $n \to \infty$ by a large finite M

$$\Gamma_{\text{eff}} \approx \frac{-1}{M\delta} \log E \exp\left(-\delta \sum_{i=1}^{M} C_i\right). \tag{24}$$

What we now have is an effective capacity computation performed over Ms units of time. We will assume that the process is ergodic and stationary. Hence, we approximate the expectation by the average of N sets of samples, each set taken over Ms units of time. Note that since the process is stationary and ergodic, the N intervals need not be disjoint for the following argument to work. Then, denoting C_{ij} as the ith link capacity value ($i \in \{1, M\}$) in the jth block of M intervals ($j \in \{1, N\}$), we have

$$\Gamma_{\text{eff}} \approx \frac{-1}{M\delta} \log \frac{1}{N} \sum_{j=1}^{N} \exp\left(-\delta \sum_{i=1}^{M} C_{ij}\right) \tag{25}$$

$$= \frac{-1}{M\delta} \log \frac{1}{N} - \frac{1}{M\delta} \log \sum_{j=1}^{N} \exp\left(-\delta \sum_{i=1}^{M} C_{ij}\right). \tag{26}$$

As motivated above, we now take δ to be large. This yields

$$\overset{\delta\to\infty}{\approx} \frac{-1}{M\delta} \log \exp\left[-\delta\left(\min_{j\in N} \sum_{i=1}^{M} C_{ij}\right)\right] \tag{27}$$

$$= \min_{j\in N} \frac{1}{M} \sum_{i=1}^{M} C_{ij}. \tag{28}$$

We notice that this essentially means that *we average capacities over N sliding blocks, each block representing M s units of time, and feed back the minimum of these values* (see Fig. 7).

The formula that has been obtained [(28)] has a particularly simple form. The above derivation should be viewed more as a motivation for this formula. The formula, however, has independent intuitive appeal (see below). In the derivation, it was required that M and N should be large. We can, however, study the effect of the choice of M and N (large or small) on the performance of effective capacity feedback. This is done in Section VI, where we also provide guidelines for selecting values of M and N under various situations.

The formula in (28) is intuitively satisfying; we will call it EFFCAP feedback. Consider the case when the network changes are very slow. Then, all N values of the average capacity will be the same, and each one will be equal to the capacity of the bottleneck link. Hence, the rate that is fed back to the ABR source will be the instantaneous free capacity of the bottleneck link; i.e., in this situation, EFFCAP is the same as INSTCAP. When the network variations are very fast, EFFCAP will be close to the mean capacity of the bottleneck link. Hence, EFFCAP behaves like INSTCAP for slow network changes and **adapts** to the mean bottleneck link capacity for fast changes. For intermediate rates of changes, EFFCAP is (necessarily) conservative and feeds back the *minimum* link rate.

V. Numerical and Simulation Results

In this section, we first compare our analytical results for the throughput of TCP, without ABR and with ABR with INSTCAP feedback, with simulation results from a hybrid TCP simulator involving actual TCP code, and a model for the network implemented in the loopback driver of a Linux Pentium machine. We show that the performance of TCP improves when ABR is used for end-to-end data transport below TCP. We then study the performance of the EFFCAP scheme and compare it with the INSTCAP scheme.

We recall from the previous section that the bottleneck link is Markov modulated. In our analysis, we have assumed that the modulating chain has two states, which we call the high state and the low state. In the low state, with some link capacity being used by higher priority traffic, the link capacity is some fraction of the link capacity in the high state (where the full link rate is available). We will assume that this fraction is 0.5. To reduce the number of parameters we have to deal with, we will also assume that the mean time in each state is the same, i.e., the

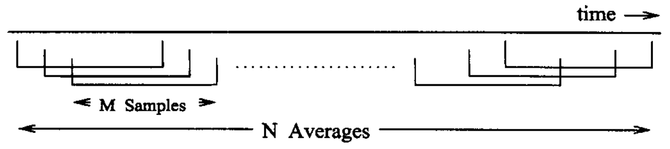


Fig. 7. Schematic of the windows used in the computation of the effective capacity-based rate feedback.

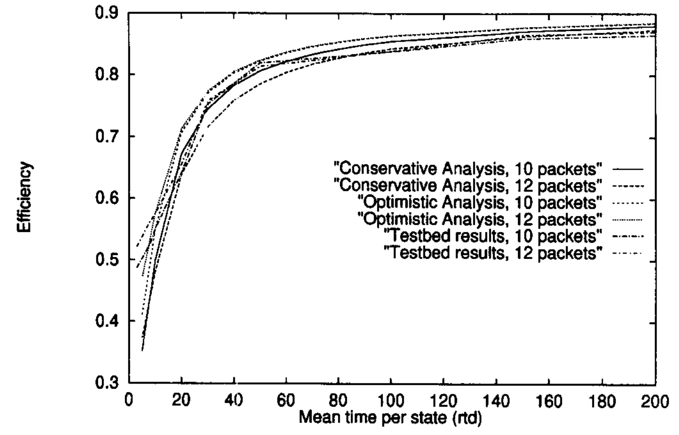


Fig. 8. Analysis and simulation results: INSTCAP feedback. Throughput of TCP over ABR: the round-trip propagation delay is 40 time units. The bottleneck link buffers are either 10 or 12 packets.

Markov chain is symmetric. We denote the mean time in each state by τ, and denote the mean time in each state normalized to Δ by ψ, i.e., $\psi := \tau/\Delta$. For example, if Δ is 200 ms, then $\psi = 2$ means that the mean time per state is 400 ms. Note that our analysis only applies to $\psi > 1$; in this section, we provide simulation results for a wide range of ψ, much smaller than 1, close to 1, and much larger than 1. A large value of ψ means that the network changes are slow compared to Δ, whereas $\psi \ll 1$ means that the network transients occur several times per RTT. In the Linux kernel implementation of our network simulator, the Markov chain can make transitions at most once every 30 ms. Hence, we take this also to be the measurement interval, and the explicit rate feedback interval (i.e., $s = 30$ ms).

