TCP OVER END-TO-END ABR: A STUDY OF TCP PERFORMANCE WITH END-TO-END RATE CONTROL AND STOCHASTIC AVAILABLE CAPACITY*

S. G. Sanjay† and Anurag Kumar
Dept. of Electrical Communication Engg.
Indian Institute of Science (IISc), Bangalore 560 012, INDIA
e-mail: anurag@ece.iisc.ernet.in

Abstract

We develop an analysis using which we compare the performance of TCP, with and without end-to-end ATM/ABR transport, when the network bandwidth is shared with time-varying CBR/VBR traffic. We show that, over ABR, the performance of TCP improves by more than 20% if the network bottleneck bandwidth variations are slow. Further, we find that TCP over ABR is relatively insensitive to bottleneck buffer size. We then validate the analytical results with results from a hybrid simulation on a TCP testbed that we have developed. We use this simulation to study two mechanisms for bottleneck rate feedback at the ABR level. We find that an *effective capacity* based feedback is adaptive to the rate of bandwidth variations at the bottleneck link, and thus yields good performance over a wide range of rates of bottleneck bandwidth variation.

1 Introduction

As the ABR service does not guarantee end-to-end reliable transport of data to the applications above it ([2]), in the first deployments of ATM networks, the Internet's Transport Control Protocol (TCP) is used to ensure end-to-end reliability for data applications. TCP has its own adaptive window based congestion control mechanism that serves to slow down sources during network congestion. Hence, it is important to know whether the adaptive window control at the TCP level, and the rate-based control at the ABR level interact beneficially from the point of view of application level throughput. In this paper, we consider the situation in which the ATM network extends upto the TCP endpoints; i.e., end-to-end ABR (as opposed to edge-to-edge ABR), see Figure 1.

Consider the hypothetical situation in which the control loop has zero delay. In such a case, the ABR source of a session (i.e., the ATM network interface card (NIC) at the source node) will follow the variations in the bandwidth of the bottleneck link without delay. As a result, no loss will take place in the network. The TCP window will grow, and once the TCP window size exceeds the window required to fill the round trip pipe, the packets will be buffered in the source buffer. Hence, we can see that congestion is effectively pushed to the network edge. As the source buffer would be much larger than the maximum window size, the TCP window will remain fixed at the maximum window size and congestion control will become a purely rate based one. If ABR service was not used, however, TCP would increase its window, overshoot the required window size, and then due to packet loss, would again reduce the window size. Hence, it is clear that, for a zero delay in the control loop, end-to-end ABR will definitely improve the throughput of TCP.

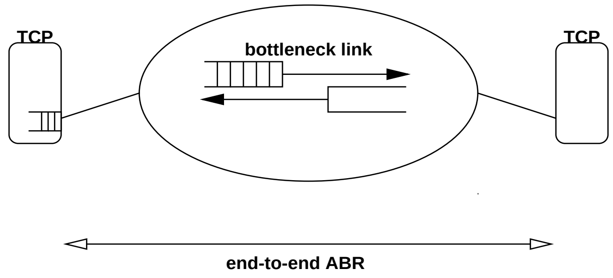

Figure 1: The scenario under study.

When variations in the bottleneck bandwidth do occur, however, and there is delay in the ABR control loop as in Figure 1, it is not clear whether there will be any improvement. Many simulation studies [8], [4], [7] have been carried out to study the interaction between the TCP and ATM/ABR control loops. Analytical work on TCP over ABR does not seem to exist in the literature. In this paper, we report the results of a study of the throughput of a TCP connection with a bottleneck link, with time varying available bandwidth, and a large round trip delay, with and without ATM/ABR transport. We then validate the analytical results using a simulation.

We also aim at optimizing the throughput of the TCP connection over ABR by feeding back a bottle-

*Research supported by a grant from Nortel
†Currently with the Coordinated Sciences Lab, University of Illinois, Urbana-Champagne, IL, USA

neck bandwidth rate that is more effective in preventing loss at the bottleneck buffers. We find that an approach based on *effective link capacity* is better than instantaneous rate feedback, and also better than mean rate feedback. Much of this work has been done using a hybrid simulator we have developed. Our results show that different types of bottleneck bandwidth feedbacks are needed for slowly varying bottleneck bandwidth, rapidly varying bottleneck bandwidth and the intermediate regime. The effective capacity based feedback *adapts* itself to the rate of bandwidth variation.

