****

ATM Forum Document Number: ATM Forum/98-0408

****

Title: Overload based Explicit Rate Switch Schemes with MCR guarantees

****

Abstract:
In this contribution, we present four overload based switch schemes which provide MCR guarantees. A typical explicit rate switch scheme monitors the load on a link and gives feedback to the sources. The overload factor is defined ratio of rate of input to link capacity. The switch schemes proposed use the overload factor to calculate feedback rates. A dynamic queue control mechanism is used to control queues and achieve constant queuing delay at steady state. The algorithms proposed are studied and compared using different configurations.

****

Source:
Bobby Vandalore, Sonia Fahmy, Raj Jain, Rohit Goyal, Mukul Goyal
The Ohio State University Department of Computer and Information Science
Columbus, OH 43210-1277

Raj Jain is now at Washington University in Saint Louis, jain@cse.wustl.edu http://www.cse.wustl.edu/~jain/

This presentation of this contribution is sponsored by NASA Lewis Research Center.

****

Date: February 1998

****

Distribution: ATM Forum Technical Working Group Members (AF-TM)

****

****

1 Introduction

In this contribution we extend our previous study [2], where we had proposed a general definition of fairness and gave an overload based ABR (available bit rate) switch scheme which provides MCR (minimum cell rate) guarantees. In this contribution we propose three additional algorithms which use overload factor to calculate explicit feedback rate. All the proposed algorithm provide MCR guarantees and generalized fairness.

The load factor (also referred as "overload factor" or "overload") is the ratio of the measured input rate to the available ABR (available bit rate) capacity. Switch schemes monitor the load on the link and calculates feedback [4, 1] based on the load. The switch schemes try to achieve unity load for efficient use of link. The current schemes converge to max-min fairness [5]. Max-min fairness assumes zero MCR values. In this contribution we have used the generalized fairness as defined in [2].

The proposed algorithms are similar to ERICA+ [1]. We first briefly describe ERICA+ and then the algorithms proposed. The algorithms are tested using simulations on various configurations. The simulations test whether the schemes provide MCR guarantees and converge to generalized fairness. We give a comparison of the algorithms based on the simulations results.

2 General Fairness: Definition

Define the following parameters:

$A_l =$ Total available bandwidth for all ABR connections on a given link l.

$A_b =$ Sum of bandwidth of underloaded connections which are bottlenecked elsewhere.

$A = A_l - A_b$, excess bandwidth, to be shared by connections bottlenecked on this link.

$N_a =$ Number of active connections

$N_b =$ Number of active connections bottlenecked elsewhere.

$n = N_a - N_b$, number of active connections bottlenecked on this link.

$\mu_i =$ MCR of connection i.

$\mu = \sum_{i=1}^{n} \mu_i$ Sum of MCRs of active connections within bottlenecked on this link.

$w_i =$ preassigned weight associated with the connection i.

$g_i =$ GW fair Allocation for connection i.

The general 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.

3 Description Switch Schemes

The general structure of algorithms proposed are similar to the ERICA+ [1]. First, we briefly discuss ERICA+ algorithm and then give the general structure of the proposed algorithms. The four different algorithms have the same structure and differ in end of interval accounting and the manner in which the feedback is calculated.

3.1 Overview of ERICA+

ERICA+ operates at 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 the "averaging interval". The measurements are done in forward direction and feedback is given in the backward direction.

ERICA+ Algorithm

At the end of Averaging Interval:

$$\text{Total ABR Capacity} \leftarrow \text{Link Capacity} - \text{VBR Capacity} \tag{1}$$

$$\text{Target ABR Capacity} \leftarrow \mathit{Fraction} \times \text{Total ABR Capacity} \tag{2}$$

$$\tag{3}$$

$$z \leftarrow \frac{\text{ABR Input Rate}}{\text{Target ABR Capacity}} \tag{4}$$

$$\text{FairShare} \leftarrow \frac{\text{Target ABR Capacity}}{\text{Number of Active VCs}} \tag{5}$$

$$\text{MaxAllocPrevious} \leftarrow \text{MaxAllocCurrent} \tag{6}$$

$$\text{MaxAllocCurrent} \leftarrow \text{FairShare} \tag{7}$$

When an FRM is received:

CCR[VC] $\leftarrow$ CCR_in_RM_Cell

When a BRM is received:

$$\text{VCShare} \leftarrow \frac{CCR[VC]}{z} \tag{8}$$

$$\text{IF } (z > 1 + \delta)$$

$$\text{THEN ER} \leftarrow \text{Max (FairShare, VCShare)} \tag{9}$$

$$\text{ELSE ER} \leftarrow \text{Max (MaxAllocPrevious, VCShare)} \tag{10}$$

$$\text{MaxAllocCurrent} \leftarrow \text{Max (MaxAllocCurrent,ER)} \tag{11}$$

$$\text{IF (ER} > \text{FairShare AND CCR[VC]} < \text{FairShare)}$$

$$\text{THEN ER} \leftarrow \text{FairShare} \tag{12}$$

$$\text{ER\_in\_RM\_Cell} \leftarrow \text{Min (ER\_in\_RM\_Cell, ER, Target ABR Capacity)} \tag{13}$$

For overload ($z > 1 + \delta$) condition, the algorithm calculates the maximum of the *FairShare* and *VCShare* as the feedback rate. For underload condition the maximum of *MaxAllocPrevious* (which is the maximum allocation given in the previous averaging interval) and the previous two terms is the feedback rate. Line 12 avoids sudden increase in the feedback rate. The *Fraction* term is used to control the queues.

3.2 Overload Based Algorithm: General Structure

The four different switch schemes have the following common algorithmic structure. They differ in the manner in which the feedback rate is calculated and accounting.

Overload Based Algorithm X

At the end of Averaging Interval:

$$\begin{aligned}
\text{Total ABR Capacity} &\leftarrow \text{Link Capacity} - \text{VBR Capacity} \\
&\quad - \sum_{i=0}^{n} \min(\mathit{SourceRate}(i), \mu_i) \qquad (14) \\
\text{Target ABR Capacity} &\leftarrow \mathit{Fraction} \times \text{Total ABR Capacity} \\
\text{Input Rate} &\leftarrow \text{ABR Input Rate} - \sum_{i=0}^{n} min(\mathit{SourceRate}(i), \mu_i) \\
z &\leftarrow \frac{\text{Input Rate}}{\text{Target ABR Capacity}} \qquad (15) \\
&\textbf{End_of_Interval_Accounting()} \qquad (16) \\
& \qquad (17)
\end{aligned}$$

When an FRM is received:
CCR[VC] ← CCR_in_RM_Cell

When a BRM is received:

$$\begin{aligned}
\text{Excess_ER} &\leftarrow \textbf{Calculate_Excess_ER()} \qquad (18) \\
\text{ER} &\leftarrow \mu_i + \text{Excess_ER} \qquad (19) \\
\text{ER_in_RM_Cell} &\leftarrow \text{Min(ER_in_RM_Cell,ER,Target ABR Capacity)} \qquad (20) \\
& \qquad (21)
\end{aligned}$$

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

3.3 Algorithm A: VCShare and ExcessFairShare

In *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 $(A-\mu)$ proportional to the weights $w(i)$.

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

$$AL(i) = minimum\left(1, \frac{\text{Source}Rate(i) - \mu_i}{ExcessFairshare(i)}\right)$$

The activity level can be used to accurately estimate the effective number of VCs [3]. We extend this notion to the weighted case by multipling the weight function with the activity level in the denominator of the *ExcessFairshare* term. Therefore the *ExcessFairshare* is:

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

In Algorithm A, the *Excess_ER* is calculated based on the $VCShare$ and the $Excessfairshare$ terms.

End_of_Interval_Accounting():
foreach VC i

$$AL(i) \leftarrow minimum\left(1, \frac{SourceRate(i)-\mu_i}{ExcessFairshare(i)}\right) \tag{22}$$

$$ExcessFairshare(i) \leftarrow \frac{(\text{Target ABR Capacity})w_i}{\sum_{j=1}^{n} w_j AL(j)} \tag{23}$$

$$\tag{24}$$

endfor

Calculate_Excess_ER():

$$\text{VCShare} \leftarrow \frac{SourceRate(i)-\mu_i}{z} \tag{25}$$

$$\text{Excess\_ER} \leftarrow \text{Max (ExcessFairshare(i), VCShare)} \tag{26}$$

$$\tag{27}$$

3.4 Algorithm B: ExcessFairShare/Overload

In this version the *Excess_ER* is calculated based on $Excessfairshare$ and overload factor z. As the network reaches steady state the overload will become one and the *Excess_ER* will converge the required fairshare. In this algorithm the *End_of_Interval_Accounting()* is the same as in the previous algorithm (algorithm A).