We denote one packet transmission time at the bottleneck link in the *high-rate state* as **one time unit.** Thus, in all the results presented here, the packet transmission time in the low-rate state is two time units. Thus, if Δ is given in these time units, then the bandwidth-delay product in the high-rate state is Δ packets, and in the low-rate state it is $\Delta/2$ packets.

We plot the bottleneck link efficiency vs. mean time that it spends in each state (i.e., ψ). We define *efficiency* as the throughput as a fraction of the mean capacity of the bottleneck link. We include the TCP/IP headers in the throughput, but account for ATM headers as overhead. We use the words throughput and efficiency interchangeably. With the modulating Markov chain spending the same time in each state, the mean capacity of the link is 0.75.

Finally, before presenting the results, we note that Δ is an absolute parameter in the curves we present since it governs the round-trip "pipe." Thus, although ψ is normalized to Δ, the curves do not yield values for fixed ψ and varying Δ. Separate curves need to be plotted if Δ is changed.

A. Results for INSTCAP Feedback

Fig. 8 shows the throughput of TCP over ABR with the INSTCAP scheme.[5] Here, we compare an optimistic analysis, a conservative one (see Section III-C), and the testbed (i.e., simulation) results for different buffer sizes. In this example, the bandwidth delay product in the high-rate state is 40 packets, and the buffer sizes considered are 10 and 12 packets, respectively, 50% and 60% of the bandwidth delay product in the low-rate state.

In our analysis, the processes are embedded at multiples of one round-trip propagation delay, and the feedback from the bottleneck link is sent once every RTT. This feedback reaches the ABR source after one round-trip propagation delay. In the simulations, however, feedback is sent to the ABR source every 30 ms. This reaches the ABR source after one round-trip propagation delay.

We see that, except for very small ψ, the analysis and the simulations match to within a few percent. Both the analyzes are less than the observed throughputs by about 10%–20% for small ψ. In our analysis, we have assumed that packets leave back-to-back from the ABR source. When the bottleneck link-rate changes from high to low, as the packets arrive back-to-back, and the source sends at twice the rate of the bottleneck link, then for every two packets arriving at the bottleneck link, one gets queued. However, in reality, the packets need not arrive back-to-back and, hence, the queue buildup is slower. This means that the probability that packet loss occurs at the bottleneck link buffer is actually lower than in our analytical model. This effect becomes more and more significant as the rate of bottleneck link variations increases. However, we observe from the simulations that this effect is not significant for most values of ψ.

Fig. 9 shows the throughput of TCP *without* ABR. We can see that the simulation results give a throughput of upto 20% less than the analytical ones (note the scales of Figs. 9 and 8 are different). This occurs due to two reasons.

i) We assumed in our analysis that no loss occurs in the slow-start phase. It has been shown in [15] that if the bottleneck link buffer is less than 1/3 of the bandwidth-delay product (which corresponds to about 13 packets or 6500 B buffer in the high-rate state), loss will occur in the slow-start phase.
ii) We optimistically compute the throughput of TCP by using an upper bound on the "reward" in the loss cycle.

We see from Figs. 8 and 9 that ABR makes TCP throughput insensitive to buffer size variations. However, with TCP alone, there is a worsening of throughput with buffer reduction. This can be explained by the fact that once the ABR control loop has converged, the buffer size is immaterial, as no loss takes place when source and bottleneck link rate are the same. However, without ABR, TCP loses packets even when no transients occur.

An interesting result from this study is that TCP dynamics do not play an important part in the overall throughput for large ψ. This is intuitively understandable for the reason described above

[5]Even if $\psi \rightarrow \infty$, the throughput of TCP over ABR will not go to 1 because of ATM overheads. For every 53 B transmitted, there are 5 B of ATM headers. Hence, the asymptotic throughput is approximately 90%.

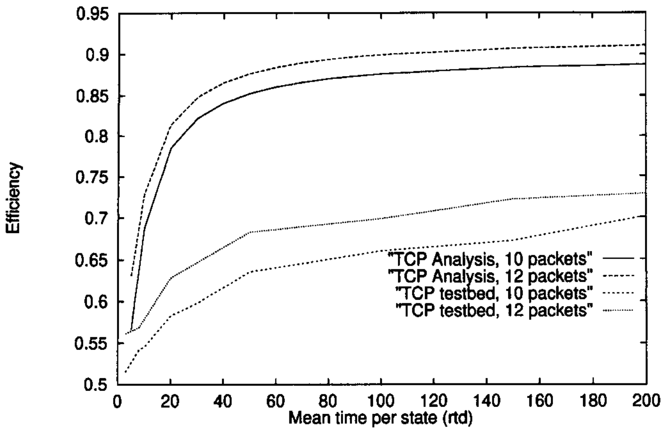


Fig. 9. Analysis and simulation results: throughput of TCP without ABR. The round-trip propagation delay is 40 time units. The bottleneck link buffers are either 10 or 12 packets.

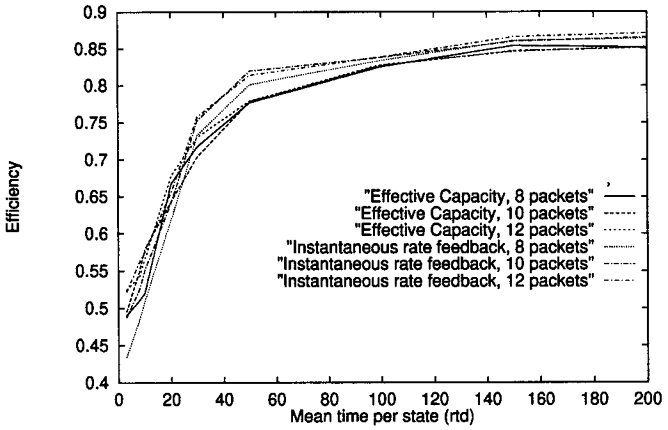


Fig. 10. Simulation results: comparison of the EFFCAP and INSTCAP feedback schemes for TCP over ABR for various bottleneck link buffers (8–12 packets). Δ is 40 time units. Here, $N = 49$ and $M = 7$ (see Fig. 7). In this figure, we compare their performance for relatively large ψ.