2 System Model and its Analysis

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 Figure 1. Let us assume that only one link (called the *bottleneck link*) causes significant queueing delays in this connection, the delays due to the other links being fixed (i.e., only fixed propagation delays are introduced due to the other links).

A more detailed model of this is shown in Figure 2. 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, and adapts to this rate feedback.

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.

In the first part of our study, we assume that the ABR feedback is an *instantaneous* rate feedback scheme; i.e., the bottleneck link periodically feeds back its instantaneous free capacity to the ABR source[1]. This feedback reaches after one round trip propagation delay.

2.1 TCP over End-to-End ATM/ABR

At time t, the ABR source transmits at the 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/sec* which is a function of an independent Markov chain. We assume that there is a 2 state Markov chain modulating the channel. In each state,

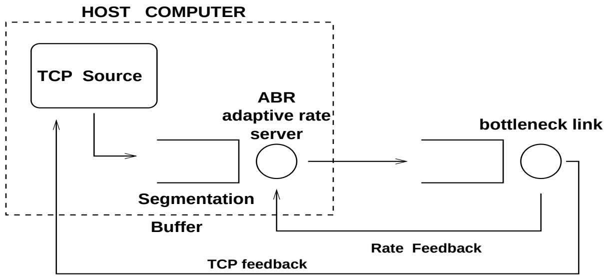

Figure 2: Schematic of the host and the bottleneck link, showing the segmentation buffer, and the ABR and TCP feedback loops.

the bottleneck link capacity is deterministic[2]. If the buffer is full when a cell arrives to 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; it is akin to the Partial Packet Discard proposed in [9]. If the packet is not lost, it gets serviced at rate R_t^{-1} (assumed constant over the service time of the packet), and reaches the destination after some deterministic delay. The destination ATM layer reassembles the packet and delivers it to the TCP receiver. The TCP receiver responds with an ACK (acknowledgement) which, after some delay (propagation + processing delay) reaches the source. The TCP source responds by increasing the window size.

The TCP window evolution can be modeled in several ways (see [6], [5]). In this study, we model the TCP window adjustments in the congestion avoidance phase (for the original TCP algorithm as proposed in [3] by Van Jacobson) probabilistically as follows: every time a non-duplicate ACK (an acknowledgement that requests for a packet that has not been acknowledged earlier) arrives at the source, the window size W_t increases by one with probability $\frac{1}{W_t}$.

On the other hand, 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.

2.1.1 Queueing Network Model

Figure 4 is a closed queueing network representation of the TCP over ABR session. We model the TCP connection during the data transfer phase; hence the data packets are assumed to be of fixed length. The buffer of the segmentation queue at the source host is assumed to be infinite in size. There are as many packets in this buffer as the number of untransmitted packets in

[1]In practice, this means a short-term average available rate feedback

[2]The lower service rate can be viewed as a minimum cell rate (MCR) available to the session.

