11
An efficient rate allocation algorithm for ATM networks providing max-min fairness

*L. Kalampoukas, A. Varma**
Computer Engineering Department
University of California, Santa Cruz, CA 95064, USA
E-mail: {lampros,varma}@cse.ucsc.edu

K. K. Ramakrishnan
AT&T Bell Laboratories, Murray Hill, NJ 07974, USA
E-mail: kkrama@research.att.com

Abstract

We describe a new algorithm for rate allocation within the individual switches of an ATM network implementing a rate-based congestion control algorithm for Available Bit-Rate (ABR) traffic. The algorithm performs an allocation in $\Theta(1)$ time, allowing it to be applied to ATM switches supporting a large number of virtual circuits. When the total available capacity or the requests of the individual connections change, the algorithm converges to the max-min allocation. Results from simulations using ATM sources show that the algorithm provides close to ideal throughput and converges to the max-min fair allocation rapidly when the available bandwidth or the individual requests change.

Keywords

Rate control, max-min fairness, ATM congestion control

*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.

R. Puigjaner (ed.), *High Performance Networking*

1 INTRODUCTION

Broadband packet networks based on Asynchronous Transfer Mode (ATM) are enabling the integration of traffic with a wide range of characteristics within a single communication network. In these networks, all communication at the ATM layer is in terms of fixed-size cells consisting of 48 bytes of payload and 5 bytes for the ATM-layer header. Routing of cells is accomplished through packet switches over virtual circuits set up between endpoints.

Many service classes have been defined for support of various types of applications in ATM networks, ranging from real-time video delivery with stringent Quality-of-Service (QoS) requirements to electronic mail services requiring no such guarantees. The *Available Bit-Rate* (ABR) service class was defined for the support of best-effort applications that may require a minimum-bandwidth guarantee [Giroux, 1995]. This service class allows applications to fully utilize the available bandwidth in the network by adjusting their instantaneous transmission rates to the available capacity. A congestion control scheme is essential for the support of ABR traffic to utilize the available network bandwidth without causing congestion, and to provide a fair allocation of the available bandwidth among the connections sharing the network. The ATM Forum Traffic Management Committee recently chose a rate-based congestion control framework to satisfy this objective [Bonomi, 1995].

With the explicit-rate option for the ATM Forum congestion control scheme, the source of each connection periodically transmits a special *resource management* (RM) cell. The RM cell specifies the bandwidth requested by the connection, as well as the current transmission rate of the source of the connection. 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.

Computing the fair share of a connection on an outgoing link of a switch requires an algorithm for fair allocation of bandwidth among the connections sharing the outgoing link. On receipt of an RM cell from a connection, the algorithm is invoked to determine the current bandwidth allocation of the connection on the output link, taking into account the available bandwidth and the current allocations of other connections. Such an algorithm for rate allocation in packet-switched networks was described by Charny [Charny, 1994]. Upon convergence, the allocations computed by this algorithm have been shown to satisfy max-min fairness. The algorithm, however, requires examining the state of other active connections on the receipt of each RM cell, making its worst-case execution time $O(N)$, where N is the number of active connections. This makes it difficult to apply in an ATM switch supporting a large number of virtual channels per output link.

Our primary contribution in this paper is a new rate allocation algorithm with an execution time of $\Theta(1)$. Because its execution time is independent of the number of virtual channels

sharing the outgoing link, the algorithm is attractive for implementation in ATM switches supporting a large number of virtual channels. When the total available capacity or the requests of the individual connections change, the algorithm can be shown to converge to the max-min allocation.

The remainder of the paper is organized as follows: We introduce the proposed algorithm in Section 2 and discuss its properties in Section 3. In Section 4, we evaluate the dynamic behavior of the algorithm by simulation in an example multi-hop network configuration. We conclude the paper in Section 5 with a summary of the results and directions for future work.

2 DESCRIPTION OF THE ALGORITHM

The rate-based congestion control framework, on which our algorithm depends, is similar to the one described by Charny [Charny, 1994]. The source of each connection requests a specific amount of bandwidth to be allocated to the connection by the switches on its path to the destination. The request is carried by special RM cells transmitted periodically by the source. We assume that each 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 transmission rate of the connection, called *current cell rate* (CCR). The CCR field is set by the source of the connection and is not modified by the switches. It is assumed that the source would set the CCR field based on the ER value in the RM cell that was returned by the network most recently. The latter reflects the bandwidth available at the bottleneck link in the network at the time the most recent RM cell traversed it.
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. In the following description of the algorithm, bandwidth allocation is performed when an RM cell traveling in the forward direction is received at a switch. No processing is performed by the switches on an RM cell traveling backwards to the source.