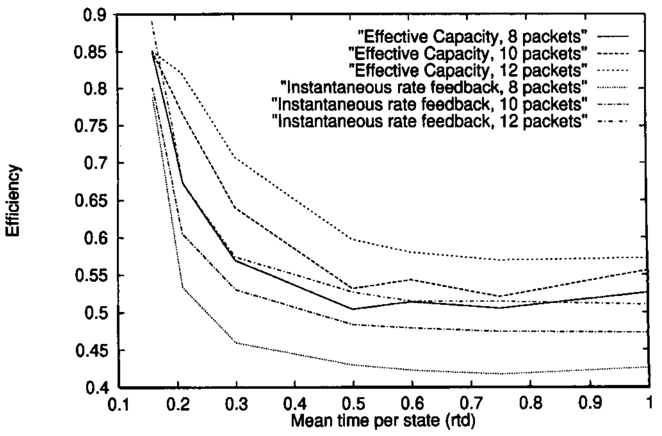


Fig. 11. Simulation results: comparison of the EFFCAP and INSTCAP feedback schemes for TCP over ABR for various bottleneck link buffers (8–12 packets). Δ is 40 time units. Here, $N = 49$ and $M = 7$ (see Fig. 7). In this figure, we compare their performances for small values of ψ.

(i.e., the TCP dynamics are "smoothed out" at the ABR buffer at the source once the ABR loop has converged). This point can also be seen from the fact that, even though our analysis of TCP window dynamics is approximate, it leads to a surprisingly good match with the simulations for TCP/ABR. However, as noted before, in the case of TCP alone, the simulation and analysis do not match very well, as the TCP dynamics plays an important role in the overall throughput.

B. Results for EFFCAP and INSTCAP Feedback

In Fig. 10, we use results from the testbed to compare the relative performance of EFFCAP and INSTCAP feedback schemes for ABR. Recall that the EFFCAP algorithm has two parameters, namely M, the number of samples used for each block average, and N, the number of blocks of M samples over which the minimum is taken. In this figure, the EFFCAP scheme uses $M = 7$, i.e., we average over one round-trip propagation delay[6] worth of samples. We also maintain a window of 8 Δ worth of averages, i.e., we maintain $N = (8 - 1) \times 7 = 49$ averages over which the bottleneck link returns the minimum to the ABR source. (We will discuss issues regarding choice of M and N in Section VI below.) The source adapts to this rate. In the case of the INSTCAP scheme, in the simulation, the rate is fed back every 30 ms.

We can see from Fig. 10 that for large ψ, the throughput with EFFCAP is worse than that with the INSTCAP scheme by about 3%–4%. This is because of the conservative nature of the EFFCAP algorithm; it takes the minimum of the available capacity over several blocks of time in an interval, and hence may feed back a lower rate than necessary. This result also shows that when ψ is large since rate changes are infrequent, it is sufficient to feedback the short term average rate.

However, we can see from Fig. 11 that for small ψ, the EFFCAP algorithm improves over the INSTCAP approach by 10%–20%. This is a significant improvement and it seems worthwhile to lose a few percent efficiency for large ψ to gain a large improvement for small ψ. When ψ is close to 1, if the short-term average rate is fed back (as INSTCAP does) then there are frequent mismatches between the source rate and the bottleneck service rate. The EFFCAP algorithm takes a minimum of the service-rate averages over several intervals, and hence, most probably feeds back the minimum link rate, thus minimizing rate mismatches. Note that the minimum link rate (0.5) normalized to the average rate (0.75) is 0.67. We will see in Section VI that with appropriate choice of M and N the throughput with EFFCAP can me made to approach this best case value.

To summarize, in Figs. 12 and 13, we have plotted the throughput of TCP over ABR using the two different feedback schemes. We have compared these results with the throughput of TCP *without* ABR. We can see that for $\phi > 20$ (Fig. 12) the throughput of TCP improves if ABR is employed for link level data transport, and the INSTCAP feedback is slightly better. When Δ is comparable to the time for which the link stays in each state (Fig. 13), then TCP performs better than TCP/ABR with INSTCAP feedback. This is because, in this regime, by feeding back the short-term average rate, the source rate and link rate are frequently mismatched, resulting in losses or starvation. On the other hand, EFFCAP feedback is able to keep the throughput

[6]A new sample is generated every 30 ms. The Δ is 200 ms in this example. Hence, $M = 200/30 = 6.667$, which we round up to 7.

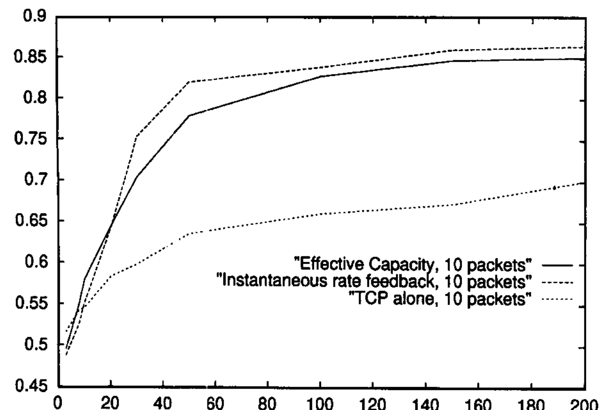


Fig. 12. Simulation results: comparison of throughput of TCP over ABR with effective capacity scheme, instantaneous rate feedback scheme and TCP without ABR for a buffer of ten packets, the other parameters remaining the same as in other simulations.