Calculate_Excess_ER():

$$\text{Excess\_ER} \leftarrow \frac{\text{ExcessFairshare(i)}}{z} \tag{28}$$

3.5 Algorithm C: MaxAllocation/Overload

The weighted maximum allocation is defined as the maximum of allocation divided by the weight among all VCs. The *Excess_ER* is calculated based on weighted maximum previous allocation ($WtMaxAllocPrevious$) and overload. Let i be the VC number in the BRM cell.

End_of_Interval_Accounting():

$$\text{WtMaxAllocPrevious} \leftarrow \text{WtMaxAllocCurrent} \tag{29}$$

$$\text{WtMaxAllocCurrent} \leftarrow 0 \tag{30}$$

$$\tag{31}$$

Calculate_Excess_ER():

$$\text{Excess\_ER} \leftarrow \frac{w(i)\text{WtMaxAllocPrevious}}{z} \tag{32}$$

$$\text{WtMaxAllocCurrent} \leftarrow \text{Max (WtMaxAllocCurrent,Excess\_ER/w(i))} \tag{33}$$
$$\tag{34}$$

Suppose j be the VC such that $Excess\_ER(j)/w(j)$ is the maximum of $Excess\_ER(i)/w(i)$. The $Excess\_ER(i)$ calculated by the above algorithm is proportional to the weight $w(i)$. As the overload converges to one, the allocation $Excess\_ER(i)$ converges to the $Excessfairshare(i)$ term.

3.6 Algorithm D: VCShare and MaxAllocation

The *Excess_ER* is calculated based on weighted maximum previous allocation ($WtMaxAllocPrevious$) and $VCShare$. In this algorithm the *End_of_Interval_Accounting()* is the same as in the previous algorithm (algorithm C).

Calculate_Excess_ER():

$$\begin{aligned}
\text{VCShare} &\leftarrow \frac{SourceRate(i) - \mu_i}{z} && (35)\\
\text{IF } (z > 1 + \delta) & \\
\text{THEN Excess_ER} &\leftarrow \text{VCShare} && (36)\\
\text{ELSE Excess_ER} &\leftarrow \text{Max (w(i) WtMaxAllocPrevious, VCShare)} && (37)\\
\text{WtMaxAllocCurrent} &\leftarrow \text{Max (WtMaxAllocCurrent,Excess_ER/w(i))} && (38)\\
& && (39)
\end{aligned}$$

4 Simulation Configurations

We used simple, transient, link bottleneck and source bottleneck configurations to test the proposed algorithms. Infinite sources were used (have infinite amount of data to send, and always send data at ACR) in all the simulations. The data traffic is only one way, from source to destination. All the link bandwidths are 149.76 (155.52 less the SONET overhead), except in the GFC-2 configuration.

4.1 Three Sources

This is a simple configuration in which three sources send data to three destinations over a two switchs and a bottleneck link. See figure 1. This configuration is used to demonstrate that the switchs algorithms can achieve the general fairness.

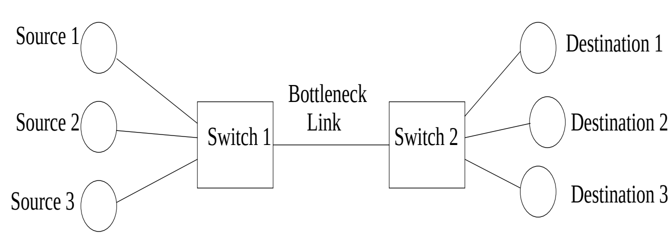


Figure 1: N Sources - N Destinations Configuration

4.2 Source Bottleneck

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

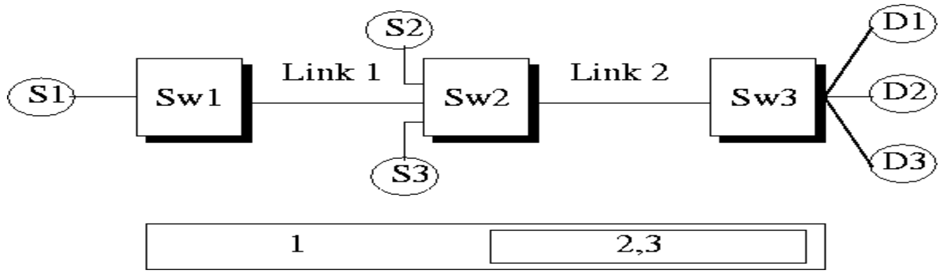


Figure 2: 3 Sources - Bottleneck Configuration

4.3 Generic Fairness Configuration - 2 (GFC-2)

This configuration is a combination of upstream and parking lot configuration (See Figure 3). In the configuration all the links are bottlenecked links. This configuration is explained in [7].