Note that ER is the bandwidth requested by the connection during the current epoch, while CCR is the bandwidth it was allocated during the last round-trip epoch. As the RM cell passes through the network, each switch reads the ER field and attempts to allocate the requested bandwidth on the output link. If a switch is unable to allocate the requested bandwidth on its outgoing link, it modifies the ER field to the maximum bandwidth it can allocate to the connection on the outgoing link. When the destination of a connection receives an RM cell, the cell is immediately returned to the source through a reverse path. On receipt of the RM cell, the source adjusts its rate to be less than or equal to the value

in the ER field, which represents the maximum fair bandwidth available to the connection on the bottleneck link along its path. The CCR of subsequent RM cells will be set to reflect this new allocated rate. The source may request a change in its allocated rate by setting the ER field of RM cells by means of a suitable increase/decrease algorithm. In this paper, our focus is on the rate allocations algorithm within the switch. Therefore, we assume a simple source algorithm in which the source always sets ER to the peak link bandwidth; both the CCR field and the current transmission rate of the source are set to the value in the ER field of the RM cell returned by the network most recently.

The rate control algorithm is applied to each switch of the network independently. Within a switch, separate instances of the algorithm are executed at each output port to control the allocation of bandwidth among the connections sharing the corresponding outgoing link.

Rate allocation within a switch must be performed according to some fairness criterion, so that the total available bandwidth on the outbound link is divided among the connections sharing the link fairly. A commonly used fairness criterion is *max-min fairness* [Bertsekas, 1992, Ramakrishnan, 1987]. Let $\rho_1, \rho_2, \ldots, \rho_N$ be the requests of individual connections sharing the link and $A_1, A_2, \ldots, A_N$ be the allocation produced by the algorithm. The allocation is max-min fair, if a request is either completely satisfied, or receives an allocation equal to the maximum among all connections. That is, connections that are not completely satisfied by the allocation receive equal shares of the output bandwidth. That is,

$$\text{if } A_i < \rho_i, \text{ then } A_i = \max_{1 \le j \le N} A_j.$$

In addition, the sum of A_i's should equal the total bandwidth available if there is at least one connection with $A_i < \rho_i$.

Before we proceed to describe the rate allocation algorithm, we pause to introduce some definitions and notations. 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 $\rho_i(t)$ the value of the most recent request from connection i, as taken from the ER field of the most recent RM cell received in the forward direction from that connection, and $CCR_i(t)$ be the corresponding value in the CCR field. $A_i(t)$ represents the corresponding allocation received by connection i during its most recent update.

Now we can describe the main ideas behind the algorithm. The goal of the algorithm is to make available to each bottlenecked connection at time t, a maximum bandwidth equal to

$$A_{max}(t) = \frac{\text{Total bandwidth available to bottlenecked connections}}{\text{Number of bottlenecked connections}} \tag{1}$$

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

$$A_{max}(t) = \frac{B(t) - \sum_{i \in S_u(t)} A_i(t)}{N_b(t)} \tag{2}$$

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) 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) to determine the maximum bandwidth that would be allocated to it. The following equation is used to determine A_{max} in this case:

$$A_{max}(t) = \frac{B(t) - \sum_{i \in S_u(t)} A_i(t) + A_j(t)}{N_b(t) + 1}. \tag{3}$$

The computations in both equations (2) and (3) can be performed without searching the state of each connection by maintaining the residual bandwidth available for allocation to bottlenecked connections, given by

$$B_b(t) = B(t) - \sum_{i \in S_u(t)} A_i(t). \tag{4}$$

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

$$B_f(t) = B(t) - \sum_{i \in S_u(t)} A_i(t) - \sum_{i \in S_b(t)} \frac{B(t)}{N(t)}. \tag{5}$$

