Overload Based Explicit Rate Switch Schemes with MCR Guarantees [1]

Bobby Vandalore, Sonia Fahmy, Raj Jain, Rohit Goyal, Mukul Goyal
The Ohio State University
Department of Computer and Information Science
Columbus, OH 43210-1277
Phone: 614-688-4482, Fax: 614-292-2911
E-mail: {*vandalor, fahmy, jain, goyal, mukul*}*@cis.ohio-state.edu*

Abstract

An explicit rate switch scheme monitors the load at each link and gives feedback to the sources. We define the overload factor as the ratio of the input rate to the available capacity. In this paper, we present three overload based switch schemes which provide MCR (minimum cell rate) guarantees for the ATM (asynchronous transfer mode) ABR (available bit rate) service. The switch schemes proposed use the overload factor and other terms including current source rate and target utilization to calculate feedback rates. A dynamic queue control mechanism is used to achieve efficient usage of the link, control queues and, achieve constant queuing delay at steady state. The proposed algorithms are studied and compared using several configurations. The configurations were chosen to test the performance of the algorithms in presence of link bottlenecks, source bottlenecks and transient sources. Finally, a comparison of the proposed algorithms based on the simulation results is given.

I. INTRODUCTION

The ATM (asynchronous transfer mode) is the chosen technology for implementing B-ISDN (broad-band integrated services digital network). ATM is a connection-oriented cell switching standard. It uses fixed size cells which are 53 bytes long. ATM offers different service categories for transporting different types of traffic. The ABR (available bit rate) service category in ATM is used to transport data traffic with minimum rate guarantee. The ABR users specify minimum rate using MCR parameter during connection setup. The ABR service gives guarantee that the ACR (allowed cell rate) is never less than MCR. ABR uses closed loop feedback to control the source rates (see Figure 1). The source sends periodically (after every $Nrm - 1$ data cells) an RM (resource management) cell to gather information from the network [1]. The RM cells are turned around at the destination. The switches along the path indicate the current rate which they can support in the explicit rate (ER) field of the RM cell. When the source receives the backward RM cells, it adjusts its allowed rate accordingly.

The specification of the ABR feedback control algorithm (switch scheme) is not yet standardized. An early switch algorithm used binary feedback to advise source about congestion information [2]. Distributed algorithms for the congestion control using explicit rate feedback were given in [3, 4]. Improved, simpler distributed algorithms were proposed in [5, 6, 7, 8, 9, 10]. Recently, a generalized definition of max-min fairness and its distributed implementation were presented in [11, 12]. A discussion of weight-based max-min fairness policy and its implementation in ABR service is given [13]. The fairness criteria in the presence of MCR guarantees is discussed [14, 15].

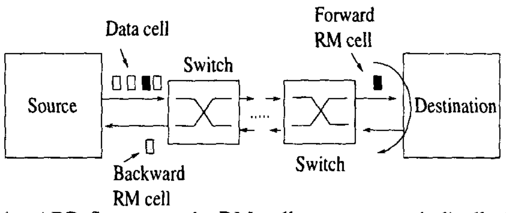


Figure 1: ABR flow control. RM cells are sent periodically by the source. The RM cell is turned around at the destination. The RM cells in the forward direction are called FRM cells and those in the backward direction are called BRM cells. The switches along the RM cell path indicate the rate which they can currently support.

In a related work, we have proposed a general definition of fairness, and gave an overload based ABR switch scheme which provides MCR guarantees [16]. In this paper, we propose three additional algorithms which use the overload factor to calculate the explicit rate feedback. All the proposed algorithms provide MCR guarantee and achieve generalized fairness.

The load factor (also referred to as "overload factor" or "overload") is the ratio of the measured input rate to the available ABR (available bit rate) capacity. ERICA+ switch scheme monitors the load on the link and calculates feedback based on the load. It tries to achieve unit load on links to efficiently use the link and also converge to max-min fairness [1]. Max-min fairness assumes zero MCR values. In this paper, we have used the generalized fairness with MCR as defined in [16].

Section II discusses the definition of generalized fairness used in algorithms. Section III gives a brief overview of ERICA+ and then discusses proposed algorithms. In section IV, various configurations used in the simulations are discussed. In section V we present the results of the simulations. The simulations test whether the schemes provide MCR guarantees and converge to generalized fairness. A comparison of the algorithms based on the simulations results is given in section VI. Finally, section VII gives the conclusions of the paper.