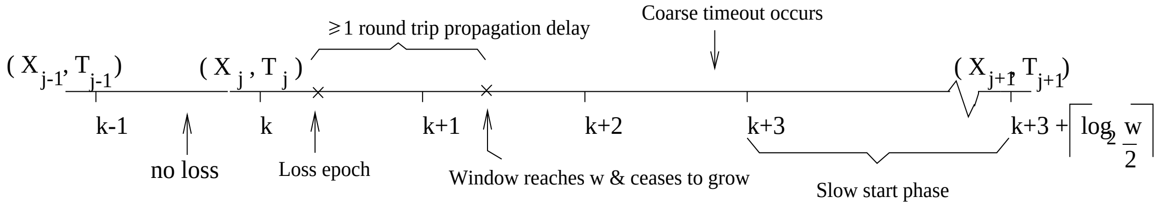

Figure 3: The embedded process $\{(X_j, T_j), j \geq 0\}$

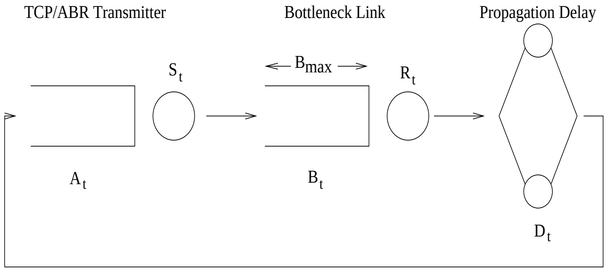

Figure 4: Queueing model of TCP/end-to-end ABR

the window. The service time at this buffer models the time taken to transmit an entire TCP packet worth of ATM cells. Owing to the feedback rate control, the service rate follows the rate of the bottleneck link. We assume that the rate does not change during the transmission of the cells from a single TCP packet, and hence we model the TCP packet service time as deterministic at this buffer. The service time (or equivalently, the service rate) follows the bottleneck link service rate with a delay of Δ units of time, Δ being the round trip (fixed) propagation delay.

The bottleneck link is modeled as a finite buffer queue with deterministic packet service time with the service time (or rate) Markov modulated by an independent Markov chain on two states 0 and 1. The round trip propagation delay Δ is modeled by an infinite server queue with service time Δ.

With "packets" being read as "full TCP packets", let A_t be the number of packets in the segmentation buffer at the host at time t, let B_t be the number of packets in the bottleneck link buffer at time t, and let D_t be the number of packets in the propagation queue at time t. Let R_t be the *service time* of a packet at the bottleneck link with $R_t \in \{r_0, r_1\}$. 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**. Finally, let S_t be the service time of a packet at the source link. S_t follows R_t with delay Δ, the round trip propagation delay, i.e., $S_t = R_{t-\Delta}$, and $S_t \in \{r_0, r_1\}$. Since the instantaneous rate of the bottleneck link is fed back, we call this the *instantaneous rate feedback* scheme.

2.1.2 Analysis of the Queueing Model

Consider the vector process

$$\{Z_t, t \geq 0\} := \{(A_t, B_t, D_t, R_t, S_t), t \geq 0\} \quad (1)$$

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

$$\{\tilde{Z}_k, k \geq 0\} = \{Z_{t_k}, k \geq 0\} \quad (2)$$

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

In the following analysis, we will make the following assumptions. (i) We assume that the rate modulating Markov chain is embedded at the epochs $(t_0, t_1, \ldots)$. (ii) We assume that there is no loss in the slow start phase of TCP. In [6], the authors show that loss will occur in the slow start phase if $\frac{B_{max}}{\frac{\Delta}{r_0}+1} < \frac{1}{3}$ even if no rate change occurs in the slow start phase. However, 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. Hence, this assumption is valid for the case of TCP alone only if $\frac{B_{max}}{\frac{\Delta}{r_0}+1} > \frac{1}{3}$.

Observe that packets in the propagation delay queue (see Figure 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. Figure 3 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 then will start a coarse timer. We assume that this timeout will occur in the interval (t_{k+2}, t_{k+3}) (see Figure 3), 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.

At time t_k, let $\tilde{Z}_k = (a, b, d, r, s)$. Note that, since no loss has occurred (since last recovery) until t_k, therefore, the TCP window at t_k is $a + b + d$. Now, given $\tilde{Z}_k$, and assuming that (i) packet transmissions do not straddle the embedded epochs, and (ii) packets arrive back-to-back into the segmentation buffer during any interval (t_k, t_{k+1}); (this leads to a conservative estimate of TCP throughput; see the discussion following Figure 8 below), 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. Suppose this window is w, then the congestion avoidance threshold in the next recovery cycle will be $m := \lceil \frac{w}{2} \rceil$. It will take approximately $\lceil \log_2 m \rceil$ round trip times (each of length Δ) to reach the congestion avoidance threshold. Assuming that no loss occurs during the slow start phase (this is true if B_{max} is not too small [6]), at $k' = k + 3 + \lceil \log_2 m \rceil$, we can determine the distribution of $\tilde{Z}_{k'}$. With the above description in mind, define

$$T_0 = t_0 = 0 \text{ and } X_0 = \tilde{Z}_0 = (1, 0, 0, r_0, r_0) \qquad (3)$$