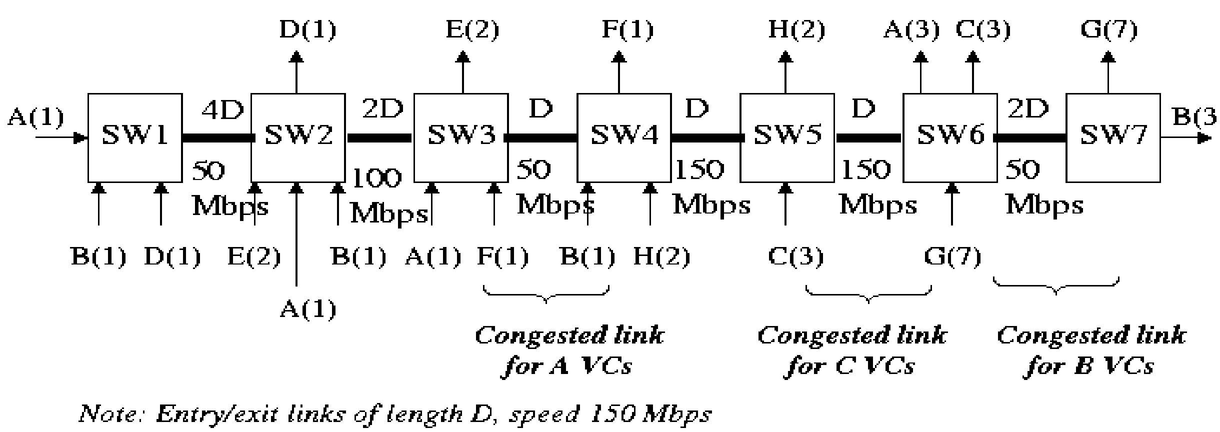


Figure 3: Generic Fairness Configuration - 2

4.4 Simulation Parameters

The simulations were done using extensively modified version of NIST ATM simulator [6]. The parameters values for different configurations is given in Table 1. The algorithms use dynamic queue control to vary the target ABR capacity depending on size of queue at the switch. The queue control function achieves a constant queue length at steady state. The "Target Delay" parameter specifies the desired queue length at steady state.

Table 1: Simulation Parameter Values

Configuration Name	Link Distance	Averaging interval	Target Delay	Weight Function
Three Sources	1000 Km	5 ms	1.5 ms	1
Source Bottleneck	1000 Km	5 ms	1.5 ms	1
GFC-2	1000 Km	15 ms	1.5 ms	1

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 algorithms A and D. The algorithms B and C are more sensitive to overload factor. So, a the decay factor of 0.4 was used to average overload in algorithms C and D.

The weight function value 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 D.

5 Simulation Results

In this section we present the simulation results of algorithms using different configurations.

5.1 Three Source: Results

The MCR value for the three soure configuration is 10,30,50 for the soure 1, source 2 and source 3 respectively. The excess bandwidth is (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, 39.92, 69.92). The figure 4(a)-(d) shows the ACRs (allowed cell rate) algorithms A,B,C and D respectively. From the figure it can be seen that the expected allocation is achieved by all the four algorithms.

5.2 Three Source transient : Results

MCR value of zero was used for all three sources in this configuration. The configuration simulation was simulated for 1.2 seconds. Source 2, is a transient source. It is active between 0.4 to 0.8 seconds of the simulation. The expected allocation is (74.88,0,74.88) during (0,0.4s) and (0.8-1.2s) which source 2 is not active. The expected allocation is (49.92,49.92,49.92) during (0.4,0.8) interval. The figure 5 (a)-(d) shows the ACRs for the algorithms A, B, C and D respectively. All the algorithms achieve the expected allocation in both non-transient and transient periods. Algorithm B is sensitive to queue control function, hence there rate oscillations during the non-transient periods.

