Dynamics of an Explicit Rate Allocation Algorithm for Available Bit-Rate (ABR) Service in ATM Networks

Lampros Kalampoukas*, Anujan Varma*
and
K. K. Ramakrishnan†

UCSC-CRL-95-54
December 5, 1995

* Board of Studies in Computer Engineering
University of California, Santa Cruz
Santa Cruz, CA 95064

†AT&T Bell Laboratories
Murray Hill, NJ 07974

ABSTRACT

In this paper we study the performance of an explicit rate allocation algorithm for ATM Networks using the available bit-rate (ABR) class of service. We examine the behavior of ABR traffic with simple cell sources, and demonstrate that the allocation algorithm is fair and maintains network efficiency in a variety of network configurations. We also study the behavior of TCP/IP sources using ABR service in a network of switches employing the rate allocation algorithm; the results show substantial improvements in fairness and efficiency in comparison with the performance of TCP on an underlying datagram-based network. Even in a dynamic network configuration with frequent opening and closing of ABR connections, TCP connections were able to make sustained progress with a switch buffer size as small as 1 Kbyte, providing an average link utilization of approximately 60% of the attainable maximum.

We also study the performance of ABR traffic in the presence of higher-priority variable bit-rate (VBR) video traffic and show that the overall system achieves high utilization with modest queue sizes in the switches, and the ABR flows adapt to the available rate in a fairly short interval.

Scalability of the ABR service is an important criterion. We demonstrate the scalability of the rate allocation algorithm with respect to the number of connections by increasing the number of active connections with very long round-trip times by a factor of 13; this caused only a three times increase in the switch queue size, while still maintaining maximal link utilization.

Keywords: Explicit rate allocation, congestion control, TCP over ATM

This research is supported by the Advanced Research Projects Agency (ARPA) under Contract No. F19628-93-C-0175 and by the NSF Young Investigator Award No. MIP-9257103.

1 Introduction

Asynchronous Transfer Mode (ATM) networks are being developed with the intent of providing a single common technology to carry voice, video and data traffic. Networks based on ATM combine the flexibility of packet-switched networks with the service guarantees and predictability offered by circuit-switched networks.

Several service classes have been defined in the context of ATM networks. The *Constant-Bit-Rate* (CBR) and *Variable-Bit-Rate* (VBR) classes are intended to support applications with real-time service requirements. Applications that take advantage of these classes often have strict requirements for bandwidth, end-to-end delay, jitter and cell loss-rate. The CBR class is expected to support isochronous applications that need consistent availability of fixed bandwidth and jitter. The VBR class of service is intended for real-time applications whose transmission rate is variable, and the service exploits statistical multiplexing to achieve efficient utilization of network resources.

The *Available-Bit-Rate* (ABR) service class [1], defined to support delay tolerant best-effort applications, uses rate-based feedback mechanisms to allow them to adjust their transmission rates to make full utilization of the available bandwidth [2]. Compliant connections are also assured of a low loss rate, and if needed, a minimum bandwidth allocation. This class of service admits to the possibility of congestion, where the aggregate demand of the sources exceeds the capacity of the network resources. Thus, there is a need for a congestion management scheme in order to allocate the available bandwidth fairly among the connections sharing the network, while achieving efficiency and an acceptably low cell loss rate. The ATM Forum Traffic Management Committee is currently defining a rate-based congestion control framework to meet this objective.

The ATM Forum rate-control framework allows a number of options for the switches to signal their congestion state to the source. With the *explicit-rate marking* option that is the focus of our work here, the source of each connection periodically transmits a special *resource management* (RM) cell to probe the state of the network. Two components of the control algorithm are identified: (i) the behavior of the source and destination end systems, and (ii) the behavior of the network elements (switches). The congestion control function within the switches is responsible for identifying and signaling their congestion state to the source end-system. The source algorithm responds to the congestion state feedback information by adjusting the rate of transmission using an increase/decrease policy.

A source specifies the bandwidth demand and the current transmission rate of the connection in each transmitted RM cell. With the explicit rate scheme, switches communicate in the RM cell, the amount of instantaneous bandwidth it can allocate to each connection to the source of the connection. The goal of the allocation is to arrive at an efficient allocation that is also *max-min fair* [3]. Each switch on the path of the RM cell may modify the request based on the bandwidth it is able to allocate to the connection on its outbound link. On reaching its destination, the RM cell is returned to the source, which now sets its rate based on that allocated on the bottleneck link in the path of the connection.

Several rate allocation algorithms that operate in the explicit-rate marking mode have been proposed [4, 5, 6]. These approaches differ in terms of their execution time, implementation complexity, level of fairness in the allocation of the available bandwidth, responsiveness to network changes, convergence time, and stability properties.

In this work we study the dynamics and evaluate the performance of the rate allocation algorithm proposed in [5]. We consider the behavior of the rate allocation algorithm in both ATM-layer-generated ABR traffic and TCP-controlled ABR traffic. In the first case we show that, in the network configurations being analyzed, the algorithm converges to a steady state, allocates the available bandwidth fairly among competing connections, and has modest buffer requirements. We demonstrate its scaling capabilities by increasing the number of active connections by a factor of more than 10.

We also study the behavior of TCP-controlled ABR traffic. Several studies have shown that in certain configurations, in the absence of an ATM-layer congestion control mechanism, TCP performance degrades when operating over ATM networks [7, 8]. In this work we study the interaction of TCP built-in mechanisms for protection against congestion with our proposed explicit rate allocation scheme in conjunction with the basic rate increase/decrease algorithm of ATM Forum's source policy. We demonstrate that the use of an explicit rate allocation scheme enhances the fairness achieved for TCP/IP traffic compared to its performance in traditional datagram networks.

The paper is organized as follows: Section 2 reviews the rate-based congestion control framework and describes the proposed rate allocation algorithm. Section 3 provides a description of the simulation models used in this work. Section 4 discusses the dynamics of the described algorithm in a network configuration consisting of connections with widely-different round-trip times. We study the behavior of the network first with only ATM-layer generated traffic, and subsequently with ABR traffic that is flow-controlled by TCP. Section 5 studies the performance of ABR connections when mixed with VBR traffic. Section 6 analyzes the ability of the algorithm to adapt to changes in bandwidth by simulating TCP-generated traffic along with ABR connections that open and close randomly. Finally, Section 7 summarizes the results and proposes directions for future work.

2 Source and Switch Behavior

We describe in this section the source and destination algorithms used in our study (the essential parts are a close approximation to the basic algorithms in the ATM Forum framework [2]). We then describe our rate allocation algorithm used by the switches in the network.

2.1 Control Loop Operation

The source of a connection (VC) transmits cells at a rate allowed by the network, termed the *allowed cell rate* (ACR), until it receives new information in an RM cell that it had transmitted previously. The source sends an RM cell every $Nrm - 1$ data cells transmitted. This proportional transmission of RM cells is to ensure that the amount of overhead for RM cells is a constant, independent of the number of VCs in the network or their rates. The RM cell has the following fields:

1. The virtual circuit identifier to identify the connection it belongs to.
2. The amount of bandwidth requested, called *explicit rate* (ER).
3. The *current cell rate* (CCR) of the connection. On transmission of the RM cell, this field is set by the source to the value of the allowed cell rate (ACR) that the source is currently operating at. The ACR, in turn, is determined by the source increase/decrease policy based

on the ER-field of the RM cell returned by the network most recently. The CCR field is not modified by the network.

4. A bit indicating the direction of the RM cell. This is necessary to distinguish an RM cell transmitted by the source of a connection from one returned by the destination to the source.

Note that ER is the bandwidth requested by the connection during the current epoch, while the CCR field reflects the current rate the source is allowed to transmit at.

The switches use the *explicit rate* option. Whenever an RM cell from connection is received at a given switch, the switch determines the allocation for the VC based on the bandwidth being requested in the ER field. If the maximum bandwidth that can be allocated is less than the value in the ER field, then the ER field is updated to reflect the new maximum possible allocation on the path of connection so far.

When the RM cell returns back to the source, the transmission rate (allowed cell rate, ACR) is updated based on the value indicated by the ER field in the returned RM cell. The value in the ER field reflects the bandwidth that can be allocated at the bottleneck link in the path of the connection. If the current transmission rate is above the value in the ER field, the source immediately reduces its rate to this value. However, if the current rate is less than the returned ER value, the transmission rate ACR is increased gradually by increasing a constant amount ($Nrm \times AIR$) to the current ACR. In addition, the source is never allowed to exceed the rate specified by the ER field of the returned cell. Thus, if $r(t)$ is the transmission rate of the source the instant just before the arrival of an RM cell, and $r(t+)$ the rate after the update, then

$$r(t+) = \min(r(t) + Nrm \cdot AIR, ER).$$

An analytical model for the rate increase process at the source can be found in [9]. In addition to the basic increase/decrease algorithm, the source policy contains mechanisms for recovering unused bandwidth from idle connections, making the system more robust to lost RM cells, etc. Since our interest in this paper is to study the basic behavior of the rate allocation algorithm, we focus on the primary control-loop operation, and ignore the issues of boundary cases (very low rate sources, substantial loss of RM cells, etc.) in order to simplify the evaluation.

2.2 Rate Allocation Algorithm

The explicit rate option assumes the existence of an algorithm within the switch that allocates the available bandwidth on each outgoing link among the connections sharing it. Such algorithms for rate allocation in packet-switched networks have been described by Charny [4], Kalampoukas, et al. [5], Jain [6], and Siu, et al. [10]. In this paper we consider the rate allocation algorithm described by Kalampoukas, et al. [5]. The main components of the algorithm are described briefly. A more extensive discussion of the algorithm can be found in [5].

The following definitions and notations are used in our description: Consider any switch in the path of a connection. Let $S(t)$ be the set of active connections sharing the outbound link of this switch at time t. At any time, connections in $S(t)$ can be in one of two states — *bottlenecked* or *satisfied*. We designate a connection as satisfied if, at the most recent update of its allocation, its request was completely satisfied. The state of the connection is marked as *bottlenecked* if the allocation it received most recently at the switch was less than its request. We denote the set of satisfied connections at time t as $S_u(t)$ and the set of bottlenecked connections by $S_b(t)$. Let $N(t)$,

$N_u(t)$ and $N_b(t)$ denote the sizes of the sets $S(t)$, $S_u(t)$, and $S_b(t)$, respectively. We use $B(t)$ to denote the total bandwidth available on the outbound link to allocate to ABR traffic, and $ER_j(t)$ the value in the ER field of the most recent RM cell received in the forward direction from connection j, and $CCR_j(t)$ be the corresponding value in the CCR field. The current request of the connection, denoted by $\rho_j(t)$, is taken as the minimum of the ER and CCR values in the most recent RM cell received. $A_j(t)$ represents the corresponding allocation received by connection j during its most recent update.