For $k \geq 1$,

$$T_k = \begin{cases} T_{k-1} + \Delta & \text{if no loss occurs} \\ & \text{in } (T_{k-1}, T_{k-1} + \Delta) \\ T_{k-1} + (3 + \lceil \log_2 \frac{w}{2} \rceil)\Delta & \text{if loss occurs} \\ & \text{in } (T_{k-1}, T_{k-1} + \Delta) \\ & \text{and the loss window} \\ & \text{is } w \end{cases} \qquad (4)$$

and $X_k = \tilde{Z}_{T_k}$. Define

$$U_k = (T_{k+1} - T_k) \quad \text{for } k \geq 0 \qquad (5)$$

We can argue that $\{X_k, k \geq 0\}$ is a Markov chain (see [10]). Further, given T_k and X_k, the distribution of T_{k+1} can be computed without any knowledge of its past history. Hence, the process $\{(X_k, T_k), k \geq 0\}$ is a Markov Renewal Process (MRP) (See [13]). It is this MRP that is our model for TCP/ABR.

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 equal to the number of packets successfully transfered in the interval. Let $\pi(x)$ denote the stationary probability distribution of the Markov chain $\{X_k, k \geq 0\}$. Denote by $\gamma_{TCP/ABR}$, the throughput of TCP over ABR. Then, by the Markov renewal-reward theorem ([13]), we have

$$\gamma_{TCP/ABR} = \frac{E_\pi V}{E_\pi U} \qquad (6)$$

where $E_\pi(\cdot)$ denotes the expectation w.r.t. the stationary distribution $\pi(x)$. In [10], we show how π, $E_\pi V$ and $E_\pi U$ are obtained; owing to lack of space, we skip these details here. We note that this approach leads to optimistic and conservative values of TCP throughput.

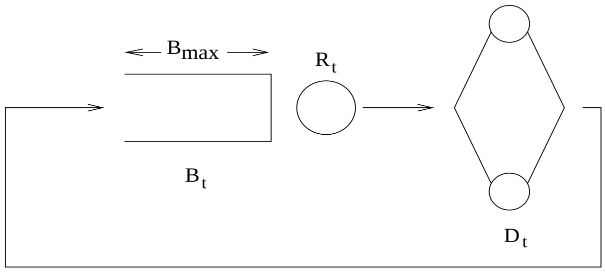

Figure 5: Queueing model of TCP without ABR

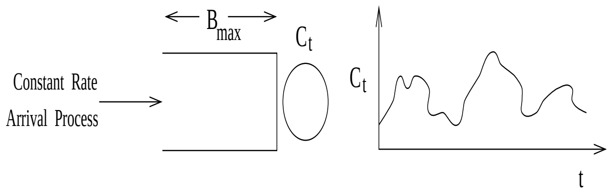

Figure 6: Single server queue with time varying capacity

2.2 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, as we see in Figure 5. The assumptions we make in this analysis, however, lead to an optimistic estimate of the throughput. The analysis is analogous to that in Section 2.1.

3 Effective Capacity 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 Figure 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 [12]). In this case, consider a slotted time queueing 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 γ_{max} such that the desired QoS (cell loss probability $\leq \epsilon$) is

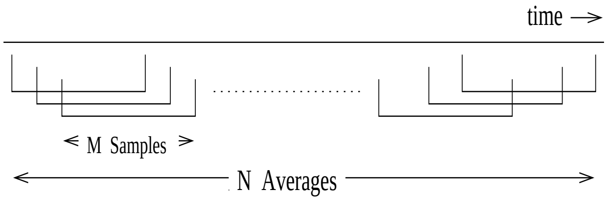

Figure 7: Schematic of the windows used in the computation of effective capacity

achieved. In [12], the following asymptotic condition is considered. If X is a random variable which represents the queue length, then[3]

$$\lim_{B_{max}\to\infty} \frac{1}{B_{max}} \log P(X > B_{max}) < -\delta \tag{7}$$

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

$$\gamma < \frac{-1}{\delta} \lim_{n\to\infty} \frac{1}{n+1} \log E e^{-\delta \sum_{i=0}^{n} C_i} \tag{8}$$

where $\delta = \frac{-log\epsilon}{B_{max}}$. Let us denote the expression on the right hand side of Equation 8 as Γ_{eff}. Then, Γ_{eff} can be called the *effective capacity* of the server. If $\epsilon \to 1$, then $\Gamma_{eff} \to EC$ and as $\epsilon \to 0$, $\Gamma_{eff} \to C_{min}$ which is what we intuitively expect. For all other values of ϵ, $\Gamma_{eff} \in (C_{min}, EC)$.

