ABSTRACT

A central issue in congestion control for available bit rate (ABR) service in ATM networks is the computation of the fair rate for every connection at each switching node in the network. The objective is to determine the fair rate for all the connections in a distributed network under dynamic changes in the absence of centralized knowledge about the network, and without the synchronization of different network components. The authors' focus in this article is on studying the problem of fair rate allocation, specifying the requirements of a fair rate allocation algorithm, and providing a survey of various proposed fair rate allocation strategies in the context of ABR service.

Allocating Fair Rates for Available Bit Rate Service in ATM Networks

Ambalavanar Arulambalam and Xiaoqiang Chen, Bell Laboratories

Nirwan Ansari, New Jersey Institute of Technology

Available bit rate (ABR) service as defined in the standard bodies is intended for applications with bursty traffic. These applications can be characterized as unlikely to be able to predict bandwidth requirements, sensitive to cell loss, but able to tolerate certain delay. ABR is designed in such a way that these applications can grab any unused network resources (bandwidth and buffer space). Gains due to statistical resource utilization, however, come at the risk of potential congestion when many applications compete for network resources. Proper congestion control must be in place to ensure that the network resources can be shared in a fair manner and that performance objectives such as cell loss ratio can be maintained. A flow control mechanism is specified by the ATM Forum in [1] which supports several types of feedback to control the source rate in response to changing transfer characteristics. This feedback information is conveyed to the source, which adapts its traffic in accordance with the feedback. The feedback information includes the state of congestion and a fair share of the available bandwidth according to a network-specific allocation policy. To ensure interoperability, an ABR end system must always implement the standard-defined source and destination behavior.

A network switch is responsible for allocating the fair share of the bandwidth among all connections that compete at this switch point. Since this allocation policy is implementation-specific, it has been at the center of switch design and implementation for the last few years. This issue has become an important differentiating factor for the next generation of commercially available switches. Our aim in this article is to provide insights into the design and implementation of fair rate allocation algorithms. We begin with an overview of design guidelines and issues designers must consider. Then our focus will shift to various fair rate allocation algorithms that have been published so far in the literature, including one algorithm designed by the authors. Early implementation of congestion control was based on binary feedback information where switches only mark a single congestion bit in the data cells in the event of congestion. The inherent shortcomings of the binary approach have recently led to the consideration and implementation of sophisticated fair rate allocation algorithms that compute fair rates for each connection and convey this information to the sources. A number of such algorithms have been proposed and studied in the literature, and they can be in general classified broadly into two approaches: fair rate approximation and exact fair rate computation, depending on the congestion monitoring criteria and congestion detection methods. The relative merits and detailed performance characteristics of these various algorithms will be presented whenever the space permits. This article is concluded with future directions on research in this area.

CONGESTION CONTROL FRAMEWORK

The ABR congestion control scheme is a rate-based, closed-loop, per-connection control which utilizes the feedback information from the network to regulate the rate of transmitting cells at the source. The source generates special probe cells called resource management (RM) cells in proportion to its current data cell rate. The destination will turn around and send back the RM cells to the source in the backward direction. The RM cells which can be examined and modified by the switches in both forward and backward directions carry the feedback information of the state of congestion and the fair rate allocation. The typical operation of the rate-based control is illustrated in Fig. 1. The details of the RM cell format can be found in [1]. A switch shall implement at least one of the following methods to control congestion at queuing points:

a) Explicit forward congestion indication (EFCI) marking: The switch may set the EFCI state in the data cell headers. Most first-generation switches implemented this mechanism before the RM cell was fully defined.

b) Relative rate marking: The switch may set the congestion indication (CI) bit or the no increase (NI) bit in forward and/or backward RM cells.

c) Explicit rate marking: The switch may reduce the explicit rate (*ER*) field of forward and/or backward RM cells.

The switches that implement options a) and b) are known as binary switches; they can reduce implementation complexity but may result in unfairness, congestion oscillation, and slow congestion response. The switches that implement option

0163-6804/96/$05.00 © 1996 IEEE

c) are generally called *ER* switches and require that sophisticated mechanisms be in place at the switches for the computation of a fair share of the bandwidth. The standard-defined source and destination behaviors, however, allow interoperation of the above three options. Once the source has received permission, it begins scheduling cells for transmission at the allowed cell rate (ACR). The ACR is initially set to the initial cell rate (ICR) and is always bounded between the minimum cell rate (MCR) and the peak cell rate (PCR) specified by the application at call setup. Transmission of data cells is preceded by sending an RM cell. The source will continue to send RM cells, typically after every N_{RM} data cells. The source places the ACR value in the current cell rate (CCR) field of the RM cell, and the rate at which it wishes to transmit cells (usually the PCR value) in the ER field. The RM cells traverse forward through the network, and the destination turns the RM cells around in the backward direction. Intermediate switches on the path notify the source of congestion by marking the EFCI bit in the data cells and the CI or NI bit in the RM cell, and/or reducing the ER value in the RM cells. Upon return of the RM cell, the source should adapt its ACR to the information carried in the RM cell. If CI is not set, the source may linearly increase its ACR by a fixed increment (RIF*PCR), where the rate increase factor (RIF) is determined at call setup. This increase can reach the ER value in the RM cell, but should never exceed the PCR. If CI is set, the source must exponentially decrease its ACR by an amount greater than or equal to a proportion of its current ACR, (RDF*ACR), where the rate decrease factor (RDF) is also determined at call setup. The factors RIF and RDF control the rate at which the source increases and decreases its rate, respectively.

If the ACR is still greater than the returned ER, the source must further decrease its ACR to the returned ER value, although it should never be below the MCR. If NI is set, the source should observe the CI and ER fields in the RM cell, but is not allowed to increase the ACR above its current value. Cell flow regulation allows for end to end, segment by segment, or link by link. Additional operational details are beyond the scope of this article and can be found in [1, 2].