[1]This research was sponsored in part by Rome Laboratory/C3BC Contract #F30602-96-C-0156.

II. General Weighted Fairness: Definition

ATM Forum Traffic Management Specification 4.0 [1] recommends different definitions of fairness to be used when MCRs are non-zero. These fairness definitions are equal share of excess bandwidth, proportional to MCR of excess bandwidth, proportional to predetermined weights of bandwidth. Here, we give the definition (as defined [16]) of generalized weighted fairness which can realize all the above fairness definitions.

Define the following parameters:

A = excess bandwidth, to be shared by connections bottle-necked on this link.

μ_i = MCR of connection i.

μ = $\sum_{i=1}^{n} \mu_i$ Sum of MCRs of active connections which are bottle-necked by this link.

w_i = preassigned weight associated with the connection i.

g_i = generalized weighted fair Allocation for connection i.

The general weighted (GW) fair allocation is defined as follows:

$$g_i = \mu_i + \frac{w_i(A - \mu)}{\sum_{j=1}^{n} w_j}$$

The excess available bandwidth ($A - \mu$) is divided in proportion to the predetermined weights.

III. The Switch Schemes

The general structure of the algorithms proposed are similar to the ERICA+ [5]. First, we briefly discuss the ERICA+ algorithm and then give the general structure of the proposed algorithms. The three switch algorithms proposed have the same structure and only differ in, the way in which end of interval accounting is done, and the manner in which the feedback explicit rate is calculated. The switch algorithm with MCR guarantees discussed in [16] also has the same structure as three these algorithms.

A. Overview of ERICA+

ERICA+ operates at the output port of a switch. It periodically monitors the load, active number of VCs and provides feedback in the BRM (backward RM) cells. The measurement period is called the "averaging interval." The measurements are done in forward direction and feedback is given in the reverse direction. The complete description of ERICA+ algorithm can be obtained from [5]. In overload conditions, the algorithm calculates the maximum of the *FairShare* (link capacity divided by number of VCs) and *VCShare* (current cell rate divided by overload) as the feedback rate. In underload conditions, the maximum of *MaxAllocPrev* (which is the maximum allocation given in the previous averaging interval) and the previous two terms is the feedback rate. Part of the available ABR capacity is used for draining queues. Target ABR capacity is obtained by multiplying the total available ABR capacity by a $Fraction$ term. $1 - Fraction$ amount of the link capacity is used to drain the queues. $Fraction$ can be either a constant less than one (e.g., $Fraction = 0.9$ implies 90% link utilization), or dynamic function of switch queue length. ERICA+ uses a two hyperbolic curves with which control the queue in underload and overload regions. To ensure that a minimum amount of link capacity is used for data, the $Fraction$ value is limited to a minimum of $QDLF$ (queue drain limit factor). Typically a value $QDLF = 0.5$ is used. The dynamic queue control function given above can be expressed as follows

$$f(Q) = \begin{cases} \frac{bQ_0}{(b-1)Q+Q_0} & 0 \le Q \le Q_0 \\ min(\text{QDLF}, \frac{aQ_0}{(a-1)Q+Q_0}) & Q_0 < Q \le \infty \end{cases}$$

The "Target Delay" parameter specifies the desired queue length ($Q0$) at steady state. The 'a-curve' (hyperbola given by $\frac{aQ_0}{(a-1)Q+Q_0}$) is used as long as it evaluates to value greater than QDLF. The design of queue control functions which enable the switch algorithms to reduce rate oscillations, achieve constant delay and high link utilization (100%) is discussed in detail in [17].

B. Overload Based Algorithm: General Structure

The three different switch schemes have the following common algorithmic structure. They differ in the manner in which the feedback rate is calculated and in accounting. The algorithms perform the following steps for calculating the feedback rate:

1. The problem of non zero MCRs is reduced to one with zero MCRs. The MCR is subtracted from each connection's current cell rate (CCR(i)) to obtain the excess rate of each source (SR(i)) over MCR. If the source is transmitting at a rate less than its MCR, an excess rate of zero is used.

2. The switch algorithm for zero MCRs is applied to these excess rates to obtain the feedback rate. The excess available capacity ($A - \mu$) is divided in proportion to the pre-determined weight.