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. (5) can also be written as

$$B_f(t) = N_u(t) B_{eq}(t) - \sum_{i \in S_u(t)} A_i(t) \tag{6}$$

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. (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), used to check state changes for bottlenecked connection, as follows:

$$A_{max}(t) = B_{eq}(t) + \frac{B_f(t)}{N_b(t)}. \tag{7}$$

Similarly, we re-write Eq. (3), used to check state changes for a satisfied connection, as follows:

$$A_{max}(t) = B_{eq}(t) + \frac{B_f(t) + A_j(t) - B_{eq}(t)}{N_b(t) + 1}. \tag{8}$$

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. (7) and (8) into a single equation:

$$A_{max}(t) = B_{eq}(t) + \frac{B_f(t) + A'_j(t) - B_{eq}(t)}{N_b(t_1) + w}, \tag{9}$$

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

Thus, Eq. (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 $A_i(t+)$ for connection j is computed based on the parameters of the RM cell received at t as

$$A_j(t+) = \min(A_{max}(t), \rho_j(t), CCR_j(t)). \tag{10}$$

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

$$B_f(t+) = B_f(t) + A'_j(t) - A'_j(t_1+). \tag{11}$$

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 (9) and (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 above description 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. The complete pseudocode of the algorithm can be found in [Kalampoukas, 1995].

3 PROPERTIES OF THE ALGORITHM

The proposed algorithm has some important properties making it especially attractive for application to ATM and to packet switching networks in general. Perhaps its most important merit is the implementation complexity. The complexity of $\Theta(1)$ makes it attractive for use in ATM switches supporting very large number of virtual channels. This improves upon the scheme proposed by Charny [Charny, 1994], which has an asymptotic complexity of $O(N)$ where N is the number of active connections.

When the individual requests ρ_i and the available bandwidth are steady, the algorithm can be shown to converge to a max-min fair allocation in finite time [Kalampoukas, 1995]. In addition, results from simulations show that the algorithm is responsive to network changes. Note that the algorithm always computes a precise allocation based on the current state of the requests and available bandwidth; no approximations are employed in the computations, and there are no parameters in the algorithm.

The algorithm can be applied to a network with heterogeneous link capacities. It is asynchronous and distributed, and is executed independently by the switches which compute the fair bandwidth allocations based only on the bandwidth requests and the bandwidth available on the outbound links. The algorithm is robust in the sense that the loss of one or more resource management cells does not affect its correctness or stability. However, it is obvious that such losses can cause a delay in the reaction of the sources.

4 SIMULATION RESULTS

To study the dynamic behavior of the rate allocation algorithm when the requests in the network change rapidly, we simulated the algorithm in several example network configurations. Results from one such configuration are presented in this section.

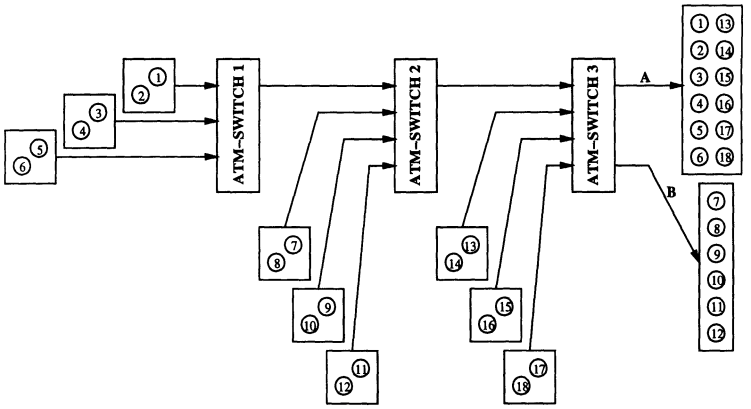
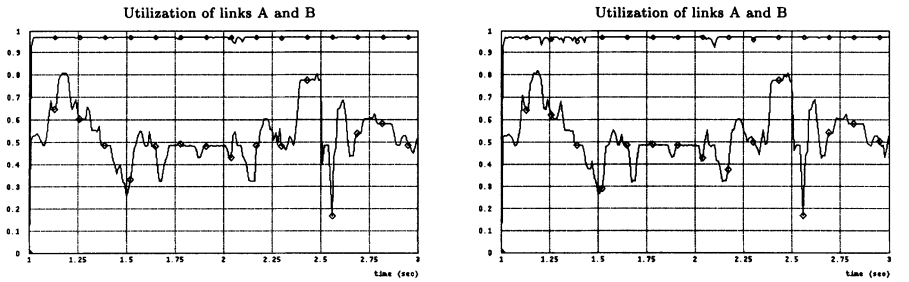


Figure 1 Network configuration for simulations with ATM sources.