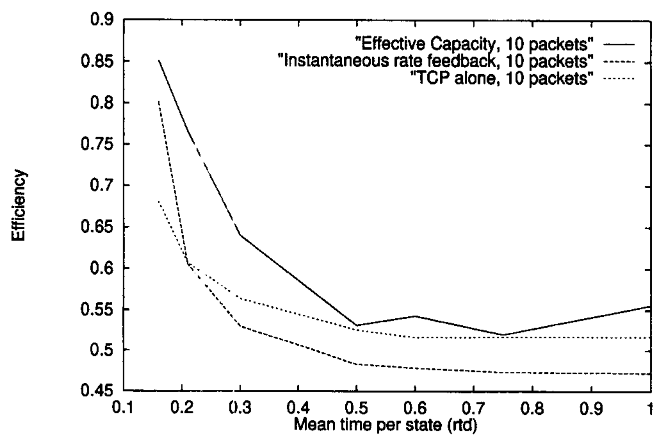


Fig. 13. Simulation results: comparison of throughput of TCP over ABR with effective capacity scheme, instantaneous rate feedback scheme and TCP without ABR for a buffer of ten packets, the other parameters remaining the same as in other simulations.

better than that of TCP even in this regime. These observations clearly bring out the merits of the EFFCAP scheme. Implicitly, EFFCAP feedback adapts to ψ, and performs better than TCP alone over a wide range of ψ. EFFCAP, however, requires the choice of two parameters M and N; in the next section, we provide guidelines for this choice.

VI. Choice of M and N for EFFCAP

From Figs. 10 and 11, we can identify three broad regions of performance in relation to ψ.

For $\psi = \tau/\Delta$ very large ($\psi > 50$), the rate mismatch occurs for a small fraction of τ. Also the rate mismatches are infrequent, implying infrequent losses, and higher throughput. Hence, it is sufficient to track the instantaneous available capacity by choosing small values of M and N. This is verified from Fig. 10 which shows that the INSTCAP feedback performs better in this region.

On the other hand, when τ is a small fraction of Δ ($\psi < 0.2$), there are frequent rate mismatches, but of very small durations as compared to Δ. Because of the rapid variations in the capacity, even a small M provides the mean capacity. Also, all the N averages roughly equal the mean capacity. Thus, the source essentially transmits at the mean capacity in EFFCAP as well as INSTCAP feedback. Hence, a high throughput for both types of feedback is seen from Fig. 11.

For the intermediate values of ψ ($0.5 < \psi < 20$), τ is comparable to Δ. Hence, rate mismatches are frequent, and persist relatively longer, causing the buffer to build up to a larger value. This leads to frequent losses. The throughput is adversely affected by TCP's blind adaptive window control. In this range, we expect to see severe throughput loss for sessions with large Δ. Therefore, in this region, we need to choose M and N to avoid rate mismatches. The capacity estimate should yield the *minimum* capacity (i.e., the smaller of the two rates in the Markov process), implying the need for small M and large N. A small M helps to avoid averaging over many samples and hence helps to pick up the two rates of the Markov chain, and a large N helps to pick out the minimum of the rates.

The selection of M and N cannot be based on the value of ψ alone; however, Δ is an absolute parameter in TCP window control and has a major effect on TCP throughput, and hence on the selection of M and N. The above discussion motivates a *small value of* M for all the ranges of ψ, a small N for large ψ, and large N for ψ close to 1 or smaller than 1. We also note that small values of ψ are more likely to occur in practice.

In the remainder of this section, we present simulation results that support the following rough design rule. If the measurement interval is s, then take M to be $\lceil \Delta/s \rceil$, i.e., the averages should be over one RTT. Take N to be in the range $8\lceil \Delta/s \rceil$ to $12\lceil \Delta/s \rceil$; i.e., multiple averages should be taken over 8–12 RTTs, and the minimum of these fed back.

We note here that the degradation of throughput in the intermediate range of values of ψ depends on the buffers available at the bottleneck link. This aspect is studied in [12].

A. Simulation Results and Discussion

Simulations were carried out on the hybrid simulator that was also used in Section V. As before, the capacity variation process is a two state Markov chain. In the high state, the capacity value is 100 kB/s, while in the low state it is 50 kB/s. The mean capacity is thus 75 kB/s. In all the simulations, the measurement and feedback interval $s = 30$ ms and link buffer is 5 kB (or 10 packets).

We introduce the following notation in the simulation results. $M : \Delta$ means that each average is calculated over $\lceil \Delta/s \rceil$ measurement intervals. $N : k\Delta$ means that $(k-1) \times \lceil \Delta/s \rceil$ averages are compared (or the memory of the algorithm is k RTTs). For example, let $\Delta = 200$ ms and $s = 30$ ms, then, $M : \Delta \Rightarrow M = 7$ measurement intervals, $N : 2\Delta \Rightarrow N = (2-1) \times 7 = 7$ (i.e., minimum of 7 averages). Similarly, $N : 8\Delta \Rightarrow N = (8-1) \times 7 = 49$ (i.e., minimum of 49 averages).

1) *Study of N:*

Case 1: Fixed Δ; varying τ

Fig. 14 shows the effect of N on the throughput for a given Δ, when τ (or equivalently the rate of capacity variation) is varying. These results corroborate the discussion at the beginning of Section VI for fixed Δ. Notice that when

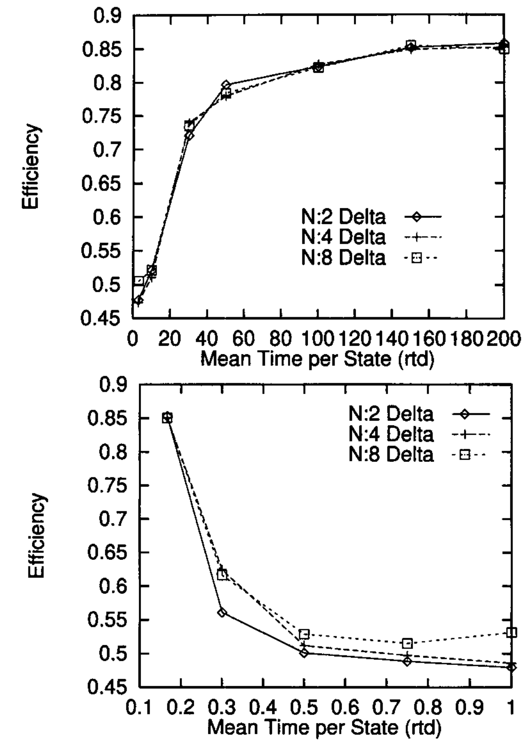


Fig. 14. Efficiency versus ψ for increasing values of N. M : Δ and N : 2Δ, 4Δ, 8Δ. We take $\Delta = 200$ ms. Thus, $M = 7$ and $N = 7$, 14, 49. τ is varied from 32 ms to 40 s.