We apply this effective capacity approach to our problem by making the ABR source (see Figure 2) adapt to the effective bandwidth of the bottleneck link server. We compute the effective capacity of the Markov modulated bottleneck link server using Equation 8. 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; $\mu \in (EC, C_{min})$. For a given Markov modulating process, we find that μ which maximizes TCP throughput. We will assume that this is the effective capacity of the bottleneck link. Now, using Equation 8, 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 queueing model for TCP over ABR can be analyzed in a manner analogous to that in Section 2.1.2. We find from the analysis that for several sets of parameters, the value of δ which maximizes TCP throughput is consistently very large. This is as expected since TCP performance is very sensitive to loss.

[3] All logarithms are taken to the base e

3.1 Effective Capacity Computation

We develop an on-line method of computing the effective bandwidth. The approach is based on Equation 8, and the observation at the end of the previous section that δ is very large.

Consider a discrete time model of TCP over ABR, the slot length being κ time units; κ being the minimum update interval of the rate feedback. We approximate the expression for effective bandwidth in Equation 8 by replacing $n \to \infty$ by a large finite M[4], i.e.,

$$\Gamma_{eff} \approx \frac{-1}{M\delta} \log E e^{-\delta \sum_{i=1}^{M} C_i} \tag{9}$$

What we now have is an effective capacity computation performed over $M\kappa$ units of time. We 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 $M\kappa$ 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

$$\begin{aligned} \Gamma_{eff} &\approx \frac{-1}{M\delta} \log \frac{1}{N} \sum_{j=1}^{N} e^{-\delta \sum_{i=1}^{M} C_{ij}} && (10) \\ &= \frac{-1}{M\delta} \log \frac{1}{N} - \frac{1}{M\delta} \log \sum_{j=1}^{N} e^{-\delta \sum_{i=1}^{M} C_{ij}} && (11) \end{aligned}$$

Taking δ to be large (see Section 3), we get

$$\begin{aligned} &\underset{\delta\to\infty}{\approx} \frac{-1}{M\delta} \log e^{-\delta(\min_{j\in N} \sum_{i=1}^{M} C_{ij})} && (12) \\ &= \min_{j\in N} \frac{1}{M} \sum_{i=1}^{M} C_{ij} && (13) \end{aligned}$$

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

This algorithm is intuitively satisfying. Consider the case when the network changes are very slow. Then, all N values 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 (actually a short-term average free capacity). When the network variations are very fast, the rate fed back will be the mean capacity of the bottleneck link which is what

[4] A large value of M means that for rapidly varying bottleneck link capacity, the algorithm computes the effective capacity which is close to that in Equation 8. However, for slow variations, a large M means that the rate fed back to the ABR source adapts very sluggishly to the bottleneck link variations. In this case, a small value of M is preferable. In the limiting case when $M = 1$ and $N = 1$ (see Equation 13), the effective capacity scheme becomes the same as the instantaneous rate feedback scheme.

should be done to get the best throughput. Hence, this algorithm behaves like the instantaneous rate feedback scheme for slow network changes and **adapts** to the mean bottleneck link capacity for fast changes. For intermediate rates of change, it is conservative and feeds back the *minimum* link rate.

4 Numerical Results

In this section, we first compare our analytical results for the throughput of TCP, with and without ABR (with the instantaneous rate feedback scheme), with simulation results from a hybrid TCP simulator involving actual TCP code, and a model for the network running in the loopback driver of a Linux machine ([1]). We then study the performance of the effective capacity scheme and compare it with the instantaneous rate feedback scheme.

4.1 Instantaneous Rate Feedback

We have assumed that the modulating chain has two states. In the low state the link capacity is some fraction of the link capacity in the high state (where the full link rate is available). In the set of results, we will assume that this fraction is 0.5. Further, we will also assume that the mean time in each state is the same, i.e., the Markov chain is symmetric. Let us denote the mean time in each state by ψ[5]. In the Linux kernel implementation of our network simulator, the Markov chain can make transitions at most once every 30msec.