3. MCR for each source is added to the explicit feedback rate calculated in the previous step. The resulting rate is indicated in the ER field of BRM cells.

The sources transmit data initially at their initial cell rate (ICR) value. After one round trip time, the feedback from the switches arrives at the sources, and sources adjust their allowed rate accordingly. When a constant queue control function is used (say $Fraction = 0.9$) rates converge to the GW fairness allocation. When a dynamic queue control function is used, the available capacity varies depending on the queue length. Therefore, the feedback rate also varies till the queues are drained. Once the queues are drained, the feedback rates converge to the GW fair allocation. We present simulation results using both the constant queue and dynamic queue control functions.

Overload Based Algorithm Structure:

At the End of Each Averaging Interval:

$$\begin{aligned}
\text{ABR Capacity} &\leftarrow \text{Link Capacity} - \text{VBR Capacity} \\
&\quad - \sum_{i=0}^{n} \min(SR(i), \mu_i) \\
\text{TargetABRCap} &\leftarrow f(Q) \times \text{ABR Cap} \\
\text{Input Rate} &\leftarrow \text{ABR Input Rate} - \sum_{i=0}^{n} \min(SR(i), \mu_i) \\
z &\leftarrow \frac{\text{Input Rate}}{\text{TargetABRCap}}
\end{aligned}$$

End_of_Interval_Accounting()

When an FRM is received:

$$\text{CCR(i)} \leftarrow \text{CCR}_{RM\_Cell}$$

When a BRM is received:

$$\begin{aligned}
\text{Ex\_ER} &\leftarrow \textit{Calculate\_Excess\_ER()} \\
\text{ER} &\leftarrow \mu_i + \text{Ex\_ER} \\
\text{ER}_{RM\_Cell} &\leftarrow \text{Min}(\text{ER}_{RM\_Cell}, \text{ER}, \text{Target ABR Cap})
\end{aligned}$$

The key steps that differentiate the algorithms are the procedures *End_of_Interval_Accounting()* and *Calculate_Excess_ER()*.

C. *Algorithm A: ExcessFairShare/Overload*

The *ExcessFairShare* term is defined as follows:

$$ExcessFairShare(i) = \frac{w_i(A - \mu)}{\sum_{j=1}^{n} w_j}$$

This divides the excess available bandwidth $(A - \mu)$ (i.e., ABR capacity) in proportion to the weights $w(i)$. An allocation of $\mu_i + ExcessFairShare(i)$ for each source i is the GW fair allocation. A source might be bottlenecked at some other link, hence using less than its fair share of link capacity. By using the notion of activity level, the effective weight of the bottlenecked source can be calculated.

The activity level for a given VC is defined as follows:

$$AL(i) = min\left(1, \frac{SourceRate(i) - \mu_i}{ExcessFairShare(i)}\right)$$

The activity level can be used to accurately estimate the effective number of VCs. Effective number of VCs is given by the following expression:

$$\text{Effective number of VCs} = \sum_{j=1}^{n} AL(j)$$

The VCs which are bottlenecked by this link will have activity level of 1 and will be counted as one VC. The VCs bottlenecked elsewhere will are counted as fractional VCs depending on their activity level. The proof that above expression estimates accurately the effective number of VCs is given in [18].

We extend the notion activity level to the weighted case by multiplying the weight function with the activity level of the *ExcessFairShare* term. Therefore the *ExcessFairShare* is:

$$ExcessFairShare(i) = \frac{w_i(A - \mu)}{\sum_{j=1}^{n} w_j AL(j)}$$

In this algorithm, the *Ex_ER* is calculated based on $ExcessFairShare$ and the overload factor z. For each source, the activity level and the $ExcessFairShare$ are updated at the end of each interval. When a BRM cell arrives, the feedback rate is calculated as the $ExcessFairShare$ term, divided by the overload. The source rate ($VCShare$ term) is not used in feedback rate calculation.

If the network is overloaded $(z > 1)$ the Ex_ER decreases since fairshare is divided by the overload factor z. In underload conditions $(z < 1)$ the sources are asked to increase their rate. Due the recursive definition of $ExcessFairShare$ and activity levels these value converge.