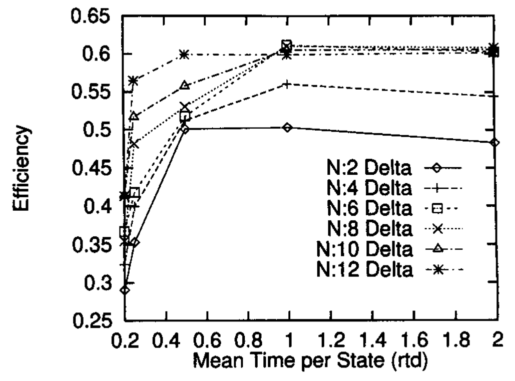


Fig. 15. Efficiency versus ψ. τ is 100 ms. Δ is varied (right to left) from 50 to 500 ms. M : Δ.

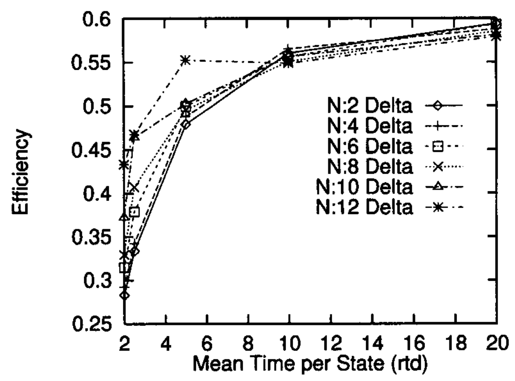


Fig. 16. Efficiency versus ψ. τ is 1000 ms. Δ is varied (right to left) from 50 to 500 ms. M : Δ.

$0.3 < \psi < 1$, as expected, an improvement in efficiency is seen for larger N.

Case 2: Fixed τ; varying Δ

Figs. 15 and 16 show the Efficiency variation with ψ for different values of N when τ is fixed and Δ is varied. Note that, N is different for different Δs on a N : $k\Delta$ curve. For example, N on the N : 4Δ curve for $\Delta = 50$ ms and $\Delta = 100$ ms is respectively 6 and 12.

Notice that, compared to Fig. 14, Figs. 15 and 16 show different efficiency variations with ψ. This is because, in the former case, τ is varied and Δ is constant, whereas in the latter case, τ is fixed and Δ varied. As indicated in Section VI, Δ is an absolute parameter which affects the throughput ($\psi = 2$ in Fig. 15 corresponds to $\Delta = 50$ ms and in Fig. 16 it corresponds to 500 ms). The considerable throughput difference demonstrates the dependence on the absolute value of Δ.

In Fig. 15, a substantial improvement in the throughput is seen as N increases. In addition, a larger N gives better throughput over a wider range of Δ. This is because, for a given Δ, a larger N tracks the minimum capacity value better. The minimum capacity is 50 kB/s, which is 66% of the mean capacity 75 kB/s. Hence, as N increases, efficiency increases to 0.6. Similarly, in Fig. 16, for $\psi < 8$, larger values of N improve the throughput. When $\psi > 10$, we see that smaller N performs better, but the improvement is negligible.

Note that for large ψ, N as low as 4Δ to 6Δ yields high throughput whereas for small ψ, N needs to be considerably higher (10Δ to 12Δ) to achieve high throughput. This can be explained as follows. We use M : Δ, which implies that for small ψ, the average over Δ yields the average rate, whereas for large ψ, it yields the peak or minimum rate. Thus, for large ψ, the minimum over just few Δs is adequate to yield a high throughput, whereas for small ψ, many more averages need to be minimized over to get the minimum rate. Notice, however, that for large ψ, increasing N does not seriously degrade efficiency.

In conclusion, the choice of N is based on ψ and Δ, but a value of N in the range $8\lceil\Delta/s\rceil$ to $12\lceil\Delta/s\rceil$ is a good compromise.

2) *Study of M:* It is already seen from Fig. 10 that for $\psi > 60$, a small value of M should be selected. To study the effect of M on the lower ranges of ψ, M is varied from 1 to 10 measuring intervals (i.e., s). Also, two settings of N are considered to differentiate its effect. The results are shown in Fig. 17 ($\tau = 1000$ ms) and Fig. 18 ($\tau = 100$ ms). The values of Δ are 50, 100, and 200 ms. Thus, the range of $\psi(= \tau/\Delta)$ is 5 to 20 in Fig. 17, and 0.5 to 2 in Fig. 18.

Recall that, in the intermediate range of ψ the bottleneck capacity estimate should yield the minimum capacity. With small M, the minimum value can be tracked better. This is seen from Fig. 17 for $\Delta = 50$ ms ($\psi = 20$); the throughput decreases slowly with increasing M. Notice from Fig. 17 that a larger value of N improves efficiency, as more samples implies a better chance of picking up the minimum rate. In Fig. 18, for N : 2Δ, the throughput is insensitive to the variation in M. Again increasing N improves throughput. Insensitivity to M is observed

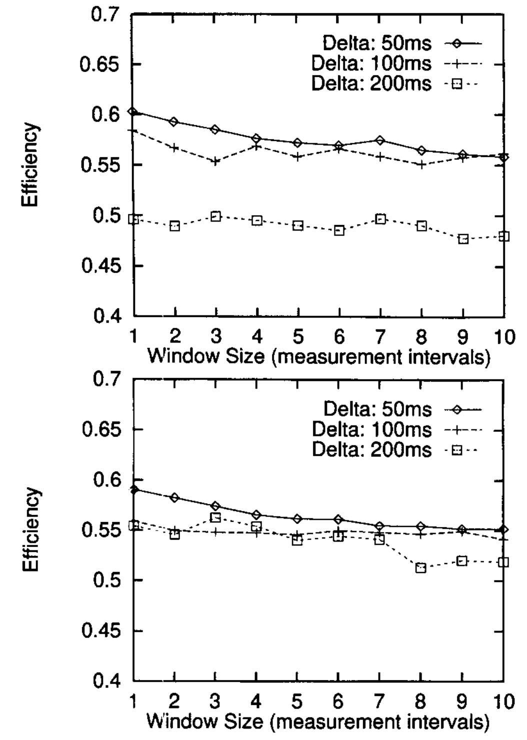


Fig. 17. Efficiency versus M. τ is 1000 ms and Δ takes values 50, 100, and 200 ms. The top graph has $N : 2\Delta$ and the bottom graph $N : 12\Delta$.