5.3 Source Bottleneck: Results

In this configuration the MCRs of (10,30,50) were used. The total simulation time was 800 ms. The source two is bottlenecked 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. The figures 6 (a)-(d) shows the ACRs for the algorithms A, B, C and D respectively. The expected allocation is (49.86,59.86,79.86) for frist 400 ms and it is (29.92,49.92,69.92) after 400 ms. The algorithms A and B do not converge to the expected allocation. The CCR values in the RM cells do not reflect the actual source rate. The algorithms C and D do converge to the expected allocation. Algorithm C performs better than algorithm D, since it has lesser oscillations. The figures 7 (a)-(b) show the ACRs using measured source rate (per VC option) instead CCR field of for algorithms A, B. When measured source rate is used the algorithms A and B do converge to expected allocations in the presence of source bottleneck.

5.4 GFC2 : Results

MCR value of zero was used for all sources. The figure 8 (a)-(d) show the ACRs of each type of VCs A through H for algorithms A, B, C and D respectively. The graphs show that the expected allocation as given in the table 2 is achieved by all the algorithms. Algorithm B and D have rate oscillations due to queue control. The maximum queue occured at switch SW6 around 500 ms for all algorithms. The value of the maximum queue was 39000, 30000, 340000 and 34000 cells for algorithms A, B, C and D respectively. Algorithm C does not have any queue control. Algorithm C had a huge queue because the 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. 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 2: GFC-2 configuration: Expected allocations

VC	A	B	C	D	E	F	G	H
Expected allocation	10	5	35	35	35	10	5	52.5

6 Comparsion of Switch Schemes

The Table 3 gives a comparision of the algorithms.

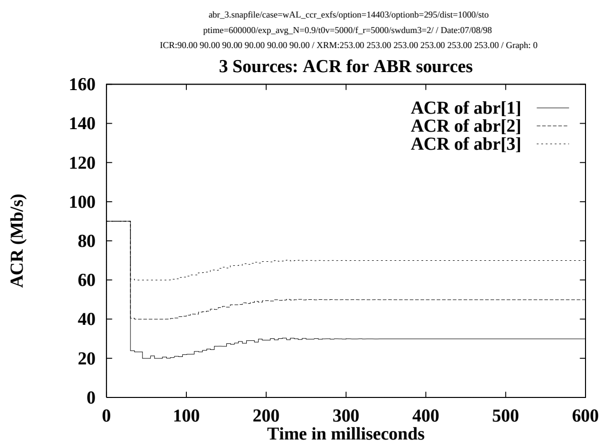
Scheme Name	End of Interval Complexity	Feedback Complexity	Max. Queue Length	Requires PerVC for Source Bottleneck	Sensitivity to Queue control
Algorithm A	O(N)	O(1)	Medium	Yes	Yes
Algorithm B	O(N)	O(1)	Medium	Yes	Yes
Algorithm C	O(1)	O(1)	Large	No	No
Algorithm D	O(1)	O(1)	Medium	No	No

Table 3: Comparison of the algorithms

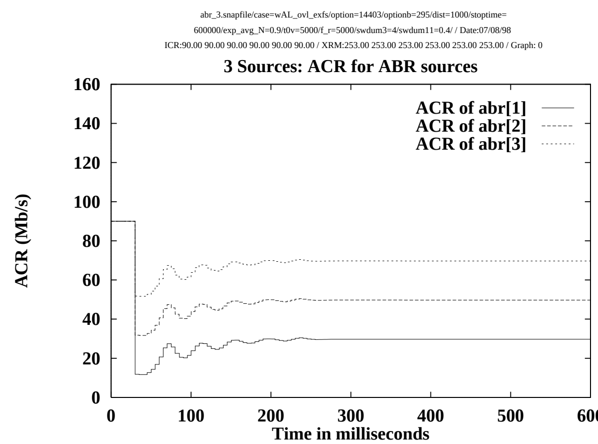
The algorithm D is the best of the proposed algorithm since it is of O(1) complexity, does not require per VC accounting and is not sensitive of the queue control function.