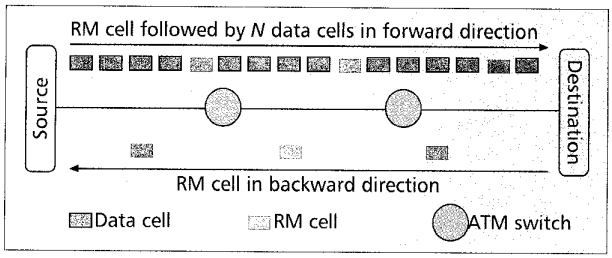


Figure 1. *Rate-based end-to-end congestion control scheme.*

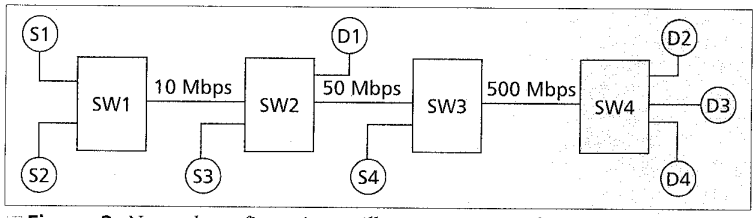


Figure 2. *Network configuration to illustrate max-min fairness.*

Design Criteria for Fair Rate Allocation Algorithms

The method of fairly allocating bandwidth is an area for implementation latitude. This section provides a high-level description of design criteria that switch designers must consider for the design and implementation of an appropriate fair rate allocation algorithm.

Fairness

The most critical component of fair rate allocation is to define a fair rate allocation policy. No set of connections should be arbitrarily discriminated against, and no set of connections should be arbitrarily favored, although resources may be allocated according to a defined policy. A number of fairness policies are possible. A commonly used fairness criterion is *max-min fairness*, introduced in [3]. However, this definition is unambiguous only if no ABR connections receive bandwidth guarantees (MCR = 0). In the case of nonzero MCRs, various fairness criteria exist [4]. In this section, the max-min fairness definition is considered. At any given link in the network, connections that are competing for bandwidth can be grouped into two categories:

Bottlenecked Connections — Those connections unable to achieve their fair (equal) share of bandwidth at the given link because of constraints imposed by their PCR requirements or, most likely, by limited bandwidth available at other links.

Non-Bottlenecked Connections — Those connections whose achievable bandwidths are only limited at the given link (usually referred to as a bottleneck link).

The max-min criterion attempts to equally allocate the available bandwidth among all connections bottlenecked at the link. This principle is fair since all connections that share a link obtain an equal share of link bandwidth provided they can all use that fair share, and the only factor which prevents a connection from obtaining higher allocation is its bottleneck link. Moreover, this principle is efficient in that it maximizes the throughput for all connections that have the minimum allocations in the network. The fair share can be computed as follows:

$$\text{Fair Share} = \frac{C_l - \sum \text{Rates of connections bottlenecked elsewhere}}{N_l - \sum \text{Connections bottlenecked elsewhere}}, \quad (1)$$

where C_l is the capacity (or bandwidth) of link l, and N_l is the number of connections using link l [5]. A simple procedure for finding the max-min fair rate allocations can be formulated iteratively as follows:

1. Find the equal share for the connections on each link.
2. Find the connection(s) with minimum allocated rate.
3. Subtract this rate at the link and eliminate the connections with minimum allocation.
4. Recompute an equal share of each link in the reduced network.
5. Repeat procedures 2–4 until all the connections are eliminated.

As an example, we consider the network configuration shown in Fig. 2. This network consists of four switches connected via three links. The bandwidths of links L1, L2, and L3 are 10 Mb/s, 50 Mb/s, and 150 Mb/s, respectively. Four connection pairs, labeled S1/D1, S2/D2, S3/D3, and S4/D4, are established such that the first link, L1, is shared by two sources, S1 and S2. The second link, L2, is shared by two sources, S2 and S3. The third link, L3, is shared by three sources, S2, S3, and S4.

In order to calculate the fair share, we divide the link bandwidth equally among contending sources on each link. On L1, we allocate 5 Mb/s each to S1 and S2. On link L2, we

Iteration #	S1	S2	S3	S4
1	5	5	25	50
2	5	5	45	72.5
3	5	5	45	100

■ **Table 1.** *Max-min fair allocation rates.*

allocate 25 Mb/s each to S2 and S3. On link L3, we allocate 50 Mb/s each to S2, S3, and S4. As a result, however, S2 cannot use its 25 Mb/s share at link L2 since it is allowed to use only 5 Mb/s at link L1. Therefore, we give 5 Mb/s to S2 and reallocate the bandwidth. Since S2 only uses 5 Mb/s on link L2, we can allocate 45 Mb/s bandwidth to S3. The new available bandwidth on link L3 is 145 Mb/s. We will divide this bandwidth among S3 and S4 equally at 72.5 Mb/s. Since S3 is bottlenecked at link L2, it can only use 45 Mb/s of its fair share of 72.5 Mb/s on link L3. Therefore, we will give S3 45 Mb/s on link L3 and reallocate the bandwidth for S4. The available bandwidth on link L3 is 150 - (5 + 45) Mb/s, or 100 Mb/s. Again, we will allocate this bandwidth to S4. After the above three iterations, the fair allocation vector for this configuration is (5, 5, 45, 100) is shown in Table 1.

As shown in the example, the ideal share of a connection in a given network configuration can be calculated given global knowledge. In practice, however, this information is not readily available, and the fair-rate allocation procedure must be implemented under dynamic network conditions in the absence of centralized knowledge about the network and without the synchronization of diverse network components. Several proposed algorithms are in fact based on the distributed implementation of the max-min criterion, discussed in the next section. Fairness should be maintained regardless of geographic location of the sources, number of hops that will be traversed, and time at which a connection is established.

Convergence

At steady state, a fair allocation algorithm should stabilize to an optimal rate vector from any initial network conditions and without causing large oscillations. Large oscillations generally result in poor link utilization, low throughput, and buffer overflow problems. Therefore, the fair rate allocation algorithm should reduce oscillations around the optimal rate.

Responsiveness

A fair rate allocation algorithm should take the minimum round-trip time to get close to the optimal rate. The ability to provide fast access to the available bandwidth and rapid rate reductions under congestion are some of important features for the design and implementation of a fair rate allocation algorithm.

Robustness

A fair rate allocation algorithm should operate correctly in dynamic load changes even in the presence of sudden traffic changes, network failures, and parameter mistunings. For those algorithms that rely on the tuning of a number of parameters in calculating the explicit rate, any parameter that has not been set correctly may lead to potential performance degradation. It may be desirable to design an algorithm that has few parameters to be tuned dynamically. Moreover, the rate increase and decrease parameters (i.e., RIF and RDF) should not dictate the performance of the algorithm.

Scalability

The network in the future is bound to grow in size and distance. It is imperative that any fair rate allocation algorithm scale with the network growth (i.e., distance, speed, number of connections, number of switches). The performance of the algorithm should be analyzed under these constraints.

Interoperability

A fair rate allocation algorithm should operate effectively in a multivendor environment where various fair rate allocation algorithms exist. There is an existing base of deployed switches, and any introduction of new fair rate allocation algorithms should interoperate with existing switches and should not dictate the switch implementation. This area deserves extensive studies in the future.