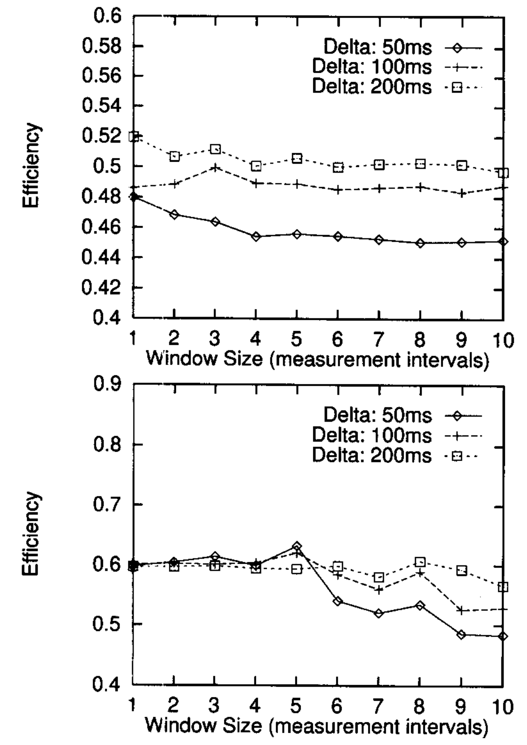


Fig. 18. Efficiency vs M. τ is 100 ms and Δ takes values 50, 100, and 200 ms. The top graph has $N : 2\Delta$ and the bottom graph $N : 12\Delta$.

in the case of $N : 12\Delta$ for $\psi = 0.5$, but for larger ψ, 1 or 2, i.e., $\Delta = 100$ or 50 ms, a 10%–15% decrease in the throughput is seen for larger values of M. This is because $N : 12\Delta$ is not sufficient to track the minimum with larger values of M.

We conclude that in the intermediate range of ψ, the throughput is not very sensitive to M. For small Δ and larger ψ (e.g., $\Delta = 50$ ms, $\psi = 20$), a small M performs better since it is possible to track the instantaneous rate. In general, a small value of M improves the throughput in all the ranges. In Figs. 17 and 18, $s = 30$ ms and we have $\lceil \Delta/s \rceil$ equal to 2, 4, and 7. We notice that, as a rule of thumb, $M : \Delta$ gives good performance in each case.

B. Implementation of EFFCAP when Δ is Not Known

The simulation results presented in Sections VI-A-1 and -2 have supported the guidelines for choosing M and N presented in Section VI. We find that these parameters depend on the RTT Δ for the connection, a parameter that will not be known at the switch at which the EFFCAP feedback is being computed. However, Δ would be (approximately) known at the source node. This knowledge could come either during the ATM connection setup, or from the RTT estimate at the TCP layer in the source. Hence, one possibility is for the network to provide INSTCAP feedbacks (i.e., the short term average capacity over the measurement interval s), and the source node can then easily compute the EFFCAP feedback value. The INSTCAP feedback can be provided in ATM Resource Management (RM) cells [1].

VII. TCP/ABR WITH EFFCAP FEEDBACK: FAIRNESS

It is seen that TCP alone is unfair toward sessions that have larger RTTs. It may be expected, however, that TCP sessions over ABR will get a fair share of the available capacity. In [19], the fairness of the INSTCAP feedback was investigated and it was shown that for slow variations of the available capacity, TCP sessions over ABR employing the INSTCAP feedback achieve fairness. In this section, we study the fairness of TCP sessions over ABR with the EFFCAP feedback scheme.

Denote by Δ_1 and Δ_2, the RTTs of Session 1 and Session 2, respectively. Other notations are as described earlier (subscripts denote the session number). In the simulations, we use $\Delta_1 = 240$ and $\Delta_2 = 360$ ms. The link buffer size is 9000 B (18 packets). In the following graphs ψ is τ (mean time per state of the Markov chain) divided by larger Δ_i, i.e., $\Delta_2 = 360$ ms. Simulations are carried out by calculating the EFFCAP by two different ways, as explained below.

A. Case 1: Effective Capacity with $M_i : \Delta_i$

In this case, we calculate the EFFCAP for each session independently. This is done by selecting M_i proportional to Δ_i, that is (with a 30-ms update interval) we select $M = 8$ for Session 1 and $M = 12$ for Session 2. We take $N_i : 12\Delta_i$, i.e., N_1 is 88 and N_2 is 132 (see Section VI). EFFCAP$_i$ is computed with M_i and N_i; session i is fedback 1/2 of EFFCAP$_i$.

Fig. 19 shows the simulation results for $\psi \leq 1$. Fig. 20 shows the comparison of TCP throughputs and TCP/ABR throughputs. It can be seen that for very small values of ψ ($\psi < 0.3$), the sessions receive equal throughput. However, for $0.3 < \psi < 20$, unfairness is seen toward the session with larger propagation delay. This can be explained from the discussion in Section VI. In this range of ψ, *due to frequent rate mismatches and hence losses, TCP behavior is dominant.* A packet drop leads to greater

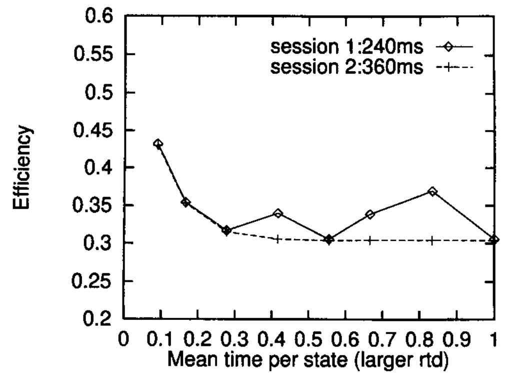


Fig. 19. Efficiency versus ψ (mean time per state normalized to $\Delta_2 = 360$ ms). $M_i : \Delta_i$ and $N_i : 12\Delta_i$. Each session is fed back the fair share (half) of the EFFCAP calculated.