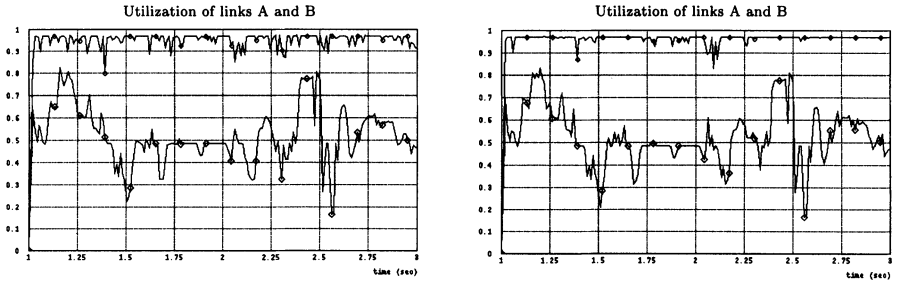
Figure 1 shows the network configuration used for the simulations. There is a total of 18 connections in the network belonging to two different types: those using link A and those using link B. Twelve of them are destined to output link A of switch 3 and the remaining six are destined to output link B of the same switch. Note that the connections using link B share the link between switches 2 and 3 with some of the connections using link A, contributing to cross traffic in the system. Each node contains two ATM traffic sources, each generating traffic based on a simple ON-OFF model where the durations of the ON and OFF periods are exponentially distributed. The means of the ON and OFF intervals were set to 0.2 seconds each in our simulations, generating on the average 45 ON/OFF requests per second. Data traffic is transmitted only in the forward direction (that is, from left to right). In all the simulations, the sources always requested the peak link bandwidth for the entire duration of their connections. A simple source policy was assumed where the transmission rate of the source was always set to the value in the ER field of the most recent RM cell belonging to the connection that was returned by the network. The spacing between consecutive cells transmitted by the source is $1/r$, where r is the current transmission rate of the connection. The rate of transmission of RM cells at each source was chosen as one every 32 cells.

Figures 2 and 3 present the utilization of links A and B for two choices of propagation delays per link, namely 0.1 ms and 1 ms, and for two buffer sizes, 8 KBytes and 64 KBytes. Each point in the graph is based on averaging the utilization over an interval of 10 ms. Link A can be expected to be fully utilized since connections 13–18 cannot be bottlenecked at any other switch. The average utilization of link B can be expected to be close to 50% based on the following reasoning: According to the ON/OFF model used, half of the connections are expected to be open on the average at any time. Therefore, the bandwidth of the output link of switch 2 should be allocated equally among the active connections of 1–6 and 7–12. The bandwidth expected to be allocated to the active connections among 1–6 is 50%.



(a) (b)

Figure 2 Utilization of links A and B for 0.1 ms delay/link. (a) buffer size = 8 KBytes (b) buffer size = 64 KBytes. The upper plot represents the utilization of link A and the lower that of link B.



(a) (b)

Figure 3 Utilization of links A and B for 1 ms delay/link. (a) buffer size = 8 KBytes (b) buffer size = 64 KBytes.

The same holds for 7–12. Since the output link of switch 2 is the only bottleneck link for connections 7–12, the expected utilization of link B is 50%. However, due to the exponential distribution of the ON/OFF interval, it is possible for the instantaneous number of active connections among 1–6 to be different from those among 7–12. Thus, during these periods, the instantaneous utilization of link B can depart considerably from 50%.

Figures 2 and 3 verify this behavior. The upper plot in these diagrams gives the utilization of link A and the lower that of link B. Note that in all cases the utilization of link A is close to 100% while the utilization of link B varies around the mean value of 50%. The average utilization of link A over 2 seconds of simulation time is close to 97% in Figure 2, the maximum achievable when the bandwidth used for transmission of the RM cells is taken into account. The average utilization is slightly lower in Figure 3, owing to the longer feedback delay.