Implementation Complexity

Significant complexity arises when explicit rate computations have to be performed on a per-connection basis. The required computations for a fair rate allocation algorithm should be kept to a minimum such that it can be implemented at a reasonable cost added to the switch design. As an example, consider a switch operating at a 155 Mb/s port rate. In this case, the switch must process and schedule each RM cell within 2.75 μs. When the switch port rate increases to 622 Mb/s or even up to 2.4 Gb/s, the corresponding cell time will become 0.68 μs or 0.18 μs.

The increased link speed is in favor of simple fair rate allocation algorithms that can be implemented in hardware. Furthermore, any requirement for floating point operations may increase the hardware complexity.

Early implementation of congestion control was based on binary feedback information where switches only mark a single congestion bit in the data cells in the event of congestion. The inherent shortcomings of the binary approach has recently led to the consideration and implementation of sophisticated fair rate allocation algorithms that compute fair rates for each connection. The current ABR congestion control specification gives considerable freedom to the switch vendors in designing and developing fair rate allocation algorithms. A number of fair rate allocation algorithms have been proposed and studied in the literature. These algorithms can be broadly classified into two categories: algorithms that approximate the fair share [6–9]), and algorithms that calculate the exact fair share [5, 10–12]).

Before proceeding to describe these algorithms, we will first examine the binary schemes, and discuss their associated problems and possible enhancements.

Binary Schemes

Early switches implemented binary schemes mostly based on a simple first-in first-out (FIFO) queuing mechanism shared by all ABR connections. The simplest example of such a switch is to mark the EFCI bit in data cell headers when congestion is detected. Congestion detection can be based on variations of queue length information such as a single threshold or dual thresholds. This arrangement could cause some connections to have unfairness access to the available bandwidth depending on the network topology and end system behavior. This unfairness problem can be best illustrated by considering the network configuration shown in Fig. 3. This configuration is known as generic fairness configuration 1 (GFC1). The GFC1 network consists of five switches and 23 connections grouped into six groups (A–F). In Fig. 3, the number inside the parentheses next to the group label represents the number of connections for that group. Connections in groups C, D, E, and F traverse a single hop, and those in groups A and B traverse three hops. Connections in both group B and group E share the same link between switch 4 (SW4) and SW5. A source in group E is closer to the shared link than that in group B. Given that all switches implement a

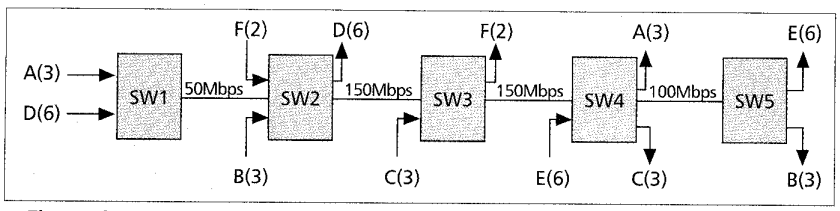


Figure 3. *Generic fairness configuration 1.*

simple FIFO queuing mechanism and mark the EFCI bit in the event of congestion, this network configuration could result in unfairness among the groups. This becomes evident from the simulation results plotted in Fig. 4, where the actual bandwidth share as a percentage of the desired bandwidth share based on the max-min criterion for each group is given. As expected, group E has more shared bandwidth than group B, for example. In general, groups A and B, which compete for bandwidth over multiple links, receive a lower max-min fair share, while other groups take advantage and receive a higher fair share. The observation indicates that connections traveling more hops have a higher probability of getting their cells marked than those traveling fewer hops. As a result, it is unlikely that the long-hop connections are able to increase their rates, and consequently they are beaten down by the short-hop connections. This is thus known as the *beat-down problem* in the literature. There are several alternatives that can alleviate the unfairness problem in binary feedback schemes.

The first alternative is to provide each connection or a group of connections with its own separate queue. Such a per-connection queuing mechanism makes it possible to determine the congestion status of a connection in isolation from the rest of the connections. Because only those connections that cause the congestion of their associated queues are subject to congestion marking, the beat-down problem in the basic binary scheme can be eliminated. Thus, this isolation ensures fair access to buffer space and bandwidth among all competing connections.

This per-connection queuing approach also enjoys several other distinct benefits. First, per-connection queuing can be used to help protect individual connections effectively since a misbehaving connection can only cause its own queue to overflow. Second, it allows the delay and loss behavior of individual connections to be decoupled from each other, leading to better QoS control. Finally, per-connection information is readily available to help the implementation of other types of congestion control, such as early packet discard mechanisms.

Although per-connection queuing offers considerable advantages over single FIFO queuing, this comes at the significant cost of switch design and implementation. Efficient buffer management and scheduling among the queues can be very costly to design and implement. Such implementation may be prohibitive when the number of connections grows substantially large.

The second feasible alternative is to provide selective or intelligent marking on a common FIFO queuing mechanism when congestion takes place. In this alternative, a switch computes a fair share and a connection is asked to decrease its rate by marking the EFCI bit only if its current cell rate is above the computed fair share. Examples of ways in which a switch can compute a fair share will be presented in the following sections. In addition to EFCI marking, CI and NI bits in the RM cells can be used to regulate the rate at the source. Although this approach can provide a good solution for the beat-down problem, it is generally slow in response to congestion because it may take several round-trips to alleviate congestion or utilize any unused bandwidth. Another problem that has been found is that it may generate large oscillations even at steady state, making it difficult to achieve very high link utilization.

Another alternative is to replace binary switches with ER switches that explicitly compute the fair share and ask sources to transmit at this fair share rate. This alternative is generally considered as a powerful means to make ABR work in a fair and effective manner. The central issue of how to design such a fair rate allocation algorithm is the subject of the next sections. Before we proceed further, the natural question to ask is whether intelligent marking or an ER switch can work well without per-connection queuing. We argued in [13] that explicit connection-based control can implicitly achieve the same effect as the per-connection queuing implementation. In the context of ABR traffic, since the rate-based scheme implicitly controls congestion on a per-connection level, a single-FIFO queue with some intelligent marking or explicit rate setting can achieve almost the same performance as the per-connection queuing approach with regard to fairness, throughput, and link utilization. Furthermore, intelligent marking or ER setting is considerably less expensive to realize in hardware than a per-connection queuing alternative.