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 2 time units. The round-trip propagation delay Δ is taken to be 40 units. To give an example, if the link has a capacity of 155Mbps during its high rate state, and TCP packets have a size of 500 bytes each, then one time unit is 25.8μsec. The round trip propagation delay (Δ) is $40 \times 25.8\mu\text{sec} = 1.032\text{msec}$. Then, $\psi = 100$ means that changes in link bandwidth occur on an average, once every 103.2msec. If the link capacity is 2Mbps during the high rate period. Let the packet size be 1000bytes. Then, the delay corresponding to 40 time units is 160msec. $\psi = 100$ here corresponds to changes occurring once every 16 seconds. Thus the curves we present are normalized and can be used to read off numbers for many scenarios.[6]

In Figures 8 to 12, 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 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.

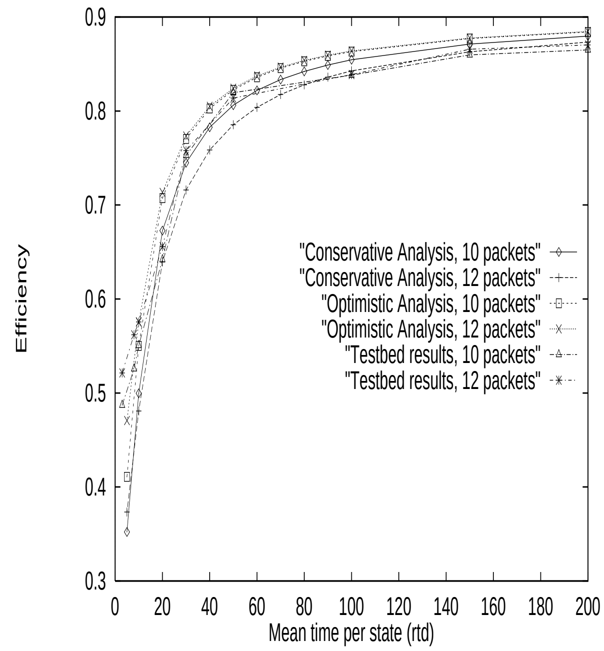

Figure 8: Throughput of TCP over ABR; analysis and simulation results with instataneous rate feedback scheme. The bottleneck link buffers = 10, 12 pkts.

Figure 8 shows the throughput of TCP over ABR with the instantaneous rate feedback scheme[7]. Here, we compare an optimistic analysis, a conservative one, and the testbed (i.e., simulation) results for different buffer sizes. We can see that, except for very small ψ, the analysis and the simulations match to within a few percent. Both the analyses are less than the observed throughputs by about 10% for small ψ. This can be explained if we note that in our model, we assume that packets arrive (leave) back to back to (from) the ABR source. When a rate change occurs at the bottleneck link, as the packets arrive back to back, and the source sends at twice the rate of the bottleneck link (in our example), for every two packets arriving to 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

[5]We express ψ in terms of the round trip *propagation* delay Δ (rtd). For example, if the rtd is 200msec, then $\psi = 0.5$ means that the mean time per state is 100msec.

[6]Note however that Δ is an absolute parameter in these curves, 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.

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

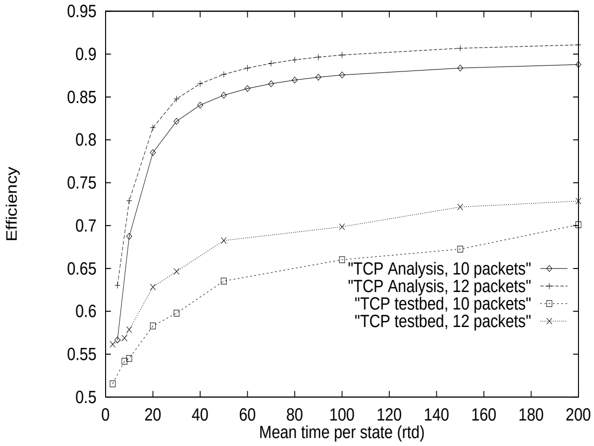

Figure 9: Throughput of TCP *without* ABR; analysis and simulation results. Bottleneck buffers $= 10, 12$ pkts.