The goal of our rate allocation algorithm is to make available to each bottlenecked connection at time t, a maximum bandwidth equal to

$$A_{max}(t) = \frac{\text{Total bandwidth available to bottlenecked connections}}{\text{Number of bottlenecked connections}}. \tag{2.1}$$

The total bandwidth available to bottlenecked connections is the bandwidth left over after allocating to satisfied connections. Therefore, the above equation becomes

$$A_{max}(t) = \frac{B(t) - \sum_{i \in S_u(t)} A_i(t)}{N_b(t)} \tag{2.2}$$

On receipt of an RM cell from connection j, say at time t, the first step in the algorithm is to determine the new state of that connection. This step is performed as follows: If the connection j is currently marked as bottlenecked, the algorithm checks whether its state needs to be changed to satisfied. This is accomplished by means of the following calculations: The maximum bandwidth available to connection j, that is $A_{max}(t)$ is determined from Eq. (2.2) above using the current values of the parameters in that equation. If the $A_{max}(t)$ so obtained is larger than the current request $\rho_j(t)$ of connection j, then its state is changed to satisfied. On the other hand, if the connection j was in satisfied state when the RM cell is received from it, then the algorithm checks if the state of the connection needs to be changed to bottlenecked, given the current values of the parameters. This checking is accomplished by temporarily setting the state of connection j as bottlenecked and going through a computation similar to that of Eq. (2.2) to determine the maximum bandwidth that would be allocated to it. The following equation is used to determine A_{max} in this case:

$$A_{max}(t) = \frac{B(t) - \sum_{i \in S_u(t)} A_i(t) + A_j(t)}{N_b(t) + 1}. \tag{2.3}$$

The computations in both equations (2.2) and (2.3) can be performed without searching the state of each connection by maintaining the residual bandwidth available for allocation to bottlenecked connections, given by

$$B_b(t) = B(t) - \sum_{i \in S_u(t)} A_i(t). \tag{2.4}$$

That is, the current bandwidth that can be allocated to bottlenecked connections is the total available bandwidth minus the total bandwidth currently allocated to connections in satisfied state. Instead of $B_b(t)$, in our algorithm we maintain the quantity

$$B_f(t) = B(t) - \sum_{i \in S_u(t)} A_i(t) - \sum_{i \in S_b(t)} \frac{B(t)}{N(t)}. \tag{2.5}$$

We refer to $B_f(t)$ as the "free bandwidth." Note that $B(t)/N(t)$ is the *equal share* of a connection and is the minimum bandwidth it is entitled to receive under any fair allocation. We denote $B(t)/N(t)$ by $B_{eq}(t)$, the equal share. Since $(N_b(t) + N_u(t)) \cdot B_{eq}(t) = B(t)$, Eq. (2.5) can also be written as

$$B_f(t) = N_u(t)B_{eq}(t) - \sum_{i \in S_u(t)} A_i(t). \tag{2.6}$$

Thus, the free bandwidth can be seen as the bandwidth available as a result of the satisfied connections not requesting their equal share. Using $B_f(t)$ to compute the allocation instead of the actual available bandwidth has an advantage: When a new connection is opened, $N(t)$ increases by one even before the connection sends its first RM cell. This has the effect of reducing $B_f(t)$ in Eq. (2.6), thus reducing the allocation to existing connections. This helps to reduce congestion during the transient period when the algorithm is converging to a new max-min fair allocation.

Since we use $B_f(t)$ instead of $B(t)$ in the algorithm, we re-write Eq. (2.2), used to check state changes for bottlenecked connection, as follows:

$$A_{max}(t) = B_{eq}(t) + \frac{B_f(t)}{N_b(t)}. \tag{2.7}$$

Similarly, we re-write Eq. (2.3), used to check state changes for a satisfied connection, as follows:

$$A_{max}(t) = B_{eq}(t) + \frac{B_f(t) + A_j(t) - B_{eq}(t)}{N_b(t) + 1}. \tag{2.8}$$

If we set

$$A'_j(t) = \begin{cases} A_j(t), & \text{for all } j \in S_u(t); \\ B_{eq}(t), & \text{for all } j \in S_b(t); \end{cases}$$

then we can combine Eq. (2.7) and (2.8) into a single equation:

$$A_{max}(t) = B_{eq}(t) + \frac{B_f(t) + A'_j(t) - B_{eq}(t)}{N_b(t_1) + w}, \tag{2.9}$$

where $w = 0$ if $j \in S_b(t)$ and $w = 1$ if $j \in S_u(t)$.

Thus, Eq. (2.9) is the actual equation used by the algorithm in its first step to detect state changes of connections. Note that, if a state change is found to occur, the parameters $N_b(t)$ and $N_u(t)$ must be updated to reflect the new state.

Once the state of connection j has been updated, the second step of the algorithm is to update the actual allocation maintained for the connection j. The new allocation $A_j(t+)$ for connection j is computed based on the parameters of the RM cell received at t as

$$A_j(t+) = \min(A_{max}(t), \rho_j(t), CCR_j(t)). \tag{2.10}$$

That is, the current allocation of the connection is chosen as the minimum among A_{max} and the values in the CCR and ER fields of the RM cell. After updating $A_j(t)$, $B_f(t)$ is updated as

$$B_f(t+) = B_f(t) + A'_j(t) - A'_j(t_1+). \tag{2.11}$$

In addition to recording the local allocation, the algorithm also updates the ER field of the RM cell before transmitting it through the outgoing link, when necessary. This update is performed as follows: If the computed $A_{max}(t)$ is less than the request $\rho_j(t)$, the ER field is set to $A_{max}(t)$. Otherwise the ER field is not modified. Thus, when the RM cell reaches the destination, the ER field reflects the bandwidth available at the bottleneck link along the path of the connection.

A problem arises when the requests of all the connections sharing the outgoing link have been satisfied completely. If, for example, one of the connections were to increase its bandwidth request in such a way that the new request exceeds the maximum bandwidth that can be allocated, the connection must now be marked bottlenecked. However, it is possible that, at that time, there is another connection in the satisfied state receiving a larger allocation than that assigned to the bottlenecked connection. This situation can be prevented by finding the connection receiving the largest allocation and marking its state as bottlenecked even if it is receiving all its requested bandwidth. This prevents a situation where a connection in satisfied state is actually receiving more bandwidth than another in the bottlenecked state. Thus, the algorithm maintains the index of the connection receiving the largest allocation in a separate variable and updates it on the receipt of every RM cell.

Equations (2.9) and (2.11) form the core of our algorithm. Both updates can be done in $O(1)$ time by maintaining current values of the following parameters, in addition to the current state of each connection.

1. The parameter $B_f(t)$ representing the free bandwidth
2. The value A_i' corresponding to the current allocation for each connection i
3. The number of bottlenecked connections and the total number of active connections, $N_b(t)$ and $N(t)$, respectively
4. The available bandwidth for ABR connections, $B(t)$, for computation of the equal share $B_{eq}(t)$.

The pseudocode in Figure 2.1 summarizes the algorithm that is invoked on the receipt of each RM cell traveling in the forward direction. In addition, the variables maintained by the algorithm must be updated when a new connection is opened, an existing connection closes, or when the available bandwidth $B(t)$ changes.

Figure 2.2 illustrates the working of the rate allocation algorithm. Assume that flows 1, 2, and 3 are in steady state, transmitting at rates of 40, 35 and 20 Mbits/sec, respectively. The link capacities are 100 Mbits/sec for the switches shown. Assume that Flow 1 transmits an RM cell with ER = 100 and CCR = 40. The rate allocation algorithm in Switch 1 determines the maximum allocation A_{max} as 65 and modifies the ER value in the RM cell to 65. However, the A_{max} value computed by the second switch is only 45, and hence the ER value is further marked down by Switch 2 to 45. When the RM cell returns to the source of the flow, the source will gradually increase its rate from 40 to 45. Consequently, subsequent RM cells transmitted by the source will have progressively increasing CCR values; as the CCR increases, so does the local allocations at the switches, until steady state is again reached with both switches allocating 45 Mbits/sec to the flow. Observe that the switches compute local allocations based on the minimum of ER and CCR values received. Computing local allocations based on ER values alone can lead to severe under-utilization of the link capacity.

3 Simulation Models

In this section, we provide an overview of the simulation models used in the paper. More detailed description of a specific topology used in a simulation will be given in the corresponding sections describing the simulation results.

The links in the network are full-duplex with a capacity of 155 Mbits/sec each, unless otherwise specified. The switches are nonblocking, output-buffered crossbars. There is one queue per output port for ABR traffic and its scheduling policy is FIFO, with each output queue being shared by

```
    /* Pseudocode invoked on receipt of forward RM cell from connection i */
1.  i ← cell(VC)                          /* get the VC number */
2.  ER_i ← cell(ER); CCR_i ← cell(CCR)    /* read ER and CCR fields of RM cell */
3.  ρ_i ← min(ER_i, CCR_i)                /* ρ_i is the request of connection i*/

    /* Compute maximum local allocation A_max.*/
4.  B_eq ← B/N                            /* compute equal share */
5.  if state_i = bottlenecked then
6.      A_max ← B_eq + B_f/N_b
7.  else
8.      A_max ← B_eq + (B_f + A'_i − B_eq)/(N_b + 1)

    /* Determine new state of connection */
9.  if (A_max < ρ_i) and (state_i = satisfied) then
10.     state_i ← bottlenecked; N_b ← N_b + 1
11. if (A_max > ρ_i) and (state_i = bottlenecked) then
12.     state_i ← satisfied; N_b ← N_b − 1

    /* Compute the local allocation A_i */
13. if (state_i = satisfied) then A_i ← ρ_i
14. else A_i ← A_max

    /* Update ER field of RM cell */
15. ER(cell) ← min(A_max, ER_i)

    /* Maintain state of connection with the largest allocation as bottlenecked */
16. if (A_i > MaxAllocation) then
                /* mark this VC as the one with the largest allocation */
17.     MaxVC ← i; MaxAllocation ← A_i
18.     if (state_i = satisfied) then     /* mark it as bottlenecked */
19.         state_i ← bottlenecked; N_b ← N_b + 1
20. if (MaxVC = i) and (A_i < MaxAllocation)
21.     /* This is the VC with the largest allocation and its allocation went down */
22.     MaxAllocation ← A_i;              /* update largest allocation */

    /* Update local allocation maintained in the switch, A'_i */
23. A_old ← A'_i                          /* save old allocation */
24. if (state_i = satisfied) then A'_i ← A_i
25. else A'_i ← B_eq

26. /* Update free bandwidth B_f */
27. B_f ← B_f + A_old − A'_i
28. forward(cell)                         /* forward RM cell to the next switch */
```

Figure 2.1: Pseudocode for the rate allocation algorithm.