7 Conclusion

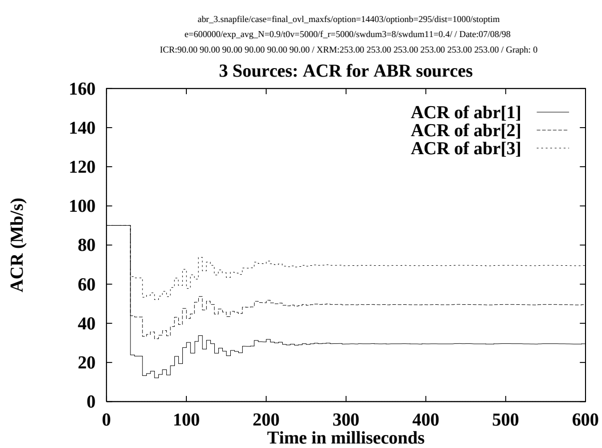
In this contribution we have presented four algorithms which achieve generalized fairness and provide MCR guarantee. The algorithms monitor the load on the link and calculate the overload factor. The overload is used with $ExcessFairshare$ or $WtMaxAllocPrevious$ to calculate the feedback. The algorithm D, which uses the $VCShare$ and $WtMaxAllocPrevious$ is the best, since it has O(1) complexity and does not require per VC accounting to handle source bottlenecks.



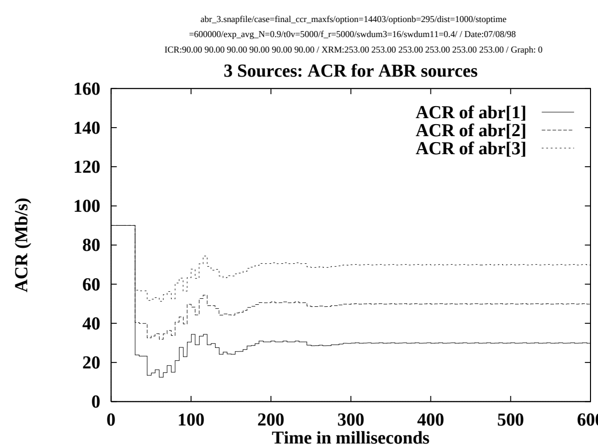
(a) Algorithm A



(b) Algorithm B

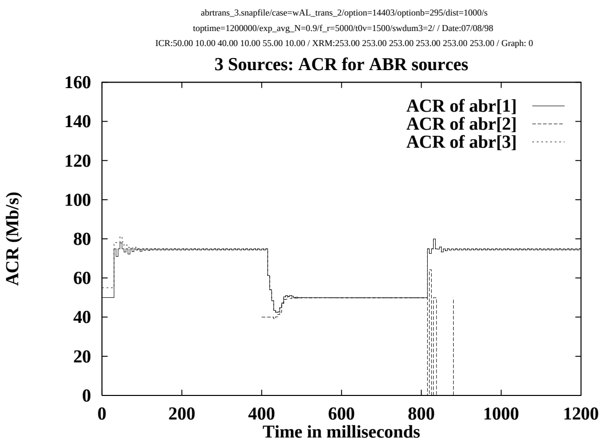


(c) Algorithm C

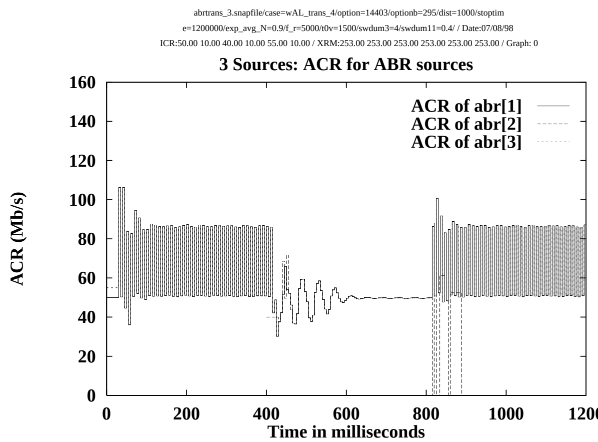


(d) Algorithm D

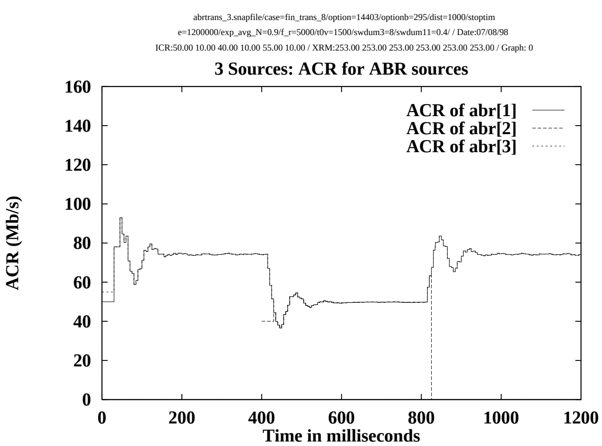
Figure 4: Three Sources: ACR graphs for algorithms A, B, C and D.