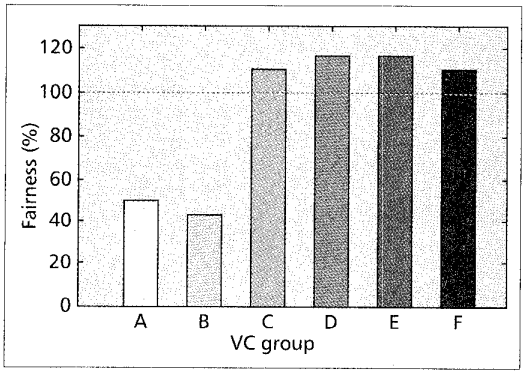


Figure 4. *The beat-down problem.*

Approximate Fair Rate Calculation Algorithms

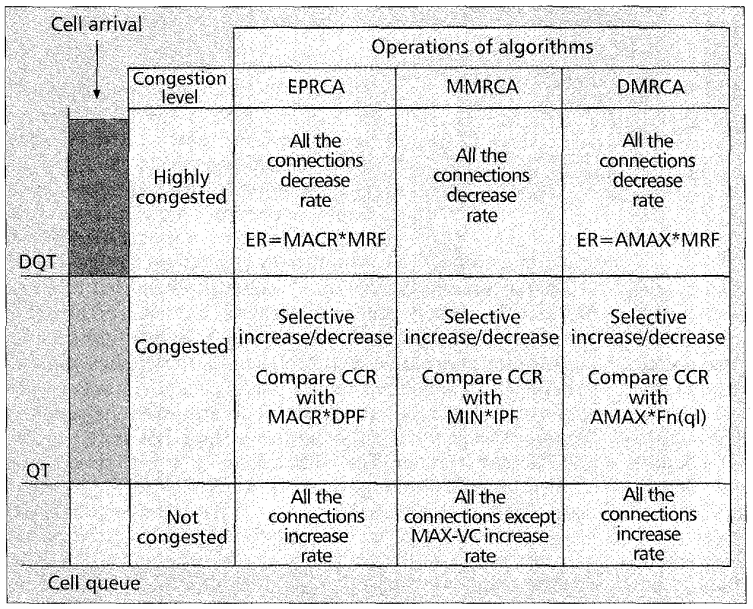
Given the max-min fairness definition in Eq. 1, there are generally two ways to compute the fair share: approximation and exact computation. In this section, we will discuss proposed algorithms that fall into the first category. Algorithms in this category tend to approximate the fair share rate derived from queue length information in conjunction with the CCR value available in RM cells. A switch maintains a running average of fair share, based on the level of congestion and CCR, in the hope that at the steady state this value reflects an approximate fair rate at the switch.

Enhanced Proportional Rate Control Algorithm

The enhanced proportional rate control algorithm (EPRCA) in [6, 14] was proposed to perform intelligent marking in order to achieve fairness. In this algorithm all the connections are assumed to share a common queue with two congestion thresholds QT and DQT. For every forward RM cell received, the switch computes a mean allowed cell rate (MACR) using an exponential running average as follows:

$$MACR = (1 - \alpha)\, MACR + \alpha\, CCR, \quad (2)$$

where α is the averaging factor and is generally chosen to be 1/16. The switch estimates the average of the CCR values of connections that are not bottlenecked elsewhere by keeping track of the



■ **Figure 5.** *A comparison of EPRCA, MMRCA and DMRCA.*

queue length. When the queue length exceeds the threshold QT, congestion is declared, and only those connections whose CCR values are above the fair share computed as DPF*MACR are asked to reduce their rates, while the rates for the rest of connections are allowed to increase. The parameter DPF, typically set to 7/8, is known as the *down pressure factor*, and is introduced to reduce the rates of those connections whose rates are very close to MACR, thus avoiding the potential oscillation. When the queue length exceeds the threshold DQT, the switch is said to be in a state of severe congestion. In this case, all the connections are asked to reduce their rates regardless of their CCR values. The switch uses the fair share to perform intelligent marking. In addition, if the value in the ER field of a backward RM cell is above the fair share and the switch is in a congested state, the ER value is lowered to its fair share; otherwise, no adjustments need to be made.

The most obvious advantage of this algorithm is low implementation complexity. To make this scheme work effectively in practice, however, CCR values and MACR must converge under all conditions. To accomplish this, the algorithm has used several multiplier factors to enforce the convergence. Unless these parameters are tuned properly, the algorithm could exhibit severe oscillations, and may not converge to the desired fair rate. Considerable unfairness could persist when the estimation of MACR is not close to the fair share to which the CCR values of the connections are supposed to converge. This situation could happen when some connections bottlenecked elsewhere in other switches cause underestimation of the fair share, when transient behaviors cause rate oscillations, and when the sources are not well behaved and/or insert incorrect CCR values.

Various enhancements to this algorithm have been proposed that could overcome the above-mentioned problems to a certain extent at additional cost. Figure 5 gives a summary of EPRCA operations and the two enhancements described below.

Max-Min Rate Control Algorithm (MMRCA)

One possible enhancement to the EPRCA is the MMRCA algorithm proposed in [7]. In this algorithm, the switch maintains the minimum rate (MIN) and the maximum rate (MAX) of all the connections and their corresponding connection numbers, recorded as MAX-VC and MIN-VC, respectively. These rates, along with the two queue thresholds QT and DQT, are used to approximate the fair rate.

When the queue length falls below the threshold QT, all the connections are allowed to increase their rates except the MAX-VC connection. Similar to the EPRCA, the switch is considered to be in a state of congestion when the queue length exceeds the threshold QT. However, congestion is indicated to only those connections whose CCR values are above a value of IPF*MIN, while the rates for the rest of the connections are allowed to increase.

The parameter IPF is known as *increase pressure factor*, and serves a similar purpose to the DPF used in the EPRCA. If the difference between the MAX and MIN values is smaller than a predefined value, no connections are allowed to increase their rates. Similar to the EPRCA, all the connections are asked to decrease the rates if the queue length exceeds the DQT threshold.

It is shown in [8] that when the MAX and MIN rates do not quickly converge to the fair rate, the algorithm can potentially lead to unfairness. An algorithm to solve this problem has recently been proposed, and is presented below.

Dynamic Max Rate Control Algorithm

Similar to the EPRCA, the dynamic max rate control algorithm (DMRCA), proposed in [8], uses two queue length thresholds for congestion detection.

Like the MMRCA, the switch monitors the maximum rate (MAX) of all connections arriving at the switch, and records the connection number corresponding to the MAX value as MAX-VC. In order to smooth out the oscillations due to dynamic changes of the MAX value, the switch maintains an exponential running average on the average maximum rate (AMAX) which is computed as follows,

$$\text{AMAX} = (1 - \alpha)\,\text{AMAX} + \alpha\,\text{MAX}. \quad (3)$$

Again, α is the averaging factor. It should be noted that the above operation is only performed whenever a new MAX value has been detected.