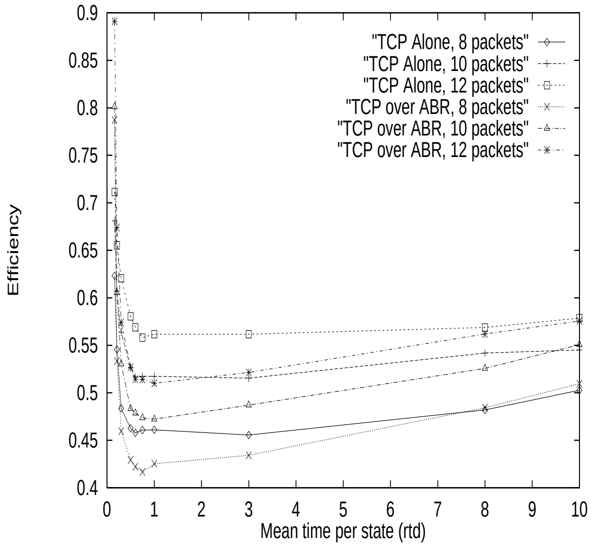

Figure 10: Simulation of TCP with and without ABR for small ψ; instantaneous rate feedback.

bottleneck link variations increase.

Figure 9 shows the throughput of TCP *without* ABR. We can see that the simulation results give a throughput of about 10 to 15% less than the analytical ones. This occurs due to two reasons. (1) We assumed in our analysis that no loss occurs in the slow-start phase. It has been shown in [6] that if the bottleneck link buffer is less than $\frac{1}{3}$ of the bandwidth-delay product (which corresponds to about 13 packets or 6500 byte buffer), loss will occur in the slow-start phase. (2) We optimistically compute the throughput of TCP by using an upper bound on the "reward" in the loss cycle.

We see from Figures 8 and 9 that whereas ABR makes TCP throughput insensitive to buffer size variations, with TCP alone there is a significant 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 there are no large drops in bottleneck link capacity.

From Figures 8 and 9, we can see that the performance of TCP improves by about 20% when ABR is employed for data transport, instantaneous bottleneck link capacity is fed back and the changes in link rate are slow.

The assumptions in our analysis render it inapplicable for very small ψ. Figure 10 compares the *simulation* results for TCP with and without ABR for various buffer sizes. These results are for ψ starting with $\psi = 10$ and going down to $\psi = 0.16$. We note that, for small ψ, the performance of TCP over end-to-end ABR is only slightly better or about 10% worse than TCP alone. In Section 4.2 we will show that, for small ψ, an effective capacity based feedback helps to improve the throughput of TCP over ABR to values greater than TCP alone.

We see from Figure 10 that when ψ becomes less than 1, the throughput of TCP increases. This can be explained by the fact that the rate mismatch occurs for an interval of time less than one round trip propagation delay. As a result, the buffer size required to handle the overload becomes less. As ψ becomes very small, each packet is sent at a different rate and hence, the ABR source effectively sends at the mean capacity. Then loss very rarely occurs as the buffers can handle almost all rate mismatches and, hence, the throughput increases.

4.2 Effective Capacity vs. Instantaneous Rate Feedback

In Figure 11, we compare the performances of the effective capacity and the instantaneous rate feedback schemes for ABR. Recall that the effective capacity 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 effective capacity scheme uses $M = 7$, i.e, we average over one round trip propagation delay[8] worth of samples. We also maintain a window of 8 rtd 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 can see from Figure 11 that for large ψ, the throughput with the effective capacity algorithm is worse than that of the instantaneous rate feedback scheme by about 3-4%. This is because of the con-

[8] A new sample is generated every 30msec. The rtd is 200msec in this example. Hence, M = 200/30 = 6.667 which we round up to 7.

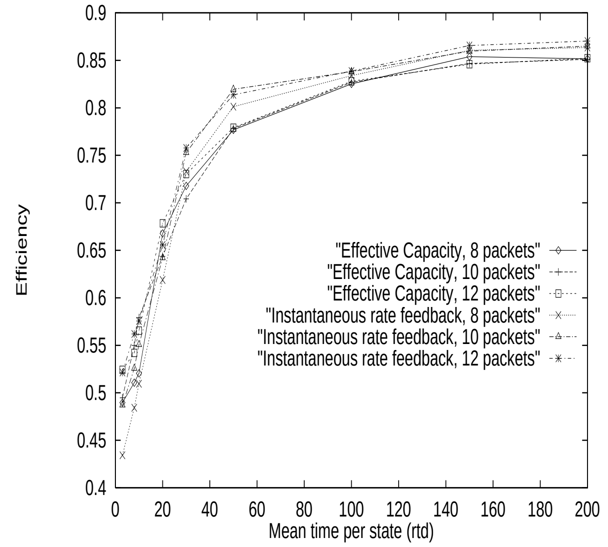

Figure 11: Simulation results; comparison of the Effective capacity and Instantaneous rate feedback schemes for TCP over ABR for various bottleneck link buffers (8-12 packets), for large ψ; $N = 49$ and $M = 7$.