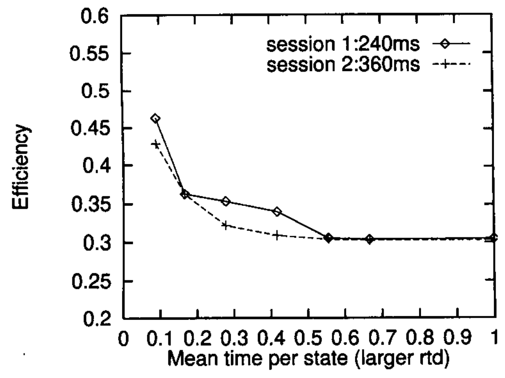


Fig. 21. Efficiency versus ψ (mean time per state normalized to $\Delta_2 = 360$ ms). $M = [(\Delta_1 + \Delta_2)]/2 = 10$ measurement intervals. $N = 110$ averages. Each session is fed back the fair share (half) of the EFFCAP calculated.

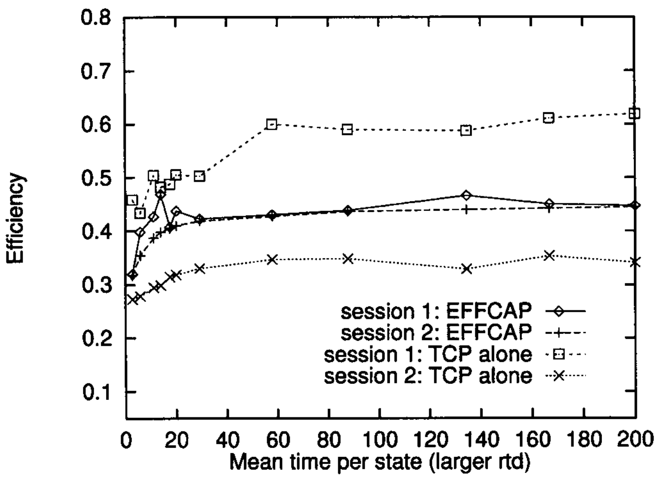


Fig. 20. Comparison between Efficiency of sessions with TCP alone and TCP over ABR employing EFFCAP feedback (Case 1: $M_i : \Delta_i$).

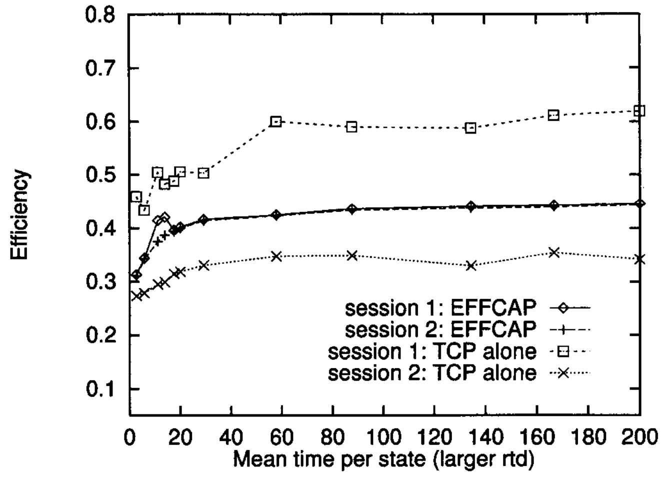


Fig. 22. Comparison between Efficiency of sessions with TCP alone and TCP over ABR employing EFFCAP feedback [Case 2: $M = (\Delta_1 + \Delta_2)/2$].

throughput decrease for a session with larger Δ than for a session with smaller Δ. The throughputs with TCP over ABR are, however, fairer than with TCP alone which results in grossly unfair throughputs.

B. Case 2: Effective Capacity with $M : (\Delta_1 + \Delta_2)/2$

In this simulation, M corresponds to the average of Δ_1 and Δ_2, i.e., 300 ms (ten measurement intervals). With $N : 12[(\Delta_1 + \Delta_2)/2]$, we have $N = 110$. By choosing M and N this way, the rate calculation is made independent of individual RTTs.

Fig. 21 shows results for $\psi < 1$. Fig. 22 shows results for TCP, as well as TCP/ABR. We notice that EFFCAP calculated in this way yields somewhat better fairness than the scheme used in Case 1. It is also seen that better fairness is obtained even in the intermediate range of ψ. However, there is a drop in the overall efficiency. This is because the throughput of the session with smaller Δ is reduced.

There is a slight decrease in the overall efficiency with TCP over ABR, but note that with TCP over ABR, the link actually carries 10% more bytes (the ATM overhead) than with TCP alone! We have also found that for $\psi < 20$, EFFCAP gives relatively better fairness than INSTCAP, based on the results for the latter that were reported in [19].

Finally, we observe that if EFFCAP is implemented with the approach suggested in Section VI-B, then the Case 1 ($M_i : \Delta_i$) discussed in this section is actually achieved.

VIII. Conclusion

In this paper, we set out to understand if running an adaptive window congestion control (TCP) over an endpoint-to-endpoint explicit rate control (ATM/ABR) is beneficial for end-to-end throughput performance. We have studied two kinds of explicit rate feedback: INSTCAP, in which the short-term average capacity of the bottleneck link is fed back, and EFFCAP, in which a measure motivated by a large deviations effective service capacity, and based on the longer term history is fed back. We have seen, from the analysis and simulation results, that the throughput of TCP over ABR depends on the relative rate of capacity variation with respect to the round-trip delay in the connection. For slow variations of the link capacity (the capacity varies over periods of over 20 times the round-trip delay) the improvement with INSTCAP is significant (25%–30%), whereas if the rate variations are over durations comparable to the round-trip delay, then the TCP throughput with ABR can be slightly worse than with TCP alone. An interesting observation is that TCP dynamics do not appear to play an important part in the overall throughput when the capacity variations are slow.