When the queue length exceeds the threshold QT, the switch performs intelligent marking based on a marking threshold. This marking threshold, expressed below, is calculated as a function of the queue length and the AMAX,

$$\text{MT} = \text{AMAX} * Fn(\text{Queue Length}), \quad (4)$$

where the function Fn(Queue Length) is a discrete, nondecreasing function of the queue length ($0 \le Fn \le 1$).

When the queue length is somewhere between thresholds QT and DQT, the CCR field in the RM cell is compared against MT, and the outcome of this comparison is used to selectively indicate congestion. It was reported that to use the ER setting when the queue length stays in this region can sometimes result in poor performance [8]. When the queue length is above the threshold DQT, the ER marking is used to reduce rates rapidly. The ER field in the RM cell is set equal to AMAX*MRF, where MRF stands for the *major reduction factor.*

The DMRCA algorithm has low implementation complexity and generally converges to the fair rate. The intelligent marking threshold, defined to be a function of the queue length, results in better control of buffer usage. The algorithm is also proved to be insensitive if the CCR values are not set correctly. Special care has to be taken, however, to tune some of the parameters, such as RIF and RDF. The algorithm has been shown to exhibit very large rate oscillations if RIF is set

too high, and sometimes can lead to potential unfairness. For the algorithm to work effectively and correctly, it seems necessary to choose a very small RIF value.

Congestion Avoidance Using Proportional Control (CAPC)

The CAPC algorithm proposed in [9] is a congestion avoidance scheme in which the rate of change of queue length is used as a load indicator. In this algorithm, the switch computes a load factor, *LF*, which is defined as a ratio of the actual input rate and a predetermined target rate for the link, and it is given as below:

$$LF = \frac{\text{Input Rate}}{\text{Target Rate}}. \quad (5)$$

The input rate is measured over a fixed averaging interval and the target rate is set slightly below the link bandwidth (e.g., 85–90 percent). The load factor is used to update the fair share which is computed differently depending on the value of the load factor. A load factor less than 1 indicates that the queue is underloaded, whereas a load factor greater than 1 indicates that the queue is overloaded.

During underload, the fair share is increased according to

$$\text{Fair Share} = \min(ERU, 1 + (1 - LF)\, Rup)\ (\text{Fair Share}), \quad (6)$$

where *Rup* is a slope parameter between 0.025 and 0.1, and ERU determines the maximum allowed increase of the fair share. Accordingly, during overload, the fair share is decreased and updated as

$$\text{Fair Share} = \max(ERF, 1 - (LF - 1)Rdn)\ (\text{Fair Share}), \quad (7)$$

where *Rdn* is a slope parameter between 0.2 and 0.8, and *ERF* is the minimum allowed decrease and is chosen to be 0.5. If the calculated fair share is lower than the ER value in the RM cell, then the ER field in the RM cell is set to the fair share. Similar to the EPRCA, the CAPC algorithm does not require maintenance of any variables on a per-connection basis. The parameters required to force convergence, however, must be set carefully; incorrect settings could cause large oscillations in rates and potential unfairness. Similar to other approximate algorithms, the CAPC is sensitive to the choice of RIF and RDF values.

Exact Fair Rate Calculation Algorithms

The algorithms that fall in this category, as the name implies, attempt to directly compute the fair share given in Eq. (1) one way or another in a distributed fashion, based on the available bandwidth and per-connection state information.

In order to perform this calculation, the switch must maintain a connection-based table to collect and store per-connection information. The use of per-connection information presents an opportunity to make aggressive rate increases and generate an oscillation-free steady state. The algorithms differ in the complexity of implementation in terms of required storage memory and number of floating point divisions required.

Congestion Control with Explicit Rate Indication (CCERI)

The algorithm presented in [5] was an early proposal to compute explicit rate in a distributed manner, originally formulated in the context of packet switching. The development of this algorithm has had a significant influence on the fair rate allocation algorithms for ABR service. At the time of its development, much of the ABR specification did not exist. Thus, many of the features available now in RM cells were not utilized in its design.

In this algorithm, each switch monitors its traffic and calculates its available capacity per flow or per connection. This quantity is called the *advertised rate*. The switch keeps track of the number of bottlenecked connections at the switch and the last seen ER values. When an RM cell arrives at the switch, if its ER is less than the advertised rate, then the associated connection is assumed to be bottlenecked elsewhere. A bottleneck bit is marked and the current rate of the connection is stored in a connection table. At each iteration, Eq. (1) is used directly to compute the advertised rate. If at any time a connection previously marked transmits at a rate larger than the advertised rate, the corresponding bottleneck bit is then unmarked, and the advertised rate is recalculated.

The algorithm is proved to converge to the optimal max-min rate from any initial conditions, and the convergence time is upper-bounded by $4M$ round-trip times, where M is the number of bottleneck links in the network. The steady state bandwidth utilization of this algorithm does not oscillate, and has a fast transient response. The algorithm however requires per-connection information for each of the established connections at the time of advertised rate computation. Therefore, the algorithm has a computational complexity of $O(n)$, where n is the number of active connections.

The computation time would be significant when n grows larger. Moreover, this algorithm requires that all the switches along its path execute the same algorithm, thus preventing it from inter-operating with any other switches that use a different fair rate allocation algorithm. This requirement is due to the fact that the advertised rate calculation is primarily based on the ER field in the RM cell without any added intelligence. Consequently, if a simple switch that does not mark the ER field exists in the network, the congestion information from the simple switch will be ignored.

In order to overcome the computational complexity, a quantized (discrete) rate allocation process is presented in [15]. This scheme requires that sources be allowed to send cells at a rate chosen from a discrete set of possible transmission rates. Thus, switches are only needed to store these discrete values, thus reducing the complexity significantly. For this new scheme to work effectively, it is required that all the sources be aware of this discrete set of rates, and that all the switches in the network are implemented with the same algorithm.

Explicit Rate Indication for Congestion Avoidance (ERICA)

The ERICA algorithm proposed in [10] is another variation of congestion avoidance schemes. This algorithm calculates two quantities: fair share and this connection's (VC's) share. Similar to the CAPC, it computes a load factor based on the input and target rates. Utilizing the load factor LF, the VC share is calculated as

$$\text{VCshare} = \frac{\text{CCR}}{LF}, \quad (8)$$

where CCR is derived from the forward RM cell received to ensure that the most current information is used to provide fast feedback.