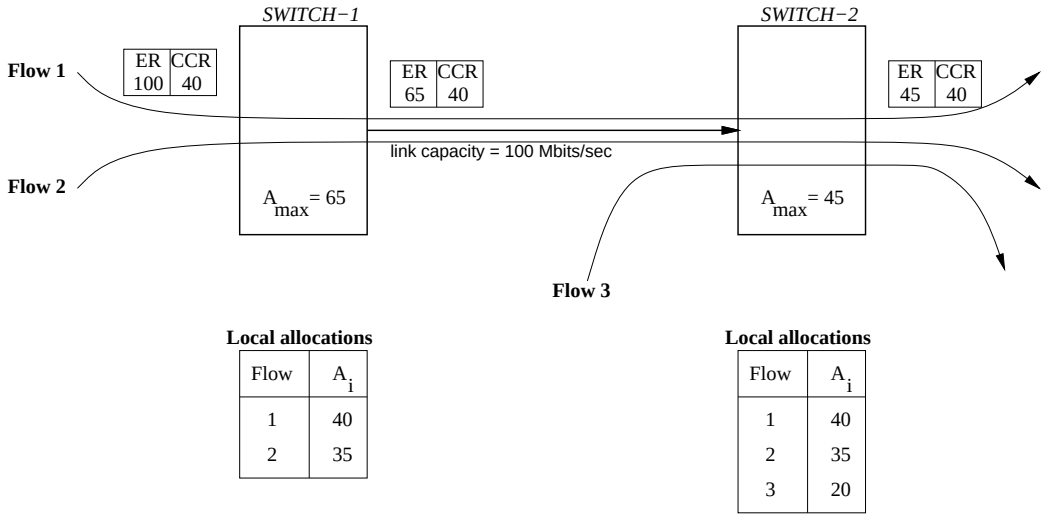

Figure 2.2: Example for the rate allocation algorithm.

all the virtual circuits (VCs) that are sharing the outgoing link. We assume that all the switches support the explicit rate allocation algorithm of Section 2 and the sources follow the source policy described. The parameters for the source-end systems are set as follows:

- Nrm = 32 cells.
- ICR = PCR/50 ≈7300 cells/sec. This is the initial cell rate of the source, upon start up, until an RM cell is returned with a new rate.
- AIR = 180 cells/sec per cell transmitted (about 5760 cells/sec maximum rate increase every Nrm cells).
- PCR = 365,566 cells/sec. The peak rate for all VCs (link rate).

An important observation we make is that the same set of parameters are used for both WAN and LAN configurations which have widely different characteristics.

When simulating TCP over ATM, we use the ATM Adaptation Layer Type 5 (AAL 5) [11]. AAL 5 performs segmentation and re-assembly between IP packets and ATM cells. Each IP packet is extended by eight bytes to accommodate the AAL header. Thus, the number of ATM cells produced by the original IP datagram is given by $\left\lceil \frac{\text{IP packet size}+8}{48} \right\rceil$.

The model for TCP used in the simulations is based on the TCP-Reno version. It supports the congestion control mechanism described by Jacobson [12], exponential back-off, enhanced round-trip (RTT) estimation based on both the mean and the variance of the measured RTT, and the *fast retransmit and fast recovery* mechanisms. However, some adjustments had to be made to the TCP timers; since the RTT values in some of our simulations are of the order of just a few milliseconds, the coarse-grain timers used in Unix TCP implementations (typically with a granularity of 500 ms) would make the comparison of the schemes difficult. To avoid the masking of the performance differences of TCP (which we believe will eventually be the case when we have finer-grained timers) due to coarse-grain timers, we used double-precision floating-point arithmetic in the RTT estimation algorithm. Therefore, both the RTT measurements and the timeout delays are represented by double-precision floating-point numbers. The TCP segment size was set to 9180 bytes in all the simulations.

We focus on the following performance measures: 1) The ACR, or transmission rate, at the source. In all the graphs we show the ACR value on every change, without any filtering of the

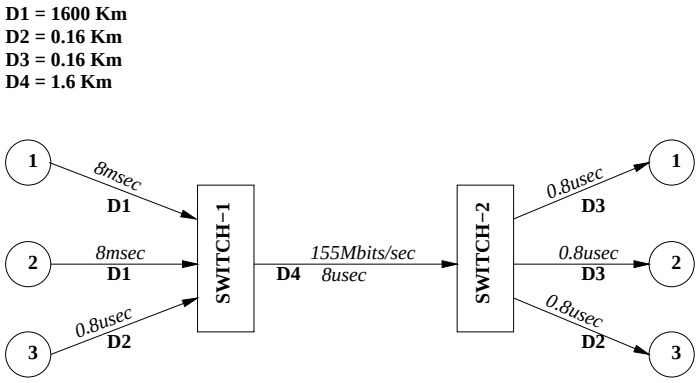

Figure 4.1: Configuration R1.

collected information. 2) The utilization of the links. We present the utilization averaged over 5 millisecond intervals. 3) The queue length at the switch for individual links. We present in the plots the queue length, which is the maximum value observed during a 5 millisecond interval. 4) In the case of TCP traffic, we also show the TCP sequence number growth for each individual TCP connection and the corresponding window size in bytes, where appropriate.

4 Interactions of Connections with Unequal Feedback Delays

In this section we consider the dynamics of the rate allocation algorithm in a network configuration where connections with widely different feedback delays interact. We first study the behavior of the algorithm with ABR traffic from cell sources, and subsequently characterize its behavior with TCP-generated ABR traffic.

4.1 System Behavior with ATM-Layer-Generated ABR Traffic

We begin the evaluation of our rate-allocation algorithm with a simple configuration, referred to from now on as the *R1 configuration* , shown in Figure 4.1. It consists of three connections which open simultaneously and request peak bandwidth. Data flows from the end-systems on the left to the ones on the right. Connections are set up between corresponding end-systems, identified by the same index within circles. The link propagation delays and capacities are as shown in the figure. The reason we find the configuration R1 interesting is because of the large difference (three orders of magnitude) between the round-trip times of the different connections. The round-trip delay of connection 3 (to be referred from now on as the *short connection*) is 11.2 μseconds while that of connections 1 and 2 (from now on to be referred as *long connections*) is 16.076 milliseconds. We expect D4 to be the bottleneck link in the configuration. All the sources follow the source policy outlined in Section 2. The sources are assumed to be greedy, that is, they always set the ER field of every transmitted RM cell to the peak link capacity of 155 Mbits/sec.

Because of the large difference in the feedback delay between the long and short connections, and the small initial rates of the flows (ICR), we expect that the short connection (connection 3) will quickly ramp up to acquire a larger than fair share of the bottleneck link bandwidth. This initial start-up transient is clearly seen in Figure 4.2 which shows the exact evolution of the transmission rate at the source (ACR) for the three VCs. However, as time passes, the returned RM cells for

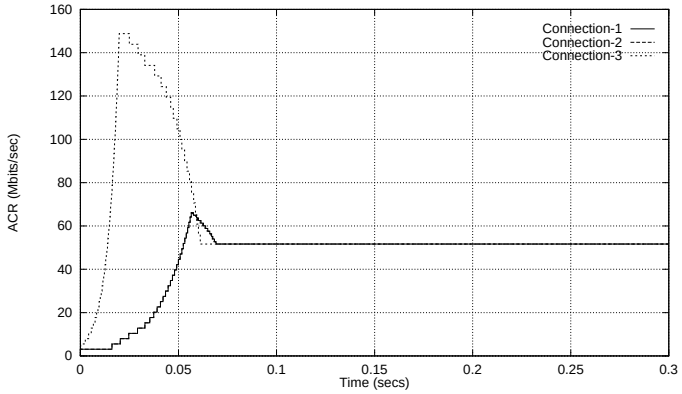

Figure 4.2: Transmission rates (ACRs) of the three connections in the R1 configuration.

the long connections allow them to acquire their fair share, and the short connection releases the bandwidth it acquired during the slow start-up of the long connections. Eventually, all the rates converge to their final allocation of one-third of the link bandwidth (about 51.5 Mbits/sec per connection).

The behavior of the rate decrease for the short connection during the start-up phase is gradual, rather than in a small number of discrete steps (these steps would have been, in an idealized system with zero latency and an instantaneous increase at the sources to the ER value returned in an RM cell, going from link rate, to half the link rate when an RM cell is seen from one long connection and finally to a third of the link rate when the RM cell from the other long connection is seen). The reason for the slower than ideal decrease in the ACR for the short connection is because of the source increase policy and the allocation based on CCR. During the transient buildup of transmission rate for the long connections, the CCR transmitted in RM cell is smaller than the ER value; since the allocation at the switches is based on the CCR value (which is necessary to avoid under-utilization), the result is an allocation that is smaller than the ER value returned previously by the switch. As the long connections gradually increase their rates, the CCR values in their RM cells increase, causing a corresponding reduction in the ER values returned to the short connection.