As the network reaches steady state, the overload will become one and the *Ex_ER* will converge to the required $ExcessFairShare$, achieving GW fair allocation. Proof convergence is similar to the proof given in [16]. Since the overload factor varies every averaging interval, the rate oscillations increase for this algorithm. If the "averaging interval" is greater than the feedback loop, then the switch adjusts to the feedback rate before the next averaging interval. If the averaging interval is smaller, then before the source can adjust to the feedback rate, multiple feedbacks are given. In this situation $ExcessFairShare$ calculation is not accurate, since the source rate does not reflect the feedback rate. Exponential averaging of the $ExcessFairShare$ term is used to overcome this problem. Exponential averaging is done as follows:

$ExcessFairShare = \alpha\ ExcessFairShare + (1 - \alpha) PrevExcessFairShare$

where the $PrevExcessFairShare$ is the value calculated for the previous averaging interval. The parameter α is called the decay factor. In the simulations a value of $\alpha = 0.9$ was used.

End_of_Interval_Accounting():

```
foreach VC i do
    $AL(i) \leftarrow min\left(1, \frac{SourceRate(i) - \mu_i}{ExcessFairShare(i)}\right)$
    $ExcessFairShare(i) \leftarrow \frac{(\text{TargetABRCap}) w_i AL(i)}{\sum_{j=1}^{n} w_j AL(j)}$
endfor
```

Calculate_Excess_ER():

$$\text{Ex\_ER} \leftarrow \frac{\text{ExcessFairShare(i)}}{z}$$

D. Algorithm B: MaxAllocation/Overload

Let the GW fair allocation be $(g_1, g_2, \ldots, g_n)$ for n bottle-necked sources. The excess bandwidth is divided proportional to weights $(g_i - \mu_i)/w(i) = (g_j - \mu_j)/w(j)$. Let m be the VC such that term $(g_m - \mu_m)/w(m)$ is the maximum of such terms of all VCs. An allocation which assigns rates as $\mu_i + w(i)(g_m - \mu_m)/w(m)$ will achieve GW fairness. The term $(g_m - \mu_m)/w(m)$ is defined as the weighted maximum allocation. In algorithm B, the feedback is calculated as term proportional to weight of VC and the weighted maximum allocation is divided by overload. The overload in the denominator increases or decreases the allocation depending on the load. The source rate is not used in the feedback calculation. This algorithm can give rise to large queues if the weighted maximum allocation is measured incorrectly. This problem of large queues occurred during the simulation of this algorithm using GFC-2 configuration.

End_of_Interval_Accounting():

$$\begin{aligned}\text{WtMaxAllocPrev} &\leftarrow \text{WtMaxAllocCur}\\ \text{WtMaxAllocCur} &\leftarrow 0\end{aligned}$$

Calculate_Excess_ER():

$$\begin{aligned}\text{Ex\_ER} &\leftarrow \frac{\text{w(i)WtMaxAllocPrev}}{z}\\ \text{WtMaxAllocCur} &\leftarrow \text{Max (WtMaxAllocCur,Ex\_ER/w(i))}\end{aligned}$$

E. Algorithm C: VCShare and MaxAllocation

In this algorithm, the *End_of_Interval_Accounting()* is the same as in the previous algorithm (algorithm B). The *Ex_ER* is calculated based on the weighted maximum previous allocation ($WtMaxAllocPrev$) and $VCShare$ under underloaded conditions ($z < 1 + \delta$). For overloaded conditions, the $VCShare$ is given as the feedback rate. The problem of large queues explained in the previous algorithm is not expected to occur since the maximum previous allocation is given as feedback only if the link underloaded. As the overload converges to one, the $VCShare$ converges to $w(i)(g_m - \mu_m)/w(m)$, achieving the GW fair allocation.

Calculate_Excess_ER():

```
VCShare         ←  $\frac{max(0, SourceRate(i) - \mu_i)}{z}$
IF ($z > 1 + \delta$)
    THEN Ex_ER  ←  VCShare
    ELSE Ex_ER  ←  max(w(i) WtMaxAllocPrev, VCShare)
WtMaxAllocCur   ←  max(WtMaxAllocCur,Ex_ER/w(i))
```

IV. Simulation Configurations

We used simple, link bottleneck and source bottleneck configurations to test the proposed algorithms. Infinite sources were used (which have an infinite amount of data to send, and always send data at ACR) in all the simulations. The rates are expected to converge to GW fair allocation values in the presence of infinite sources. The algorithms are expected to give minimum rate guarantees for Poisson or self-similar sources. These types of sources were not used since the GW fair allocation for source with varying rates is also dynamically varying. The data traffic is only one way, from source to destination. Using two-way traffic would produce similar results, expect that the convergence time would be larger since the RM cells in the backward direction would travel with traffic from destination to source. All the link bandwidths are 149.76 (155.52 less the SONET overhead), except in the GFC-2 configuration.