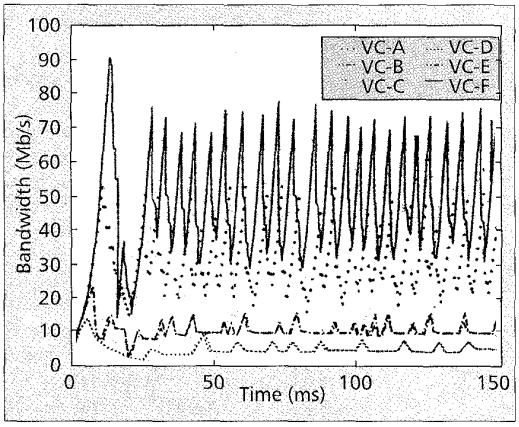
In order to achieve fairness, this algorithm allows connections to increase their rates to a fair share during underload. The fair share is calculated by the following formula:

$$\text{Fair Share} = \frac{\text{Target Rate}}{N} \quad (9)$$

where N is the number of active connections. Upon reception of a backward RM cell, its *ER* field is marked down as follows:

$ER = \min\{ER, \max(\text{Fair Share}, \text{VCshare})\}$ (10)

The ERICA algorithm can operate in a congestion-avoidance state, is insensitive to parameter variations, and proves to be very robust. The rates converge very quickly and with little oscillation. This algorithm, however, has some fundamental limitations in achieving desired fairness for all connections and buffer requirements. In some cases, a connection that gets started late gets its equal link share, but does not get the max-min fair rate. Furthermore, during transient periods, and if the desired target utilization is set close to the full link rate, the queue grows rapidly and results in heavy cell loss. Although not discussed here, there exist many extensions and modifications of this algorithm. These extensions, with added complexity, try to eliminate some of the problems found in the basic ERICA algorithm. For complete details, the reader is referred to [16].



■ **Figure 6.** *Instantaneous bandwidth utilization in GFC1 network: EPRCA algorithm.*

Efficient Rate Allocation Algorithm (ERAA)

The algorithm presented in [11] attempts to solve the computational complexity of the CCERI algorithm. Similar to that algorithm, the switch maintains per-connection information such as bottleneck state and bottleneck bandwidth of each connection. The switch calculates an equal share bandwidth (EQB) which is given by

$$\text{EQB} = \frac{\text{Available bandwidth}}{\text{Total number of connections}}. \quad (11)$$

When a connection does not use the its equal share bandwidth, the connection is considered to be bottlenecked elsewhere. The sum of free bandwidth not used by the bottlenecked connections is referred to as the *free bandwidth* (FB). A portion of the FB is allocated to the nonbottlenecked connections in addition to their EQB, and this new rate is referred to as A_{MAX}. A complete description and pseudocode of this algorithm can be found in [17].

The calculation of A_{MAX} occurs when a forward RM cell arrives. In order to perform the calculations in $O(1)$ time, only the change in the connection state is considered. It is important to note that the switches compute A_{MAX} based on the minimum of CCR and ER values received. This makes it possible for this algorithm to interoperate with other algorithms. The algorithm in steady state converges to max-min fair rates quickly, while being efficient and stable.

The ERAA algorithm, however, assumes that the exact transmission rate is carried in the CCR field. If a connection is bursty or idle for long time, the CCR field in the RM cell may not carry the correct value. The inaccuracies in CCR value may present problems for ERAA to function efficiently. There are no mechanisms in ERAA to modify the rate allocations to reflect the CCR inaccuracies, and further study is required to address this problem.

Fast Max-Min Rate Allocation (FMMRA) Algorithm

This algorithm is based on measurement of available capacity and exact calculation of fair rates. The goals of this algorithm are to reduce the computational complexity imposed by the CCERI algorithm discussed earlier, and at the same time make the algorithm interoperable. An additional important feature of this algorithm is that it is not sensitive to inaccuracies in CCR values.

The algorithm can be executed by any switch component experiencing congestion (input port, output port, etc.). Each ABR queue in the switch computes a rate it can support. This rate is referred to as the *advertised rate*, γ. The advertised rate along with the ER field in the RM cell are used to determine if the connection is bottlenecked elsewhere. If a connection cannot use the advertised rate, it is marked as bottlenecked elsewhere and its bottleneck bandwidth is recorded.

The ER field in the RM cell is read and marked in both directions to speed up the rate allocation process. The bidirectional ER marking in this algorithm makes it possible for downstream switches to learn bottleneck bandwidth information of upstream switches, and upstream switches to learn bottleneck bandwidth information of downstream switches. Many of the proposed algorithms mark the ER field only in the backward direction. Because of the unidirectional ER marking, the switches closer to the source get more accurate ER information than those closer to the destination. As a result, this may result in slower response to congestion. This bidirectional updating of *ER* in the RM cell plays a significant role in drastically reducing the convergence time of the max-min fair rate allocation process.

In contrast to the algorithm presented in [5], which requires the switch to inspect the states of all the connections in the connection table when calculating the fair rate, the FMMRA algorithm only considers the changes in the per-connection variables. In addition, it only requires knowledge of the connections seen by the switch at the time of update. When a backward RM cell is received, the advertised rate is recomputed by incorporating the bottleneck state of the received connection. This feature reduces the computational complexity of this algorithm to be $O(1)$.

This algorithm does not use the CCR carried in the RM cells; instead, it uses the load factor and the ER to compute an exponential running average of the maximum value of ER, denoted as ER_{max}. This operation is shown below:

$$ER_{\max} = (1-\alpha)ER_{\max} + \alpha\max\left(ER, \frac{ER_{\max}}{LF}\right), \quad (12)$$

where α is again an averaging factor and can be set to 1/8. The computation of ER_{max} is done in the backward direction, and reflects the advertised rate after taking the load into consideration. The load factor reflects how well the ABR bandwidth is utilized. For example, LF < 1 indicates that some of connections are sending cells at rates less than their allowed rates. This presents an opportunity for the nonbottlenecked connections in the link to increase their rates.

Based on the level of congestion, which is determined as a function of the queue length and the load factor, the ER field in the RM cell (both forward and backward) is updated according to

$$ER = \min\{ER, \max(\gamma, (1-\beta)\, ER_{max})\}, \quad (13)$$

where β is a single bit value indicating the connection is bottlenecked elsewhere. The use of the advertised rate along with

ER_{max} help the fair rate to converge faster. They also help to adjust the fair share according to load variations in the network, in order to achieve full link utilization. Moreover, the added intelligence provided by the use of LF and ER_{max} makes it possible for this algorithm to interoperate with other algorithms.