It is important to note the small overshoot in the transmission rates of the long connections before convergence is finally reached. This overshoot is a direct result of the allocation based on CCR values of the connections, and can be explained as follows: Assume that each of the two long connections transmits at time t_1 a forward RM cell with its CCR field containing transmission rates $r_1(t_1)$ and $r_2(t_1)$, respectively. These RM cells arrive at Switch 1 at time $t_2 = t_1 + 8$ msecs. Let $A_3(t_2)$ be the current allocation for the short connection in Switch 1 at that time. A computation for rate allocation is performed at time t_2 for each of the long connections. Assume that the RM cell from Connection 1 is the first seen by the switch. The updated ER value in its RM cell will now be $B - (A_3(t_2) + r_2(t_1))$, where B is the link capacity. It is easy to observe that, if $(A_3(t_2) + r_2(t_1))$ is less than $2B_{eq}$, the ER value signaled to Connection 1 can be larger than B_{eq}. The same ER value is also signaled to Connection 2 when its RM cell is processed. When these RM cells reach the sources of the long connections, the sources attempt to gradually increase their rates to the new ER values signaled, resulting in the rates exceeding the fair value B_{eq} temporarily. This overshoot is soon corrected when the increased CCRs of the long connections reach the switch, which clamps their allocations to B_{eq}. However, because of the long feedback delay, convergence to B_{eq} occurs slowly at the sources of the long connections.

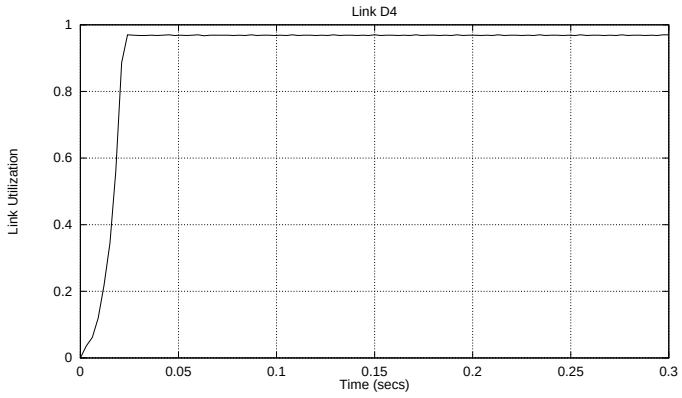

Figure 4.3: Total utilization of link D4 in the R1 configuration with three connections.

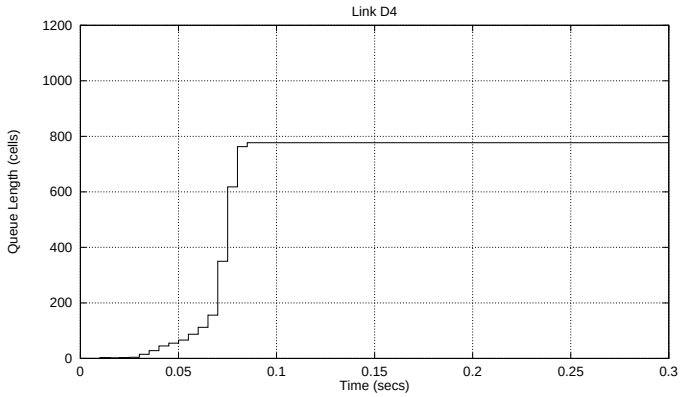

Figure 4.4: Queue length at the bottleneck link in switch-1.

Notice here that the transient bandwidth over-allocation may be reduced if we update the ER field of the RM cells going in the backward direction also. In that case, if the value in the ER field carried by an RM cell is larger than the most recent value of A_{max}, we update the ER field with the new A_{max}. This might improve the convergence of the rate allocation process in the general case; however, the modification would have little effect in this specific example because the congested switch is very close to the destination.

In Figure 4.2, the transmission rates of the sources converge to their final values within 70 msecs. Considering the 16 msecs round-trip delay of the long connections, this is quite reasonable, After the rate allocation process is completed, the transmission rates remain constant and the overall behavior is stable as long as the network state remains unchanged (that is, no connections open or close, and the bandwidth available to ABR traffic remains constant). We will later study the behavior of the scheme in a more dynamic setting where connections open and close frequently.

Although the feedback delay affects responsiveness of long connections to network changes, the utilization of the congested link is less affected. This is because of the short connection is able to utilize the excess bandwidth of the link while the long connections are gradually increasing their rates. Figure 4.3 shows the utilization of the link D4. Note that the utilization reaches its maximum value within approximately 25 msecs and remains constant thereafter. The maximum link utilization reached is about 97%, the theoretical maximum achievable after accounting for the overhead due to RM cells.

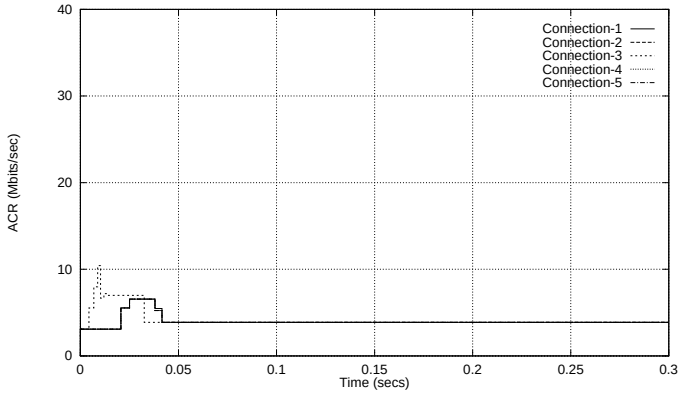

Figure 4.5: Transmission rates for the five sets of connections in the modified R1 configuration with 40 connections.

The transient bandwidth over-allocation causes a queue build-up in switch-1, as illustrated by Figure 4.4. The length of the queue is a function of the amount of the bandwidth over-allocation and the duration of the transient phase. As shown in Figure 4.4, however, even in a network with a round-trip delay of the order of 16 msecs, the built-up queue was relatively small, approximately 780 cells (approx. 40 Kbytes). Once built up, the queue size remains steady until a change in network state occurs, because our target link utilization is set at 100 %.

To examine how the allocation algorithm scales with the number of connections, especially with respect to its convergence time, the required amount of buffering and the bottleneck link utilization, we slightly extended configuration R1. The new configuration (the figure is omitted due to space constraints) contains 5 nodes on each side. Every source node on the left now originates eight connections, thus increasing the total number of VCs to 40. The round-trip delay for the two new sources was chosen identical to that of the long connections in Figure 4.1. Thus, the configuration consists of 32 connections with 16 msecs round-trip delay (long connections), and 8 connections with very small (about 11.2 μsecs) round-trip delay (short connections).

The transmission rates (ACR) for the connections in this modified configuration are shown in Figure 4.5. For simplicity, we have plotted only the transmission rate for a single connection in the set of connections originating at each source node. The behavior of the transmission rates is almost identical to that of the original R-1 configuration with three connections. However, convergence of the transmission rate to the final values is faster than before, taking only about 45 msecs (compared to 70 msecs with 3 connections). Therefore, the convergence time scales well with increasing number of connections.

As before, setting the target link utilization at 100% can lead to queue buildup at the bottleneck switch during the transient phase. However, the behavior of the queue size at the bottleneck link in switch 1, shown in Figure 4.6, indicates that the queue at the bottleneck link does not grow rapidly with the number of active connections. An increase by a factor of about 13 for the number of connections results in increasing the queue size by only a factor of 3. When we increased the number of short connections rather than the number of long connections in the R1 configuration, the increase in queue size was even smaller, about 30%. This is in spite of choosing a fairly high initial transmission rate of about 3.1 Mbits/secs in comparison to the final rate of 3.8 Mbits/sec for each connection. A relatively large initial rate, with multiple increases by the short connections, results in the total allocation going beyond the capacity for a short period, but the connections eventually

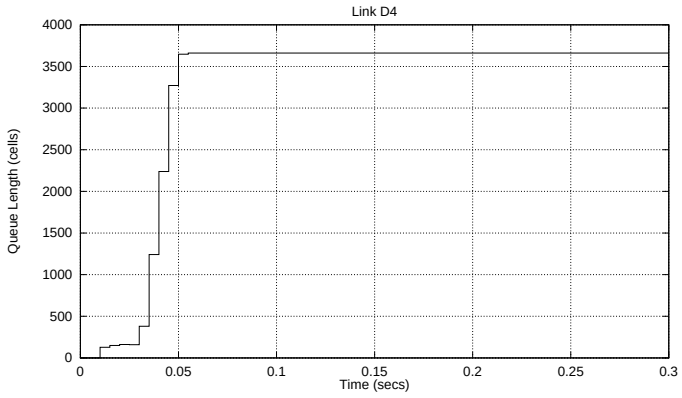

Figure 4.6: Queue length for the bottleneck link in switch-1 in the modified R1 configuration with 40 connections .

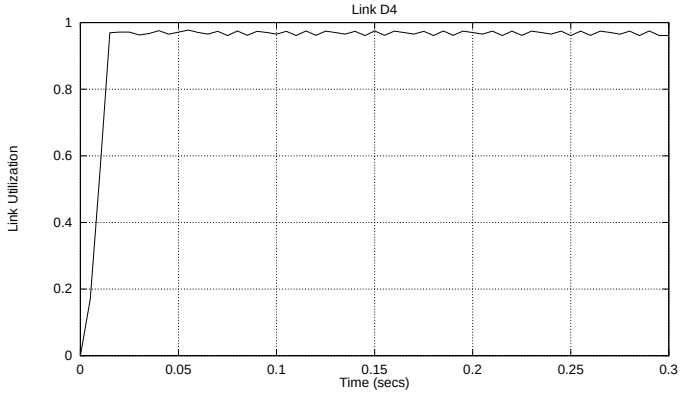

Figure 4.7: Utilization of link D4 in the modified R1 configuration with 40 connection.

adjust to their fair share. If desired, the queue sizes can be maintained close to a set target by an adaptive scheme that varies the ER values signaled to the sources based on the target queue size and the control delays, using an approach similar to that proposed by Mishra and Kanakia [13].

The utilization of the bottleneck link achieves its maximum value somewhat quicker when the number of connections is increased. As shown in Figure 4.7, the utilization of the bottleneck link achieves its maximum in about 15 msecs, compared to 25 msecs for the same configuration with three connections. This is due to a combination of the relatively large aggregate initial rate for all the long connections and the fast ramp-up by the short connections due to their small feedback delay. The plot in Figure 4.7 is based on measuring the link utilization at 5 millisecond intervals. Note that, with a large number of low-rate sources, the difference in the number of RM cells seen over the measurement intervals can be significant, contributing to the "wavy" nature of the plot.

From our observations of the limited increase in the queue size, fast convergence and maintenance of high utilization, we believe that the allocation algorithm scales well with the number of connections.