A. Three Sources

This is a simple configuration in which three sources send data to three destinations over two switches and a bottleneck link (see Figure 2). This configuration is used to demonstrate that the switch algorithms can achieve the GW fairness.

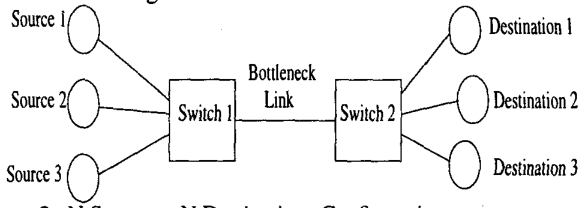


Figure 2: N Sources - N Destinations Configuration

B. Source Bottleneck

In this configuration, the source S1, is bottle-necked to rate (10 Mbps), which is below its fairshare (50 Mbps) for first 400 ms of the simulation (see Figure 3). This configuration tests whether the fairness criterion can be achieved in the presence of source bottleneck.

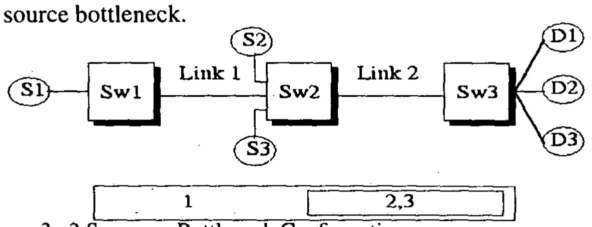


Figure 3: 3 Sources - Bottleneck Configuration

C. Generic Fairness Configuration - 2 (GFC-2)

This configuration is a combination of upstream and parking lot configuration (see Figure 4). In this configuration, all the links are bottle-necked links. This configuration is explained in [19].

D. Simulation Parameters

The simulations were done using an extensively modified version of NIST ATM simulator [20]. The parameter values for different configurations are given in Table 1. The algorithms were simulated using both constant queue control function ($Fraction = 0.9$) and using dynamic queue control function. For dynamic queue control, hyperbolic functions was used, with curve parameters $a = 1.15$ and $b = 1$. The $QDLF$ value was set to 0.5.

Exponential averaging was used to decrease the variation in measured quantities such as overload and number of VCs. Exponential averaging of overload factor and number of VCs were done with a decay factor of 0.8 for algorithm C. The algorithms A and B are more sensitive to overload factor. So, a the decay factor of 0.4 was used to average overload in algorithms A and B.

Table 1
Simulation Parameter Values

Configuration Name	Link Dist	Averaging interval	Target Delay	Wt Func
Three Sources	1000 Km	5 ms	1.5 ms	1
Src Bottleneck	1000 Km	5 ms	1.5 ms	1
GFC-2	1000 Km	15 ms	1.5 ms	1

The weight function of one was used in all configurations. This corresponds to MCR plus equal share of excess bandwidth. The value of $\delta = 0.1$ was used for algorithm C. In [16] it was shown that an overload based explicit rate algorithm achieves GW fairness for various weight functions.

V. Simulation Results

In this section we present the simulation results of algorithms using different configurations. Table 2 gives the expected GW fair allocation with constant queue control function (shown as configuration name and CQF) and with dynamic queue control function (shown as configuration and DQF) for each simulation using three source configuration. The queues when using constant queue control function took longer time to drain. The bottleneck link utilization was as expected 90% when constant queue control was used. With dynamic queue control the algorithms achieved 100% link utilization for bottlenecked links.