As the link propagation delays increase, the responsiveness of the algorithm can be expected to decrease. Thus, although the average utilization may remain high, there may be

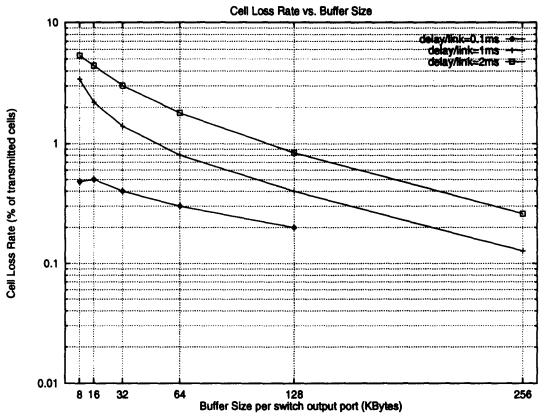


Figure 4 Cell loss rate versus switch buffer size per output port for the A connections.

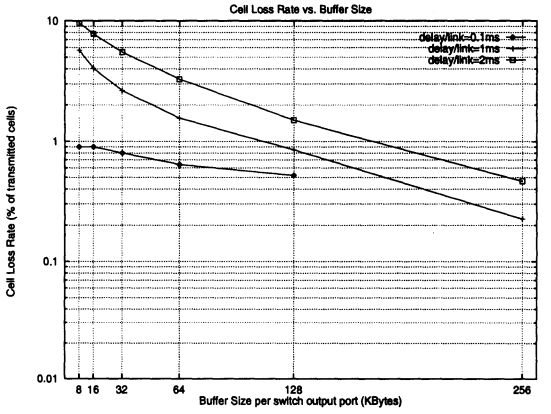


Figure 5 Cell loss rate versus switch buffer size per output port for the B connections.

intervals where the link is under- or over-utilized by a connection. The duration of these periods of instability is determined by the convergence time of the algorithm in the specific network configuration. Over-utilization of bandwidth can result in cell losses. The cell loss rates for the connections using links A and B are shown in Figures 4 and 5, respectively, for varying buffer size and link delays. The cell loss rate for a specific buffer size is shown in the graph only if it is non-zero. The loss rate for a link-delay of 0.1 ms is zero when the buffer size is at least 128 KBytes. Similarly, for link delays of 1 and 2 ms, the loss rate is zero when the buffer size is larger than 256 KBytes. Note that these loss rates are highly exaggerated because of the greedy source policy assumed in our simulations. In practice, the sources increase their rates only gradually when the returned RM cell indicates an increase in its

allocation [Bonomi, 1995], resulting in substantially lower loss rates than those in Figures 4 and 5.

5 CONCLUSIONS

In this paper, we proposed a new algorithm for rate allocation within the individual switches of an ATM network implementing a rate-based congestion control algorithm for ABR traffic. The algorithm performs an allocation in $\Theta(1)$ time, allowing it to be applied to ATM switches with a large number of virtual channels. When the total available capacity or the requests of the individual connections change, the algorithm converges to the max-min fair allocation. We have also verified that the scheme maintains max-min fair allocation even when the round-trip times of connections sharing a bottleneck link are widely different.

In the future we plan to study the dynamic behavior of the algorithm in more complex network configurations and with more complex source policies. We are also investigating the possibility of implementing the algorithm in a hardware simulation testbed [Varma, 1995].

REFERENCES

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

[Bonomi, 1995] 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, March/April 1995, pp. 25–39.

[Charny, 1994] A. Charny, "An algorithm for rate allocation in a packet-switching network with feedback", M.S. Thesis, Laboratory for Computer Science, MIT, Technical Report MIT/LCS/TR-601, April 1994.

[Kalampoukas, 1995] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, "An efficient rate allocation algorithm for ATM networks providing max-min fairness," Technical report UCSC-CRL-95-29, Computer Engineering Dept., University of California, Santa Cruz, June 1995.

[Varma, 1995] A. Varma and D. Stiliadis, "FAST: An FPGA-based simulation testbed for ATM switching networks," Proc. INTEROP '95 Engineer Conference, March 1995.

[Bertsekas, 1992] D. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, Inc., 1992.

[Ramakrishnan, 1987] K. K. Ramakrishnan, D.-M. Chiu, and R. Jain, "Congestion avoidance in computer networks with a connectionless network layer, Part IV: A selective binary feedback scheme for general topologies," Technical Report DEC-TR-510, Digital Equipment Corporation, November 1987.

[Siu, 1995] K.-Y- Siu and H.-Y. Tzeng, "Intelligent congestion control for ABR service in ATM networks," *ACM Computer Communication Review*, Vol. 24, No. 5, October 1995, pp. 81–106.