EFFCAP rate feedback has the remarkable property of automatically adapting what it feeds back to the rate of variation of the bottleneck link capacity, and thus achieves higher throughputs than INSTCAP, always beating the throughput of TCP alone. The EFFCAP computation involves two parameters M and N; at each update epoch, EFFCAP feeds back the minimum of N short-term averages, each taken over M measurement intervals. For EFFCAP to display its adaptive behavior, these parameters need to be chosen properly. Based on extensive simulations, we find that, as a broad guideline (for the buffer sizes that we studied) for ideal performance, EFFCAP should be used with each average being taken over a RTT, and the minimum should be taken over several averages taken over the previous 8–12 RTTs.

Finally, we find that TCP over ABR with EFFCAP feedback provides throughput fairness between sessions that have different RTTs.

REFERENCES

[1] *The ATM Forum Traffic Management Specification Version 4.0*, Apr. 1996.

[2] S. P. Abraham and A. Kumar, "A new approach for distributed explicit rate control of elastic traffic in an integrated packet network," *IEEE/ACM Trans. Networking*, vol. 9, pp. 15–30, Feb. 2001.

[3] F. Bonomi and K. W. Fendick, "The rate-based flow control framework for the available bit rate ATM service," *IEEE Network*, pp. 25–39, Mar./Apr. 1995

[4] H. Chaskar, T. V. Lakshman, and U. Madhow, "TCP over wireless with link level error control: Analysis and design methodology," *IEEE/ACM Trans. Networking*, vol. 7, pp. 605–615, Oct. 1999.

[5] C. Fang, H. Chen, and J. Hutchins, "A simulation of TCP performance in ATM networks," in *Proc. IEEE Globecom'94*, 1994.

[6] B. Feng, D. Ghosal, and N. Kannappan, "Impact of ATM ABR control on the performance of TCP-Tahoe and TCP-Reno," in *Proc. IEEE Globecom'97*, 1997.

[7] V. Jacobson. (1990, Apr.) Modified TCP Congestion Avoidance Algorithm. end2end-interest mailing list . [Online]. Available: ftp://ftp.isi.edu/end2end/end2end-interest-1990.mail.

[8] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, and B. Vandalore, "The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks," *IEEE/ACM Trans. Networking*, vol. 8, pp. 81–98, Feb. 2000.

[9] S. Kalyanaraman *et al.*, "Buffer requirements for TCP/IP over ABR," in *Proc. IEEE ATM'96 Workshop*, San Francisco, CA, Aug. 1996.

[10] S. Kalyanaraman *et al.*, " Performance of TCP over ABR on ATM backbone and with various VBR traffic patterns," in *Proc. ICC'97*, Montreal, Canada, June 1997.

[11] L. Kalampoukas and A. Varma, "Analysis of source policy and its effects on TCP in rate-controlled ATM networks," *IEEE/ACM Trans. Networking*, vol. 6, pp. 599–610, Oct. 1998.

[12] A. Karnik, "Performance of TCP congestion control with rate feedback: TCP/ABR and rate adaptive TCP/IP," M. Eng. thesis, Indian Institute of Science, Bangalore, India, Jan. 1999.

[13] A. Karnik and A. Kumar, "Performance of TCP congestion control with rate feedback: Rate adaptive TCP (RATCP)," in *Proc. IEEE Globecom 2000*, San Francisco, CA, Nov. 2000.

[14] A. Kumar, "Comparative performance analysis of versions of TCP in a local network with a lossy link," *IEEE/ACM Trans. Networking*, vol. 6, pp. 485–498, Aug. 1998.

[15] T. V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth delay products and random loss," *IEEE/ACM Trans. Networking*, vol. 5, pp. 336–350, June 1997.

[16] T. J. Ott and N. Aggarwal, "TCP over ATM: ABR or UBR," unpublished.

[17] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," *IEEE/ACM Trans. Networking*, vol. 8, pp. 133–145, Apr. 2000.

[18] A. Romanov and S. Floyd, "Dynamics of TCP traffic over ATM networks," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 633–641, May 1995.

[19] S. G. Sanjay, "TCP over end-to-end ABR: A study of TCP performance with end-to-end rate control and stochastic available capacity," M. Eng. thesis, Indian Institute of Science, Bangalore, India, Jan. 1998.

[20] G. de Veciana and J. Walrand, "Effective bandwidths: Call admission, traffic policing and filtering for ATM networks," *Queuing Systems Theory and Applications (QUESTA)*, 1994.

[21] R. Wolff, *Stochastic Modeling and the Theory of Queues*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

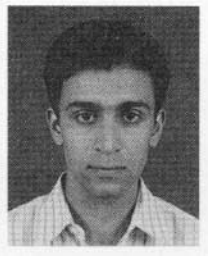
Sanjay Shakkottai received the B.E. degree in electronics and communications engineering from University Visvesvaraiya College, Bangalore, India, and the M.E. degree in communications engineering from the Indian Institute of Science, Bangalore. He is currently working toward the Ph.D. degree at the University of Illinois at Urbana-Champaign.

Anurag Kumar (S'81–M'81–SM'92) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, in 1977, and the Ph.D. degree from Cornell University, Ithaca, NY, in 1981.

He was with AT&T Bell Labs, Holmdel, NJ, for over six years. Since 1988, he has been with the Indian Institute of Science (IISc), Bangalore, India, in the Department of Electrical Communication Engineering, where he is now a Professor. His research interests are in the areas of modeling, analysis, control, and optimization problems arising in communication networks and distributed systems.

Dr. Kumar is a Fellow of the Indian National Academy of Engineering.

Aditya Karnik received the B.E. degree in electronics and communications engineering from the University of Pune, Pune, India, and the M.E. degree in communications engineering from the Indian Institute of Science, Bangalore, where he is currently working toward the Ph.D. degree in communications engineering.



Ajit Anvekar received the B.E. degree in electronics and communications engineering from SIT, Tumkur, India.

He was a Project Assistant in the ERNET Lab, Electrical Communication Engineering Department, Indian Institute of Science (IISc), Bangalore, until October 2000. He is now with the IISc Simputer Consortium, Bangalore, India.