4.2 Dynamics of TCP Traffic over ABR Service in a Network Configuration with Unequal Feedback Delays

The ABR traffic will not, in general, consist of ATM-layer-generated data only. Many applications use a transport protocol to provide reliable end-to-end transmission of data. Since TCP is currently the most widely used reliable transport protocol, ATM will likely be used widely as the datalink layer for the TCP/IP Internet as a means of evolving from the current infrastructure. Using TCP over the ABR service has the advantage of allowing existing TCP applications to be used in an ATM network, and has the potential for utilizing the bandwidth efficiently while providing reliable data delivery. In this subsection we study how the rate allocation algorithm at the ATM layer influences the behavior of TCP.

Of particular interest is to study how the TCP congestion control mechanisms affect the behavior of the rate allocation algorithm. The TCP congestion control algorithm is based on end-to-end windows and consists of several components. Key components are the *slow-start* algorithm, a congestion-avoidance mechanism, and an algorithm to estimate round-trip delays. The slow-start algorithm is used to increase the window-size at start-up by initializing it to one segment and doubling it once every round-trip time. It is also used to perform congestion recovery on packet losses. The function of the congestion avoidance mechanism is to probe for additional available bandwidth in the network by gradually increasing the window at the rate of one segment per round-trip time. The delay-estimation algorithm attempts to maintain a good estimate of the round-trip delay which is used as a basis to set the retransmission timers. The TCP Reno Version, introduced in 1990, added the *fast retransmit and fast recovery* algorithm to avoid performing slow-start when the level of congestion in the network is not severe to warrant congestion recovery by slow-start.

For this study, we use the same R1 configuration considered in the previous subsection with two long connections and one short connection. The only difference is that the traffic of each ABR connection is now flow-controlled by TCP. To avoid packet losses at the sources from affecting our results, we have assumed a buffer size at the source IP layer large enough to prevent losses.

In addition to studying the initial start-up phase and the steady-state behavior of the connections, we also examine the dynamics of the connections when a packet loss occurs. This is achieved by dropping a cell from a TCP segment from connection 1 at time $t = 0.5$ seconds. In this case the AAL5 layer at the receiving end will detect a corrupted packet and discard all the remaining cells from that packet. The segment loss is later detected by the TCP source, which then retransmits the segment.

Figure 4.8 shows the ACR values at the sources of the three connections. The source rate behavior during the start-up phase is similar to that with cell sources, except for the more abrupt increase and decrease steps. This change in behavior is due to the TCP slow-start algorithm which increases the window size by doubling it every round-trip time. This produces intervals of time during which sources have no data to transmit. Since the source rate is allowed to increase only on the receipt of an RM cell, the idle intervals produce breaks in the rate increase process. However, the ACRs eventually converge to the fair values and remain steady. Since the rate allocation algorithm maintains a steady allocation regardless of the burstiness of the sources, events at the TCP layer do not influence the allocated rates of connections once convergence has been reached. That is, once the rate allocation algorithm converges, the behavior is similar to each connection operating over a dedicated link with no interference from other connection.

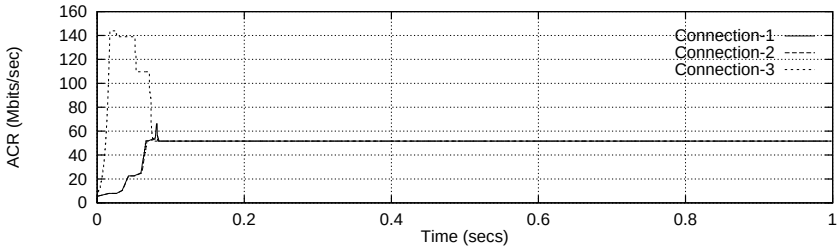

Figure 4.8: ACRs for the three connections in R1 configurations with TCP-generated traffic (in Mbits/second).

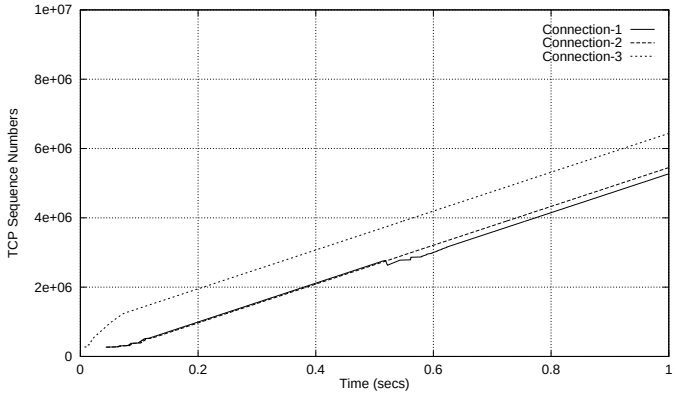

Figure 4.9: TCP sequence numbers for the three connections.

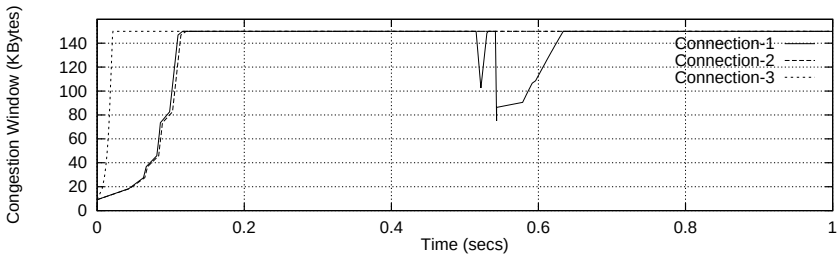

Figure 4.10: TCP congestion windows for all three connections.

Figure 4.9 shows the increase in the sequence numbers of TCP segments transmitted by the three connections as a function of time. The plot for the short connection has a substantially higher slope during the slow-start phase, owing to its much smaller round-trip delay. However, in steady state, the rate of increase for all the connections is identical, demonstrating the effectiveness of the rate allocation scheme in providing fairness among connections with widely different round-trip delays. This is considerably different from the behavior of TCP in a datagram-based network, where it has been shown to favor short connections [14].

Figure 4.10 shows the behavior of the congestion windows for the three connections. Once steady-state has been reached, the congestion windows remain at their maximum set value of 150 Kbytes until 0.5 second, when a packet loss is simulated by discarding a single ATM cell from connection 1.

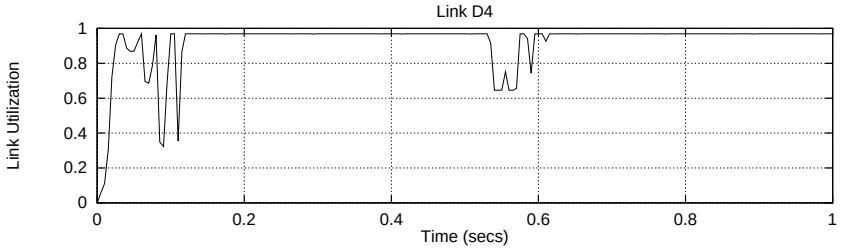

Figure 4.11: Utilization of link D4.

The source TCP of connection 1 soon detects the segment loss and enters the fast-retransmit and fast-recovery phase after the receipt of three duplicate acknowledgements. This results in its congestion window being set to half the size before the packet loss occurred and increased slowly at the rate of one segment per round-trip time. The cell loss, however, has no effect on the other TCP connections owing to the isolation provided by the rate allocation algorithm.

The overall utilization of the bottleneck link D4 is shown in Figure 4.11. In this case, the maximum utilization is reached within 200 ms after start-up. This long transient phase is because of the delays introduced by the TCP slow-start process. The oscillations in the link utilization during this transient period are due to the bursty behavior of the TCP sources during slow-start. When all connections have competed their slow-start phases, the utilization remains steady until the simulated packet loss at 0.5 second. Since the ATM source policy we have implemented does not incorporate any provisions for recovering bandwidth from idle sources, the unused bandwidth of connection 1 during its recovery phase is not made available to other connections, causing a temporary drop in the link utilization. However, the utilization is soon restored to its original value when connection 1 recovers fully from the packet loss.

In summary, the simulation results in this subsection show that substantial improvements in fairness and efficiency in the operation of TCP can be obtained by the use of ABR service in conjunction with our rate allocation algorithm. It should be pointed out, however, that the results here assume the ATM rate control loop extending all the way to the end-systems, with no packet losses at the end-systems. In practice, the control loop may not extend to the end-system, resulting in bottlenecks at the boundaries. In addition, the TCP congestion avoidance algorithm is designed to increase the window until a packet loss occurs; in a rate-controlled ATM network, this may cause the congestion window to grow until its maximum set limit, or until a packet loss occurs in the source buffer. This calls for a mechanism to signal the current ATM-layer rate to the source TCP layer so that the window size can be set not to exceed the current distance-bandwidth product. This topic needs further investigation.

5 ABR Performance in the Presence of Cross-Traffic

To further study the fairness and convergence properties of our allocation scheme, we examine its performance in a configuration where long connections traversing multiple switches interact with cross-traffic from short connections at one or more hops within the network. The configuration we use is shown in Figure 5.1, and will be referred to as the *R2 configuration*. Such configurations are often called *parking lot* configurations where additional sources begin to share the network as we go further into the network from left to right. In the basic configuration there are 6 source nodes and

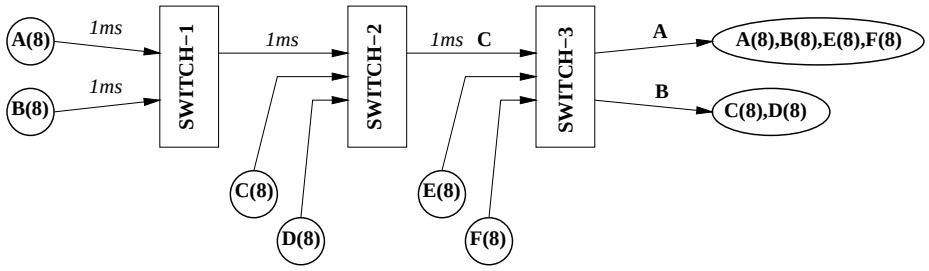

Figure 5.1: Configuration R2.