Selective measures are taken in updating the ER field in order to control the queue growth to prevent potential cell loss. If the queue length reaches a low threshold QT and LF > 1, only the advertised rate is used in marking the ER field in the RM cell, as shown below.

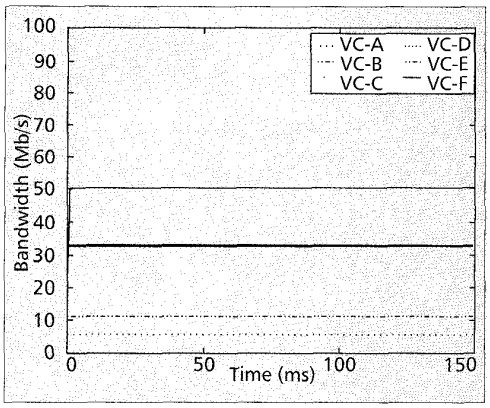
$$ER = \min(ER, \gamma), \quad (14)$$

Under severe congestion, ER_{max} is set to the advertised rate. This ensures that whenever the potential for congestion is detected, even if some connections are idle, the nonidle connections are not given any extra bandwidth, allowing the queue to drain. Furthermore, if the queue length exceeds a high threshold DQT (indicating heavy congestion), the target utilization factor is reduced by a target rate reduction factor (TRRF) until the queue length drops below the low threshold QT. This technique allows the queue to drain and operate at a desired queue length whenever the switch is heavily congested, for example, when many new connections are established.

Simulation Results

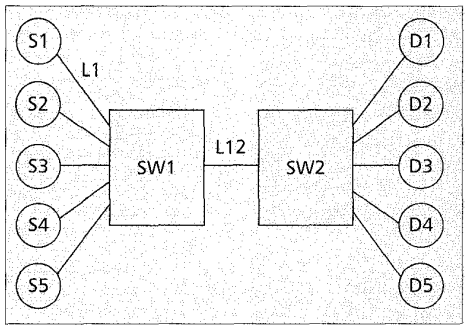
In order to compare the performance of some of the algorithms we have discussed so far, we have simulated these algorithms for the network configuration GFC1 shown in Fig. 3. Traffic sources are assumed to be well behaved and persistently greedy, and to always transmit at the maximum allowed cell rate. The use of persistent sources presents a tough challenge for multiple congested links, and helps to illustrate the concept of fairness easily. The expected max-min fair allocation rates for the connections from groups A–F are 5.56, 11.11, 33.33, 5.56, 11.11, and 50 Mb/s, respectively. In order to fully understand the performance of these algorithms, simulations should also be done for bursty sources and background traffic, although these results are not presented here due to space limitations.

The first simulation is done to show the problems associated with those algorithms that do not take into consideration per-connection information and which use approximation techniques to derive a fair rate. To illustrate this, the EPRCA algorithm is used as the switch algorithm in all the switches. From Fig. 6 it can be seen that, although the EPRCA converges to fair rates, significant levels of oscillation are found at steady state. It was observed that the rates of connections receiving higher bandwidth have much larger oscillations than connections receiving lower bandwidth. Note that in this simulation the RIF value is set such that sources increase the allowed cell rate by 3.2 Mb/s every time an RM cell arrives at the source. However, in a LAN environment it may be desired to allow the sources to increase their rates to their fair share almost instantly by using a higher value of RIF. This can increase ACR up to PCR or ER instantly. It was observed that using a higher value of RIF with the EPRCA results in poor convergence, very large oscillations, and poor link utilization.

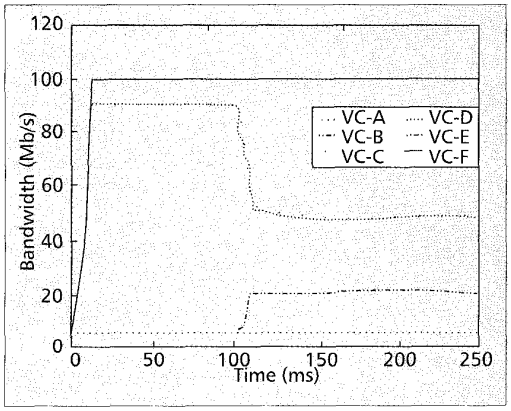


■ **Figure 7.** *Instantaneous bandwidth utilization in GFC1 network: FMMRA algorithm.*

Moreover, the selection of threshold values is also very important to achieve good performance. Improper selection of threshold values results in poor link utilization, especially in a WAN configuration. Thus, the EPRCA significantly limits the network to operate aggressively (i.e., using large RIF) and to be robust (i.e., tuning parameters). Similar problems are experienced with the MMRCA, DMRCA, and CAPC algorithms, which use many parameters to force convergence. It was observed that the parameters RIF and RDF had to be retuned with varying network topologies and the number of connections involved. On the other hand, the algorithms based on exact computation converge very fast with no oscillations at steady state, regardless of RIF and RDF values (Fig. 7). The freedom to use a high RIF allows fast convergence and instant bandwidth usage. To show the convergence problem experienced in the ERICA algorithm, a network topology (Fig. 8) is simulated using the ERICA algorithm. The network consists of five connections and two switches. The distances of links L1 and L12 are 1000 km and 100 km, respectively. All the links have 100 Mb/s capacity and a propagation delay of 5 μs/km. We consider the situation where some connections are bottlenecked. We make connections 1 and 3 bottlenecked by setting the PCR value of the sources S1 and S3 to 5 Mb/s. Since connections 2 and 4 are not active until t = 100 ms, connection 2 can use 90 Mb/s on link L12. At any time t > 100 ms, connections 2, 4, and 5 should receive equal shares of bandwidth on link L12. Thus, the steady-state bandwidth share for connections 1 and 3 is 5 Mb/s,



■ **Figure 8.** *The single-hop network.*



■ **Figure 9.** *Instantaneous bandwidth utilization in single-hop bottleneck LAN configuration – ERICA algorithm.*

Algorithm	Storage Complexity	Computations Complexity	Floating Point Division	Sensitive to RIF/RDF	Fairness	Responsiveness	Sensitive to CCR Errors
EPRCA	O(1)	O(1)	No	Yes	Not always	Slow	Yes
MMRCA	O(1)	O(1)	No	Yes	Not always	Slow	Yes
DMRCA	O(1)	O(1)	No	Yes	Mostly	Slow	No
CAPC	O(1)	O(1)	No	Yes	Not always	Slow	Yes
CCERI	O(N)	O(N)	Yes	No	Always	Fast	-
ERICA	O(N)	O(1)	Yes	No	Not always	Fast	No
ERAA	O(N)	O(1)	Yes	No	Always	Fast	Yes
FMMRA	O(N)	O(1)	Yes	No	Always	Fast	No

■ **Table 2.** *Summary of fair rate allocation algorithms.*

and for connections 2, 4, and 5 is 30 Mb/s. From Fig. 9, it can be seen that the ERICA algorithm fails to converge to the fair rate for both LAN and WAN environments. Connections that start late do not get their fair share of 30 Mb/s, and instead get a share of 20 Mb/s.