Table 2
GW fair allocation for different configurations

Config. and Queue Control	Src 1	Src 2	Src 3
Simple and CQF	24.93	44.93	64.93
Simple and DQF	29.92	49.92	69.92
Src Bottleneck + CQF (0-0.4s)	32.39	52.39	72.39
Src Bottleneck + DQF (0-0.4s)	39.86	59.86	79.86
Src Bottleneck + CQF (0.4-0.8s)	24.93	44.93	64.93
Src Bottleneck + DQF (0.4-0.8s)	29.92	49.92	69.92

A. Three Source: Results

The MCR value for the three source configuration is 10,30,50 Mbps for the source 1, source 2 and source 3 respectively. For the simulation with queue control, the excess bandwidth (149.76 - 90 =) 59.76 is divided equally among the three sources. The expected allocation is (10+59.76/3, 30+59.76/3, 50+59.76/3) = (29.92, 49.92, 69.92). Figure 5(a)-(c) shows the ACRs for algorithms A,B, and C (with dynamic queue control) respectively. From the graphs, it can be seen that the expected allocation is achieved by all the three algorithms.

B. Source Bottleneck: Results

In this configuration the MCR values of (10,30,50) Mbps were used. The total simulation time was 800 ms. The source S1, is bottle-necked at 10 Mbps for first 400 ms of the simulation. It always sends data up to 10 Mbps even its ACR larger than 10 Mbps. Figures 6 (a)-(c) shows the ACRs for the algorithms A, B, and C with constant queue control respectively. The expected allocation is (32.39,52.39,72.39) for first 400 ms and it is (29.92,49.92,69.92) between 400 ms and 800 ms. The algorithms do converge to the expected allocation. The algorithm A has lesser rate oscillations and converges faster than algorithm B and C.

C. GFC-2: Results

MCR value of zero was used for all sources expect for type A sources which had a value MCR of 5 Mbps. The expected allocation for each type of VC using constant queue control and dynamic queue control is given in the table 3. Figure 7 (a)-(c) show the ACRs of each type of VCs A through H for algorithms A, B, and C with constant queue control respectively. Algorithm A converges to the expected allocation. Algorithm B with constant queue control, does not converge to expected GW fair allocation within the simulation time as seen in figure 7(f). This is due the presence of sources with different round trip times sharing the same bottleneck link. In such a case the sources with larger round trip time take a long time to adjust the rates compared to the ones which have smaller round trip times. For example maximum allocation for A type VC which has large round trip time was assigned for seven VCs of type G which have small round trip time. This also led to large switch queues and slow convergence of algorithm B. The input rate at the link between SW6 and SW7 is overloaded by a factor of seven which gives rise to the huge queues.

Table 3
GFC-2 configuration: Expected allocations when using DQF and CQF for each type of VC

A	B	C	D	E	F	G	H
11.25	5	33.75	33.75	33.75	6.25	5	50.62
9	4.5	31.5	31.5	31.5	9	4.5	47.25

VI. Comparison of Switch Schemes

Table 4 gives a comparison of the algorithms.

Algorithm A has higher complexity than the other two algorithms. Algorithm B results in large switch queues since in presence of multiple round trip time. The algorithm C is the

Table 4
Comparison of the algorithms

Algo-rithm	End of Intrvl Complexity	Feedback Complexity	Max Q Len	Sensitivity to Q cntrl
A	O(N)	O(1)	Med	High
B	O(1)	O(1)	Large	Low
C	O(1)	O(1)	Med	Low

best of the proposed algorithm since it has O(1) complexity both at the end of interval and when a BRM cell arrives. Also, algorithm C has smaller queues compared to algorithm B.

VII. Conclusion

In this paper, we have presented three algorithms which achieve GW fairness and provide MCR guarantee. The algorithms monitor the load on the link and calculate the overload factor. The overload and other quantities ($ExcessFairShare$ or $WtMaxAllocPrev$) are used to calculate the feedback rates.

The algorithms proposed have similar structure. The algorithms differ in the end of interval accounting and feedback calculation. Simulations show that the algorithms converge to GW fairness in most cases. Queue control can be done using constant function and dynamic (hyperbolic) function. Algorithm A has O(N) complexity for the end of interval calculations. Algorithm B, can give rise to large queues if the configuration has sources with different round trip times sharing the same link. The algorithm C, which uses the $VCShare$ and $WtMaxAllocPrev$ is the best, since it has O(1) complexity and is less sensitive to queue control function.