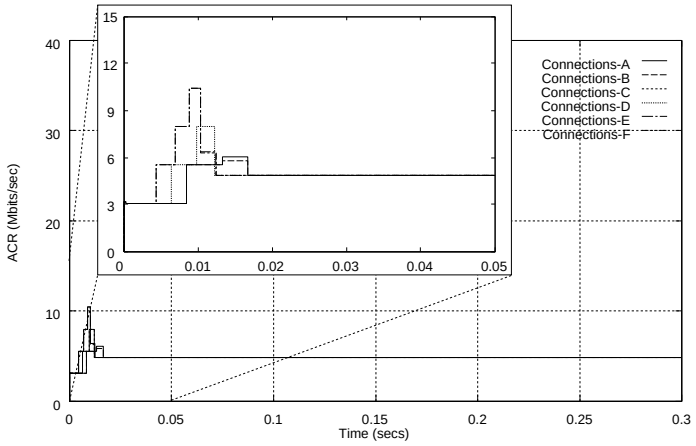

Figure 5.2: ACR for a representative connection from each set in the R2 configuration, with the initial part enlarged to show details.

2 destinations. All the connections traverse the network from left to right. The set of connections A, B, E, F reach their destination through link A and connections C and D go though link B. The number of connections originating/terminating at each node is shown within the circles. Each of the sources originates eight connections, providing a total of 48 connections. We first study this configuration with ABR cell sources only, and subsequently examine the effect of mixing variable bit-rate (VBR) video traffic with the ABR traffic.

5.1 Behavior with ABR Traffic Only

We first study the R2 configuration with only ATM-layer-generated ABR traffic. The source parameters chosen are the same as those used for the R1 configuration in the last section.

The behavior of the ACRs for a representative connection from each of the sources is shown in Figure 5.2, with the initial start-up phase enlarged to show the dynamic behavior of the source rates during convergence. The rates converge to their steady state values in about 16 msecs. Each of the bottleneck links A and C is shared by 32 connections. Thus, every connection sharing one of those links will eventually be allocated $155/32 = 4.84$ Mbits/sec. Since every connection in this configuration crosses at least one of the links A or C, the final rate for each of the connections in the network will be 4.84 Mbits/sec. This is exactly the set of allocations computed by the algorithm in Figure 5.2.

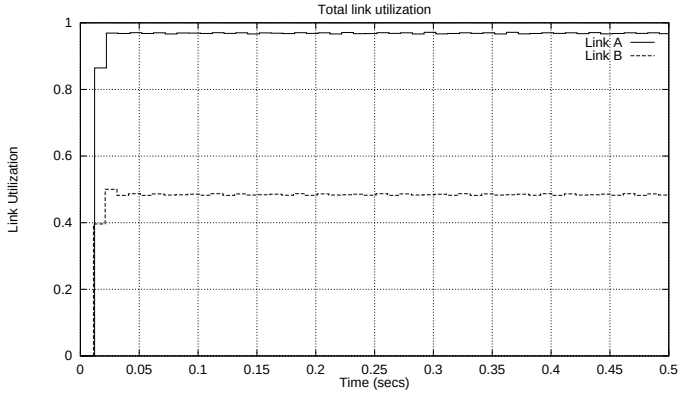

Figure 5.3: Total utilization of links A and B in the R2 configuration.

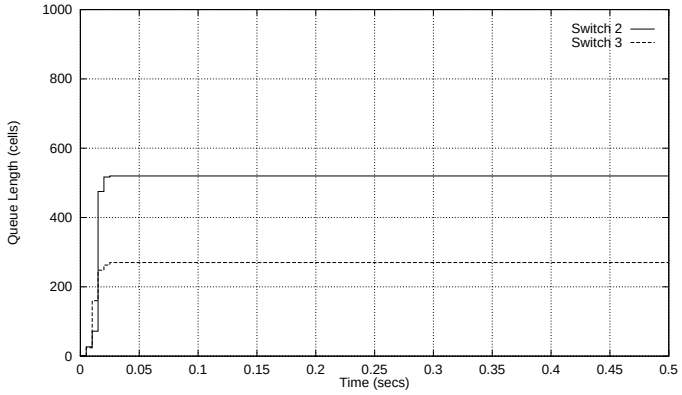

Figure 5.4: Queue length for switches 2 and 3.

There are 16 connections (sets C and D) utilizing link B, all of which share the bottleneck link C (along with sets A and B). Hence, in steady state, link B should reach a utilization of close to 50% (minus the RM cell overhead of 3%). Link A has a total of 32 connections utilizing it, and hence should achieve a utilization close to 100%. The utilizations for these two links are shown in Figure 5.3, and behave as expected. Note also that, because of the relatively small RTT delay for connection sets C,D,E, and F, both links A and B reach their maximum utilization in less than 25 msecs.

From Figure 5.2, it takes about 16 msecs for the rates to converge to their final allocations. Primarily because of the slightly larger RTT delay for connection sets A and B, there is a transient period (as we observed in Figure 4.2), when the arrival rate at a switch exceeds the capacity of the output link. This is because the allocation is based on RM cells that were transmitted previously, resulting in a small amount of over-allocation. This causes a queue to be built up in both the switches 2 and 3 during the transient period. However, the size of these queues is relatively small — about 520 cells for switch-2 and 270 cells for switch 3 (at link A). The queue size for switch-2 is larger because of the larger RTT delay for connections A and B that go through it.

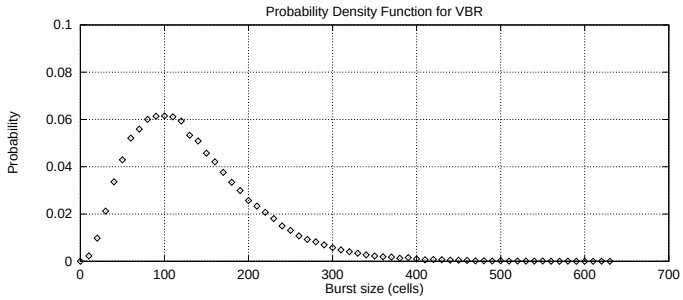

Figure 5.5: Probability density function for the burstiness of VBR traffic.

5.2 Performance of ABR Traffic Mixed with VBR Traffic

Up to now, we have focused on the rate allocation process when considering only ABR traffic. It is important to examine the ability of the scheme to adapt to changes in the available bandwidth, when there is a mix of high-priority traffic such as video and voice. In this section, we study the effects of variable-rate real-time traffic (VBR traffic) on ABR traffic that is rate-controlled using our rate allocation algorithm.

In this set of simulation we use configuration R2 again, but now each source consists of four VBR and four ABR connections, instead of the eight ABR connections in the ABR-only experiments. All the ABR connections request peak bandwidth. The VBR connections have an allocated bandwidth, which is based on the average rate for the video data generated by the application.

The VBR traffic is based on the model described by Heyman et. al. [15]. One frame of video data is generated approximately every 1/25 seconds (the model assumes a PAL system, not an NTSC system which transmits 30 frames/sec) and the size of the frame expressed in number of cells follows the probability density function given in Figure 5.5. The resulting process has a distribution that generates data with an average rate of about 1.5 Mbits/secs. We reserve, in all the links on the path from the source to the destination, a bandwidth of 1.9 Mbits/sec for each VBR connection. Thus, the average utilization of the reserved bandwidth for a VBR connection is expected to be about 75%.

Overall, there are 16 VBR and 16 ABR connection sharing each of the links A and C. Link B carries eight VBR and eight ABR connections. Therefore, a total of approximately $1.9 \times 16 = 30.4$ Mbits/sec is reserved for VBR traffic on links A and C, and about $1.9 \times 8 = 15.2$ Mbits/sec on link B. The VBR traffic is expected to occupy on the average about 15% of the available bandwidth on links A and C, and about 7.5% on link B.

The VBR traffic generated with the model described earlier may exhibit very bursty behavior. In order to limit the burstiness of each VBR connection, we shape its traffic using a token bucket. The bucket size is set to 50 cells and the rate of token arrival was set to be equal to the bandwidth reserved to each connection, that is 4,500 cells/sec=1.9 Mbits/sec. In order to avoid any synchronization between the video streams as much as possible, the corresponding VBR connections open at random times that are uniformly distributed in the interval (0,50 msecs). The signaling activity of opening and closing of a VBR connection is simulated by sending a special cell through the path of the connection, informing the switches on the path about the open/close request. The cell which requests the opening of the connection also carries the the requested bandwidth for the VBR connection. As a result, the available bandwidth to be allocated for ABR connections dynamically changes based on the requests for opening and closing of the VBR connections.

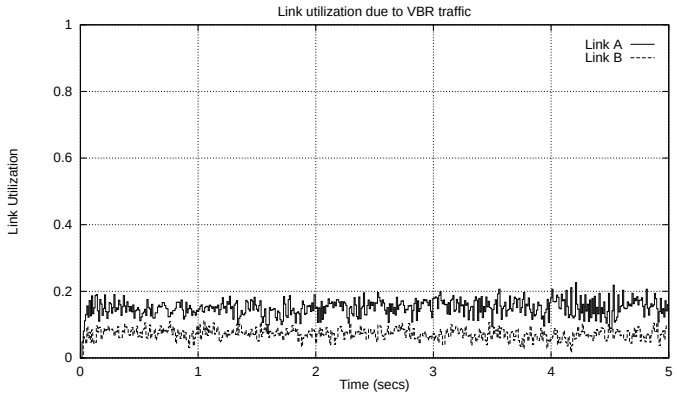

Figure 5.6: Utilization of links A and B due to VBR traffic only.

We assume that the VBR and ABR classes of traffic are buffered in the switches in separate queues. Scheduling between the two classes is based on static priorities, with the VBR traffic always taking higher priority. Thus, the switch transmits an ABR cell only when the VBR queue is empty. Since we are primarily interested in the effect of VBR service on the ABR class, we use a single FIFO queue for all VBR traffic. As always, we assume that the ABR traffic share a single common FIFO queue at each outgoing link of a switch.

To provide an estimate of the portion of the link bandwidths used by VBR traffic, Figure 5.6 shows the utilization of links A and B due to VBR traffic only. The VBR utilization of link A is about 15% and that of link B about 7%. The spikes in the plots are because of the burstiness of the VBR sources. This burstiness is due both to the allowed burstiness of a single connection (cells may be transmitted at peak rate when there are tokens in the token bucket), and to multiple connections transmitting video frames simultaneously.