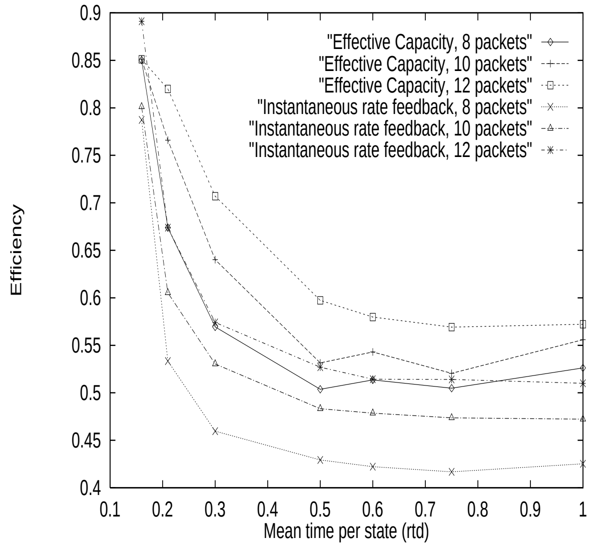

Figure 12: Simulation results; comparison of the effective capacity and instantaneous rate feedback schemes for TCP over ABR for various bottleneck link buffers (8-12 packets), for small ψ. $N = 49$ and $M = 7$.

servative nature of the effective capacity algorithm (it takes the minimum of the available capacity over several blocks of time in an interval).

However, we can see from Figure 12 that for small ψ, the effective capacity algorithm improves over the instantaneous feedback 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.

To summarize, from a comparison of Figures 9, 10, 11 and 12, we can see that for all values of ψ, the effective capacity scheme performs considerably better than TCP alone. Further, while being adaptive to ψ, the effective capacity scheme succeeds in keeping the TCP throughput better than the minimum link rate, which the instantaneous rate feedback scheme fails to do (for small ψ). Thus an MCR in the ABR connection may be used to guarantee a minimum TCP throughput.

Acknowledgement: The authors are grateful to Ajit Anvekar for helping develop the hybrid simulation environment, and for conducting many of the simulations.

References

[1] Ajit Anvekar and S. G. Sanjay, "A Hybrid TCP Simulator using Actual TCP code and the Virtual Loopback Device in the Linux Operating System", *ERNET Project Technical Report*, Indian Institute of Science, under preparation.

[2] *The ATM Forum Traffic Management Specification Version 4.0*, April 1996.

[3] Van Jacobson, "Congestion avoidance and control", *Proc. ACM Sigcomm'88*, August 1988.

[4] Shiv Kalyanaraman, Raj Jain, et. al. "Performance of TCP/IP over ABR Service on ATM Networks", *IEEE Globecom'96*.

[5] Anurag Kumar, "Comparative performance analysis of versions of TCP in a local network with a lossy link", *IEEE/ACM Transactions on Networking*, August 1998.

[6] T.V. Laksman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth delay products and random loss," *IEEE Transactions on Networking*, Vol. 5, No. 3, pp 336-350, June 1997.

[7] Teunis J. Ott and Neil Aggarwal, "TCP over ATM: ABR or UBR", manuscript.

[8] C. Pazos et. al. "Performance of TCP over ATM for various ABR Control Policies", manuscript.

[9] Allyn Romanov and Sally Floyd, "Dynamics of TCP Traffic over ATM Networks", *IEEE JSAC*, May 1995.

[10] S. G. Sanjay, "TCP over End-to-End ABR: A Study of TCP Performance with End-to-End Rate Control and Stochastic Available Capacity" *Master of Engg. Thesis*, Indian Institute of Science, Bangalore, India, January 1998.

[11] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", Internet RFC 2001.

[12] G. de Veciana and J. Walrand, "Effective Bandwidths: Call Admission, Traffic Policing and Filtering for ATM Networks", *Queueing Systems Theory and Applications (QUESTA)*, 1994.

[13] Ronald Wolff, *Stochastic Modeling and the Theory of Queues*, Prentice Hall, 1989.