VIII. References

[1] Shirish S. Sathaye. "ATM Forum Traffic Management Specification Version 4.0". April 1996

[2] Nanying Yin and M. G. Hluchyj. "On closed-loop rate control for ATM cell relay networks". *In Proc. of IEEE INFOCOM*, pp. 99-108, 1994.

[3] Anna Charny. "An algorithm for rate allocation in a packet-switching, network with feedback". *Master's thesis*, MIT, Cambridge, May 1994.

[4] Danny H. K. Tsang, Wales K. F. Wong. "A new rate-based switch algorithm for ABR traffic to achieve max-min fairness with analytical approximation and delay adjustment". *In Proc. IEEE Globecom'96*.

[5] Shivkumar Kalyanaraman, Raj Jain, Rohit Goyal, Sonia Fahmy, and Bobby Vandalore. "The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks," [2] Submitted to *IEEE/ACM Transactions on Networking*, April 1999,

[6] C. Fulton, San-Qi Li and C. S. Lim. "UT: ABR feedback control with tracking". *Preprint.*

[7] L. Kalampoukas, A. Varma, K. K. Ramakrishnan. "An efficient rate allocation algorithm for ATM networks providing max-min fairness". *In Proc. of the 6th IFIP International Conference on High Performance Networking*, September 1995.

[8] K. Siu and T. Tzeng. "Intelligent congestion control for ABR service in ATM networks". *Computer Communication Review*, vol. 24, no. 5, pp. 81-106, October 1995.

[9] Y. Afek, Y. Mansour, and Z. Ostfeld. "Phantom: A simple and effective flow control scheme". *In Proceedings of the ACM SIGCOMM*, August 1996.

[10] L. Roberts. "Enhanced PCRA (Proportional Rate Control Algorithm)". *ATM Forum Contribution/AF-TM 94-0735R1*, August 1994.

[11] Yiewei T. Hou, Henry H.-Y. Tzeng, Shivendra S. Panwar. "A generalized max-min rate allocation policy and its distributed implementation using the ABR flow control mechanism". *In Proc. of INFOCOM*, April 1998.

[12] Santosh P. Abraham and Anurag Kumar. "A stochastic approximation approach for a max-min fair adaptive rate control of ABR sessions with MCRs". *In Proc. of INFOCOM*, April 1998.

[13] Y. T. Hou, H. Tzeng, and S. S. Panwar. "A Simple ABR switch algorithm for the weighted max-min fairness policy". In Proc. IEEE ATM'97 Workshop, pp. 329-338, Lisbon, Portugal, May 25-28, 1997.

[14] D. Hughes. "Fair share in the context of MCR". *ATM Forum contribution/AF-TM 94-0977*, October 1994.

[15] N. Yin. "Max-min fairness vs. MCR guarantee on bandwidth allocation for ABR". *In Proc. of IEEE ATM'96 Workshop*, San Franscisco, CA, August 25-27, 1996.

[16] Bobby Vandalore, Sonia Fahmy, Raj Jain, Rohit Goyal, Mukul Goyal, "A Definition Definition of General Weighted Fairness and its Support in Explicit Rate Switch Algorithms", *ICNP '98*, Austin, Texas, October 1998, pp 22-30.

[17] Bobby Vandalore, Raj Jain, Rohit Goyal, Sonia Fahmy "Design and Analysis of Queue Control Functions for Explicit Rate Switch Schemes", *IC3N'98*, October 1998, pp 780-786.

[18] Sonia Fahmy, Raj Jain, Shivkumar Kalyanaraman, Rohit Goyal and Bobby Vandalore, "On Determining the Fair Bandwidth Share for ABR Connections in ATM Networks," *Proc. of the IEEE International Conference on Communications (ICC) 1998*, June 1998

[19] Robert J. Simcoe. "Test configurations for fairness and other tests". *ATM Forum*/94-0557, July 1994.