Figure 5.7 shows the ACR for a representative ABR connection from each source node (other connections from the same source exhibit similar behavior). The rate allocation process converges quickly (within 50 msecs) to the final allocation. After convergence, the rate allocated to each connection is a fair share of the available bandwidth: this is the total bandwidth minus the bandwidth reserved for VBR traffic. Here, the bandwidth available to ABR traffic on links A and C is 124.6 Mbits/sec and therefore each ABR connection has an available capacity of about 7.8 Mbits/sec.

The expected link utilization is close to 100% for link A and 50% for link B. However, the instantaneous link utilization (averaged over 5 msec intervals) exhibits spikes as shown in Figure 5.8. Accounting for the overhead of 3% of the available bandwidth for RM cells, the utilization of the links is close to the maximum attainable for both the links A and B. The difference is simply because of the over-allocation of bandwidth that we did for the VBR connections: on the average, the VBR traffic should utilize only 75% of its allocated bandwidth. However, this conservative over-allocation for the VBR connections has the desirable side effect of maintaining the queue sizes small. Although the utilization is kept high, the queue sizes for ABR traffic remain small even in the presence of VBR traffic. The queue lengths for ABR traffic in switches 2 and 3 are shown in Figures 5.9 and 5.10, respectively. The queue sizes have an average of about 50 cells (which is also the size of the token bucket of a VBR connection) and a maximum of about 200 cells in steady state.

Figure 5.11 shows the behavior of the ACRs of the ABR connections when the VBR connections open in a staggered fashion. Six VBR connections (one from each source) open every 100 msecs

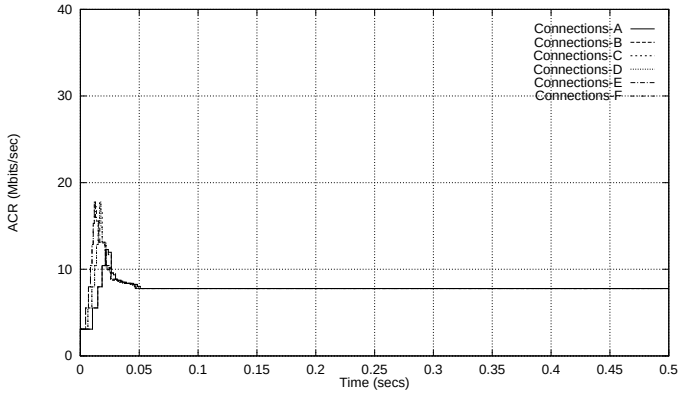

Figure 5.7: ACR for each connection set.

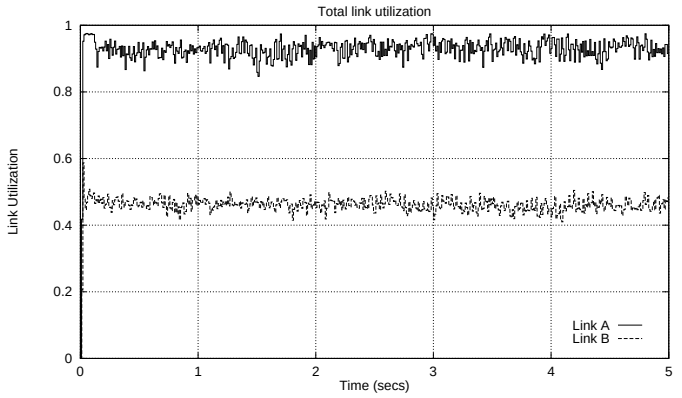

Figure 5.8: Total utilization of links A and B.

starting at time $t = 0$ seconds. Then, at time $t = 0.5$ seconds, these VBR connections start closing in a similar manner; that is, one VBR connection from each source closes every 100 msecs. Any time a VBR connection opens or closes, the bandwidth available to the ABR connections changes. Therefore, a recomputation of the transmission rates for the ABR connections is triggered. As shown in the figure, the convergence to the final allocation after each change in available bandwidth is rapid. In the worst case, the rate allocation process is completed within 20 msecs.

When a VBR connection opens, the convergence of the ABR connections to the final rate is faster than when a VBR connection closes. This is because, on opening a VBR connection, the allocation to all the ABR connections decreases, due to a reduction in the available bandwidth. Since all ABR connections request peak bandwidth, they are allocated B_{eq} of their available capacity at the bottleneck link, and convergence to this value is rapid. When a VBR connection closes, however, the released available bandwidth is first given to the shorter ABR connections. Subsequently, as RM cells are received from the longer connections, the rate allocation process fairly allocates this bandwidth among all the ABR connections. This explains the longer convergence time on the closing of an ABR connection, and hence a small queue build-up. This queue quickly drains because of the difference between the allocated bandwidth for the VBR connections and their average rates.

Our results suggest that the rate allocation algorithm will perform well even in the presence of different service classes. For the specific configuration considered, its performance was efficient

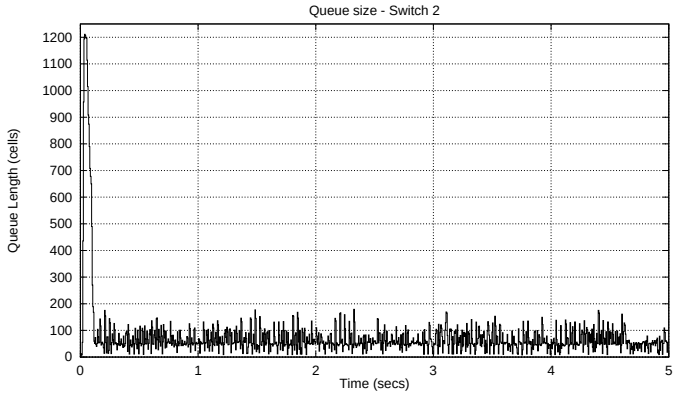

Figure 5.9: Queue size for ABR traffic – Switch 2.

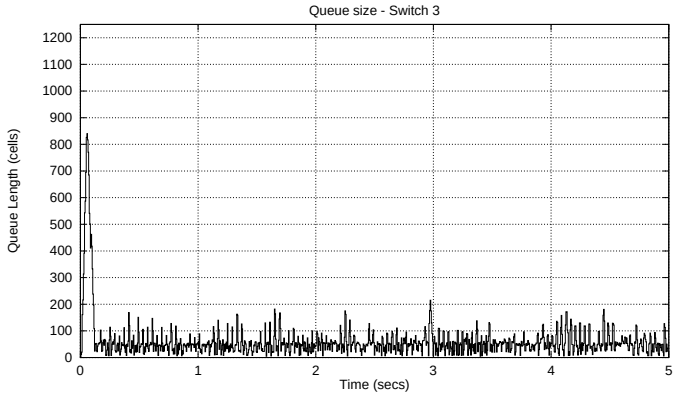

Figure 5.10: Queue size for ABR traffic – Switch 3.

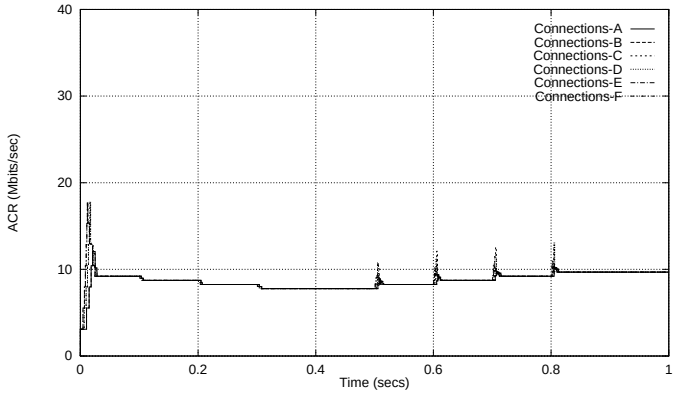

Figure 5.11: ACR for each connection set with staggered opening and closing of VBR connections at 100 millisecond intervals.

and scaled well with the number of connections. However, further work is needed to understand in greater detail the dynamics of the algorithm in more general network configurations.

6 Performance of TCP in a Dynamic Environment

In Section 4.2, we examined the behavior of the rate-allocation algorithm in a network configurations with only TCP-generated ABR traffic. In practice, TCP traffic may share the network with ATM-layer-generated ABR traffic. In addition, ABR connections may open and close frequently, thus triggering re-allocations of transmission rates at the switches frequently. In this section, we study the performance of TCP in a network configuration in which TCP connections share the links with connections from ABR cell source which are made to open and close continually.

The configuration we use for the simulations in this section is identical to the parking-lot configuration R2, except that the number of connections is different. The modified configuration is shown in Figure 6.1, and will be referred to as the *R3 configuration*. Each of the links has a capacity of 155 Mbits/sec and a propagation delay of 1 msec. Two types of connections are simulated: There are eight TCP connections that start up at time 0 and are kept open throughout the simulation. In addition, there are four ABR connections that carry ATM-layer-generated traffic. To simulate a dynamic network environment, the latter connections are made to open and close randomly, thus changing the bandwidth available for allocation at the switches to the TCP connections. The ON and OFF periods of these connections are exponentially distributed, with a mean of both set to 100 milliseconds. Note that this mean interval is more than 10 times the round-trip delay of the longest connection.

Traffic from TCP connections is destined to output link A of switch 3, while the remaining traffic is destined to output link B of the same switch. Since TCP connections 7 and 8 are not bottlenecked anywhere in the network except at switch 3, under ideal conditions the utilization of link A can approach 100%. However, because of the overhead due to the SONET physical layer, ATM cell header, RM cells, etc., the maximum achievable effective throughput for TCP is only about 87% of the available link capacity, that is, about 135 Mbits/sec.

The random opening and closing of the ATM-layer connections trigger frequent re-computations of the allocations at the switches. To observe the effect of possible transient over-allocations during the convergence of the algorithm, we chose a very small buffer size of 1 Kbyte for the switches, making

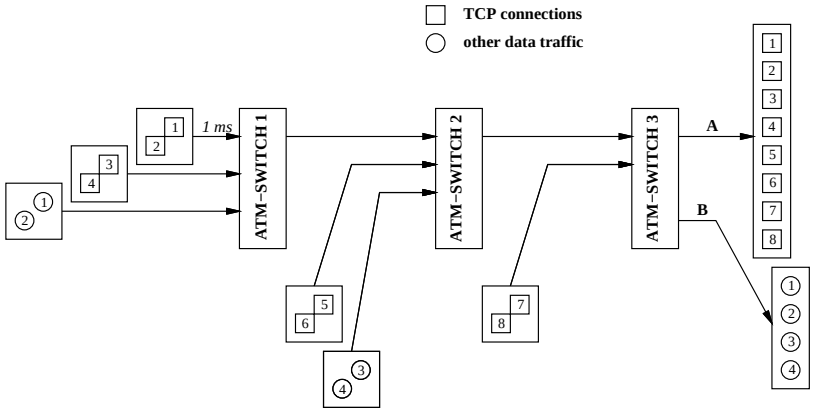

Figure 6.1: The R3 network configuration.