Summary and Future Directions

This article has discussed some of the key issues in the design and implementation of fair rate allocation algorithms for ABR service in ATM networks. A number of proposed fair rate allocation algorithms are described and discussed. A summary of these algorithms with regard to performance and complexity is tabulated in Table 2. The table highlights the general assessment of the algorithms, and does not provide a complete description. Performance may vary with varying network conditions. This list of the proposed algorithms is certainly not complete, and new algorithms are bound to be invented in the future. The selection of a fair rate allocation algorithm for a switch design and implementation is largely dependent on the appropriate balance between performance and implementation complexity. Equally challenging is the design of algorithms that can meet an appropriate set of design objectives.

There are a number of ways that further research work on the fair rate allocation discussed in this article can be extended. The first area that deserves immediate attention is the impact of nonzero minimum cell rate on the existing algorithms. For instance, interesting work has been done in [4, 18] to extend the fairness definition to cases with a nonzero minimum cell rate requirement in a network environment. It is also shown as an example how an existing fair rate allocation algorithm can be modified by taking into account this new requirement.

Another area that has not been extensively addressed is the interoperability of these algorithms in a multivendor switch network. It may be that each individual algorithm can work well on its own merit, but may fail in such an interoperational environment. Last but not least, few of the proposed algorithms have been studied and simulated on such a large scale as today's Internet. As the ATM network is bound to grow in the future, it is essential to ensure that these algorithms can truly scale.

References

[1] The ATM Forum, "The ATM Forum Traffic Management Specification," v. 4.0, 1996.

[2] F. Bonomi and K. W. Fendick, "The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service," *IEEE Network*, Mar.-Apr. 1995, pp. 25–39.

[3] J. M. Jaffe, "Bottleneck Flow Control," *IEEE Trans. Commun.*, vol. 29, July 1981, pp. 954–62.

[4] N. Yin, "Fairness Definition in ABR Service Model," ATM Forum cont. 94-0928R2, Sept. 1994.

[5] A. Charny, D. Clark, and R. Jain, "Congestion Control with Explicit Rate Indication," *Proc. ICC '95*, June 1995.

[6] L. Roberts, "Enhanced PRCA (Proportional Rate-Control Algorithm)," ATM Forum cont. 94-0735R1, Aug. 1994.

[7] S. Muddu *et al.*, "Max-Min Rate Control Algorithm for Available Bit Rate Service in ATM Networks," *Proc. ICC '96*, June 1996.

[8] F. Chiussi, Y. Xia, and V. P. Kumar, "Dynamic Max Rate Control Algorithm for Available Bit Rate Service in ATM Networks," to be presented at GLOBECOM '96, 1996.

[9] A. W. Barnhart, "Explicit Rate Performance Evaluations," ATM Forum cont. 94-0983, Sept. 1994.

[10] R. Jain S. Kalyanaraman, and R. Viswanathan, "A Sample Switch Algorithm," ATM Forum cont. 95-0178R, Feb. 1995; available at http://www.cis.ohio-state.edu/~jain.

[11] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, "An Efficient Rate Allocation Algorithm for ATM Networks Providing Max-Min Fairness," *Proc. 6th IFIP Int'l. Conf. High Performance Networking*, Sept. 1995, pp. 143–54.

[12] A. Arulambalam, X Chen, and N. Ansari, "A New Fair-Rate Allocation Algorithm for Available Bit Rate Service in ATM Networks," Tech. Rep. CCSPR-96-WAS, Electrical and Computer Eng., NJ Inst. Tech., Feb. 1996.

[13] A. Arulambalam, X. Chen, and N. Ansari, "Impact of Queuing Disciplines on Available Bit Rate Congestion Control in ATM Networks," *Proc. 30th Annual Conf. Info. Sci. and Sys.*, Mar. 1996, pp. 433–38.

[14] K. Siu and H. Tzeng, "Intelligent Congestion Control for ABR Service in ATM Networks," *ACM SIGCOMM, Comp. Commun. Rev.*, Oct. 196, pp. 81–106.

[15] A. Charny, K. K. Ramakrishnan, and A. G. Lauck, "Scalability Issues for Distributed Explicit Rate Allocation in ATM Networks," *Proc. INFOCOM '96*, Mar. 1996, pp. 1198–1205.

[16] R. Jain *et al.*, "ERICA Switch Algorithm: A Complete Description," ATM Forum cont. 96-1172, Aug. 1996.

[17] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, "Dynamics of an Explicit Rate Allocation Algorithm for Available Bit Rate Service in ATM Networks," Tech. Rep. UCSC-CRL-95-54, Univ. of CA, Feb. 1996.

[18] Y. T. Hou, H. Tzeng, and V. P. Kumar, "On Fair Rate Allocation Policies with Minimum Cell Rate Guarantee for ABR Service in ATM Networks," submitted to INFOCOM 97.

Biographies

AMBALAVANAR ARULAMBALAM [M '96] received his B.S., M.S., and Ph.D. degrees in electrical engineering from New Jersey Institute of Technology, Newark, in 1992, 1993, and 1996, respectively. In June 1996, he joined Bell Laboratories at Murray Hill, New Jersey as a member of technical staff. His research interests include congestion and flow control, ATM network and switch performance evaluation, and the application of neural networks in telecommunications.

XIAOQIANG CHEN [M '92]received the B.Eng. and M.Eng. degrees in electrical engineering from the Nanjing Institute of Communications Engineering, Nanjing, China, in 1982 and 1985, and the Ph.D. degree in computer science from the University of Cambridge, England, in 1992. Since 1992, he has been with Bell Laboratories, where he is currently a member of technical staff in the ATM Networks Research Department.

NIRWAN ANSARI [SM '94] received the B.S.E.E. (summa cum laude) from NJIT in 1982, the M.S.E.E. from the University of Michigan in 1983, and the Ph.D. degree from Purdue University in 1988. In 1988, he joined the ECE Department of NJIT, where he is an associate professor and assistant chair for graduate studies. His current research interests include ATM networks, distributed and adaptive detection in CDMA, data fusion, computational intelligence, and nonlinear signal processing.