[20] Nada Golmie. "Netsim: network simulator". *http://www.nist.gov/.*

[2] All our papers and ATM Forum contributions are available through http://www.cis.ohio-state.edu/~jain/

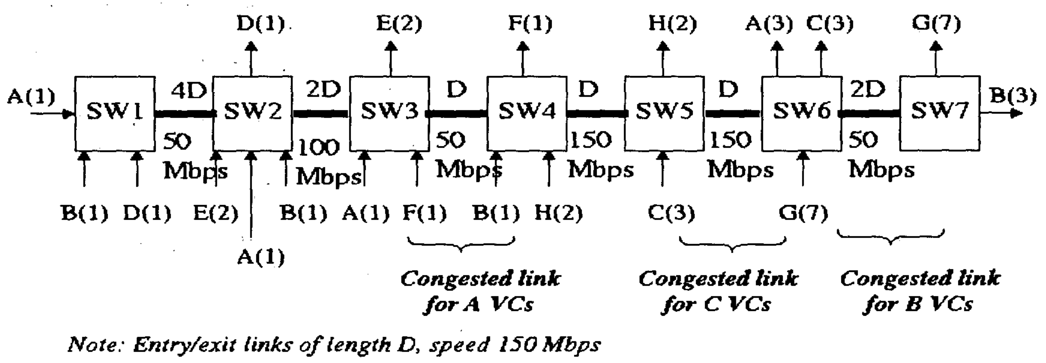
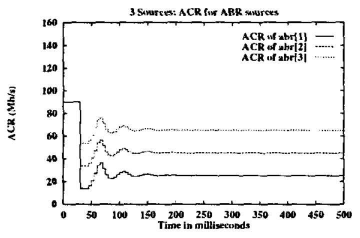
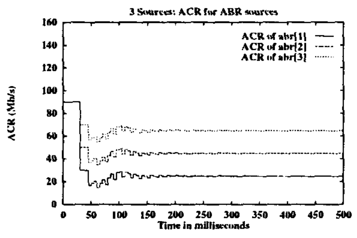


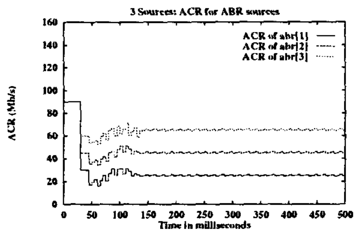
Figure 4: Generic Fairness Configuration - 2



(a) Algorithm A and DQF

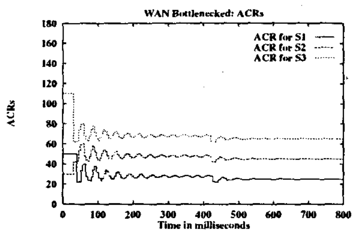


(b) Algorithm B and DQF

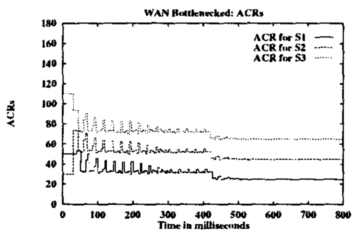


(c) Algorithm C and DQF

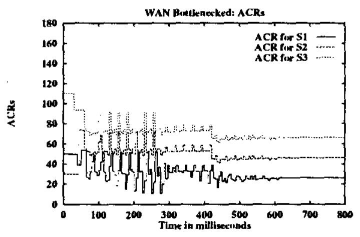
Figure 5: Three Sources: ACR graphs for algorithms A, B, and C.



(a) Algorithm A with CQF

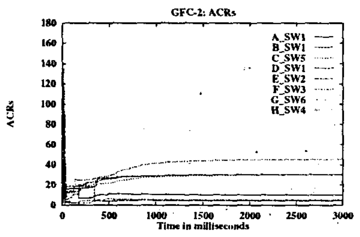


(b) Algorithm B with CQF

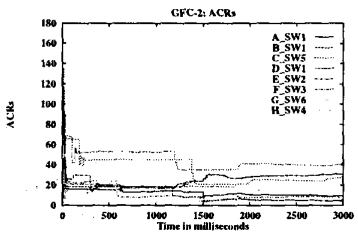


(c) Algorithm C with CQF

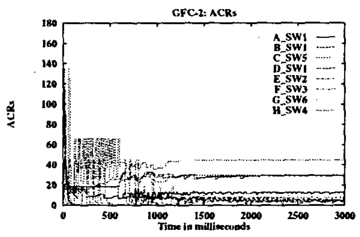
Figure 6: Source Bottleneck: ACR graphs for Algorithm A, B, and C.



(a) Algorithm A and CQF



(b) Algorithm B and CQF



(c) Algorithm C and CQF

Figure 7: GFC-2 config: ACR graphs for algorithms A, B, and C.