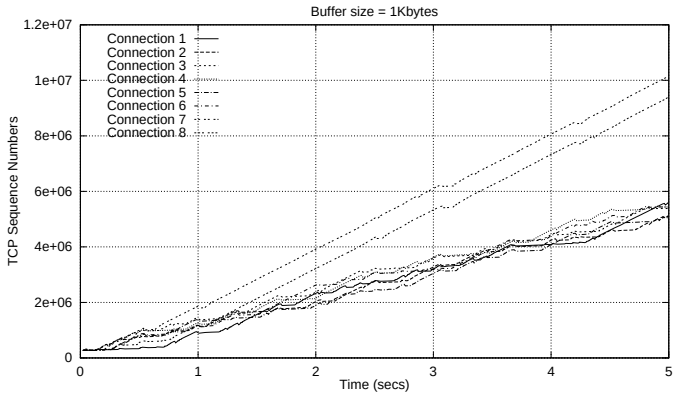

Figure 6.2: TCP sequence numbers (1 Kbytes buffer size).

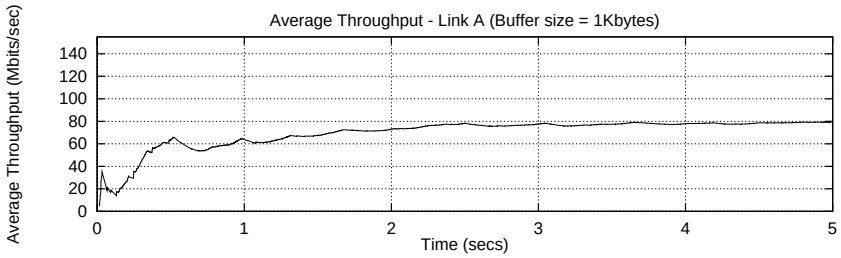

Figure 6.3: Average TCP throughput on link A (1 Kbytes buffer size).

a packet loss very likely if a transient over-allocation occurs during convergence of the algorithm after a change in the connection states. Note that the buffer size of 1 Kbytes is much smaller than the largest TCP segment; thus, it would be difficult for TCP connections to make sustained forward progress if this were a datagram-based network.

Figure 6.2 shows the progress of the sequence numbers for the eight TCP connections. All of the connections make steady progress and, except for the two connections closest to the destination, their throughputs are almost identical. Connections 7 and 8, being closest to the destination, are able to use more than their fair share during transient periods after the closing of a connection, when the new allocations have not yet reached the longer connections. Note that this behavior does not represent any inherent unfairness in the rate allocation algorithm, but is due to the delay of the control loop.

The aggregate throughput sustained by the TCP connections (as averaged from time $t = 0$ seconds) sharing the link A is shown in Figure 6.3. It is interesting to observe that, even with only 1 Kbyte of available buffer, not only all the connections make steady progress, but the average sustained throughput is about 60% of the maximum attainable throughput of 135 Mbits/sec.

7 Conclusion

ATM's Available Bit Rate (ABR) class of service has been proposed as the service appropriate for carrying data traffic. Congestion management is an inherent part of the ABR service. This is because little, if any, knowledge is assumed for the characteristics of the application-generated traffic, and connections are allowed to opportunistically take advantage of the available capacity in

the network for transmitting data. Some of the goals of the ABR class of service are to achieve efficiency in utilizing the available bandwidth, maintain very low cell-loss and delay, and to achieve some degree of fairness. One of the suggested fairness goals is that the VCs achieve rates that are *max-min* fair. The approach taken in the ATM Forum has been to specify the end-system behavior, while allowing some vendor latitude in the algorithms adopted in the switches to support the ABR class of service.

We had proposed a rate allocation algorithm in switches, that achieves, in the steady state, a *max-min* fair allocation of the available bandwidth, while being efficient and scalable [5]. The scalability comes from the fact that the algorithm is of constant complexity ($O(1)$), where a fixed amount of work is performed for each resource management (RM) cell that is received by the switch.

In this paper, we evaluated the performance of the explicit rate allocation algorithm in a variety of network configurations and with a diverse range of workloads. We examined the fairness characteristics in a configuration where connections with widely different round-trip times (and hence feedback delays) share a common bottleneck. This may be representative of environments where a switch in the premises of an end-user has both local LAN connections and WAN connections going through it. Traditionally, feedback-based congestion control mechanisms have shown a bias towards flows with a short round-trip time when they co-exist with long round-trip time flows. Because of the rate allocation mechanism we use here, this bias is eliminated, and we see a fair allocation even under the extreme condition where the round-trip times differ by three orders of magnitude.

We also show that the proposed allocation algorithm retains several desirable characteristics as we scale in the number of connections. We increased the number of connections from 3 to 40 (with a predominance of long round-trip time connections), and observed that the fairness properties are maintained. Convergence to the steady-state max-min fair allocation is even more rapid. The queueing requirements go up only by a factor of 3 with the increase from 2 to 32 of the long-round-trip time connections (with 16 milliseconds round-trip time). This is in-spite of the initial rate of the connections being over 80% of their final steady-state rate.

An important requirement for the ABR service is that it mesh well with traditional higher-layer protocols such as TCP/IP. It is well known that TCP exhibits unfairness when multiple TCP connections sharing a bottleneck link have widely different round-trip times. We show that TCP running over ABR avoids this unfairness: in the R1 configuration, throughputs of all the three TCP connections are fair; there is a dramatic reduction in the queueing requirements at the bottleneck link; and no packet losses occur due to congestion, because the total queue occupancy in the switches is less than 1K cells (the switch buffer is shared among all the connections in a FIFO manner).

An observation we make is that TCP running over ABR with our allocation algorithm continues to increase its window size up to the maximum allowed by the source, in the absence of any congestive loss. We assume that in a realistically designed end-system, there would be back-pressure from the datalink layer to the higher layers on that system to avoid any loss internally within the implementation of the protocol stack. Thus, since the congestion management algorithms in ATM's ABR service maintains the switch queue lengths small, the only time when TCP's congestion recovery procedures need to be invoked is when there is a packet loss. We find that the recovery mechanisms in TCP (fast retransmit and fast recovery) due to a packet loss (when "slow start" is not triggered) reasonably mesh with the ATM ABR congestion management/rate allocation scheme studied here. The loss in useful link utilization during the recovery interval is brief and reasonably small.

Another important need is for ABR flows to operate well when there are higher-priority VBR flows co-existing with the ABR traffic. In our simulations of ABR connections in the presence of

VBR traffic, about 15% of the link bandwidth was allocated for VBR traffic (compressed video) on the bottleneck links. The average rate of the VBR traffic was about 75 % of the allocation, with a leaky bucket size of 50 cells. The available bandwidth for the ABR flows was recalculated every time a new VBR connection was admitted, or when a VBR connection closed. In the *parking lot* configuration with 48 connections (equally divided as 24 ABR and 24 VBR flows), we achieved fairness, and full utilization of the bottleneck links. In spite of the burstiness of the VBR traffic and the greediness of the ABR flows, the queue sizes were of the order of only 100 cells, which is quite reasonable.

While we believe that the rate allocation algorithm we have proposed is robust against cell losses, including RM cells, we have not yet examined the interaction with non-cooperative sources, since there is a common FIFO queue which is shared by all of the ABR connections at the switches. The use of a per-VC queue for purposes of isolation is attractive from this perspective. In addition, we have to examine the overall system performance when not all flows use their stated bandwidth, as specified in the CCR field of the RM cells.

References

[1] N. Giroux and D. Chiswell, "ATM-layer traffic management functions and procedures," in *Proceedings of INTEROP '95 Engineer Conference*, March 1995.

[2] F. Bonomi and K. W. Fendick, "The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service," *IEEE Network*, vol. 9, no. 2, pp. 25–39, March/April 1995.

[3] D. Bertsekas and R. Gallager, *Data Networks*. Prentice Hall, 2nd ed., 1992.

[4] A. Charny, "An Algorithm for Rate Allocation in a Packet-Switching Network with Feedback," Master's thesis, Massachusets Institute of Technology, May 1994.

[5] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, "An efficient rate allocation algorithm for ATM networks providing max-min fairness," in *Proceedings of 6th IFIP International Conference on High Performance Networking, HPN'95*, September 1995.

[6] R. Jain, "Congestion Control and Traffic Management in ATM Networks: Recent Advances and A Survey." submitted to Computer Networks and ISDN Systems.

[7] S. Floyd and A. Romanow, "Dynamics of TCP traffic over ATM networks," in *Proceedings of ACM SIGCOMM'94*, pp. 79–88, September 1994.

[8] L. Kalampoukas and A. Varma, "Performance of TCP over Multi-Hop ATM Networks: A Comparative Study of ATM Layer Congestion Control Schemes," in *Proceedings of ICC'95*, June 1995.

[9] L. Kalampoukas and A. Varma, "Analysis of Source Policy in Rate-Controlled ATM Networks," submitted to ICC'96.

[10] K.-Y. Siu and H.-Y. Tzeng, "Intelligent Congestion Control for ABR Service in ATM Networks," *Computer Communication Review*, vol. 24, no. 5, pp. 81–106, October 1994.

[11] CCITT, *Draft Recommendation I.363*. CCITT Study Group XVIII, Geneva, January 1993.

[12] V. Jacobson, "Congestion avoidance and control," in *Proceedings of ACM SIGCOMM'88*, pp. 314–329, 1988.

[13] P. Mishra and H. Kanakia, "A hop by hop rate-based congestion control scheme," in *Proc. of SIGCOMM92*, (Baltimore, Maryland.), October 1992.

[14] S. Floyd and V. Jacobson, "On traffic phase effects in packet-switched gateways," *Intenetworking: Research and Experience*, pp. 115–156, September 1992.

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