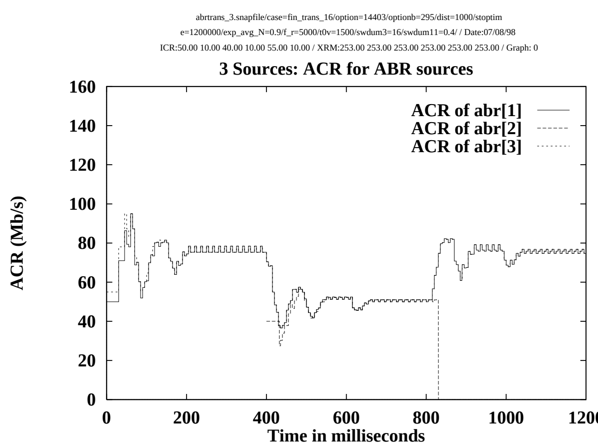
(a) Algorithm A



(b) Algorithm B

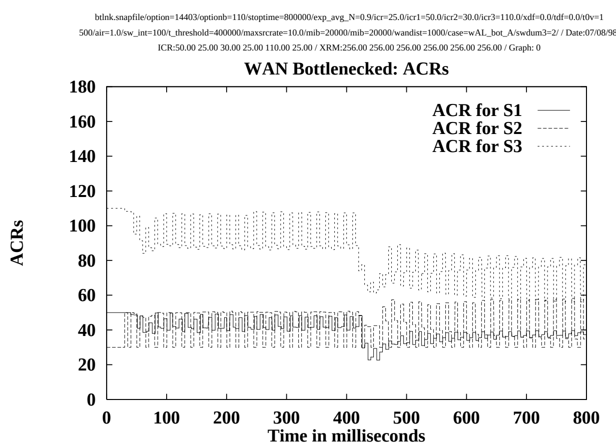


(c) Algorithm C

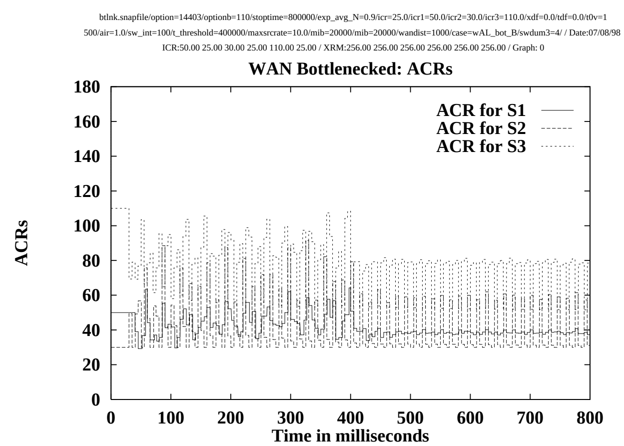


(d) Algorithm D

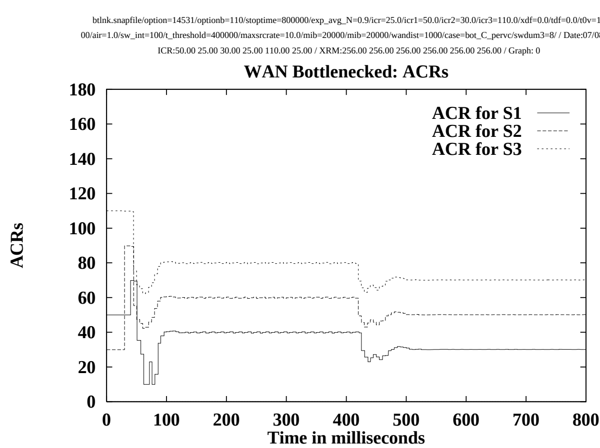
Figure 5: Three Sources transient: ACR graphs for algorithms A, B, C and D.



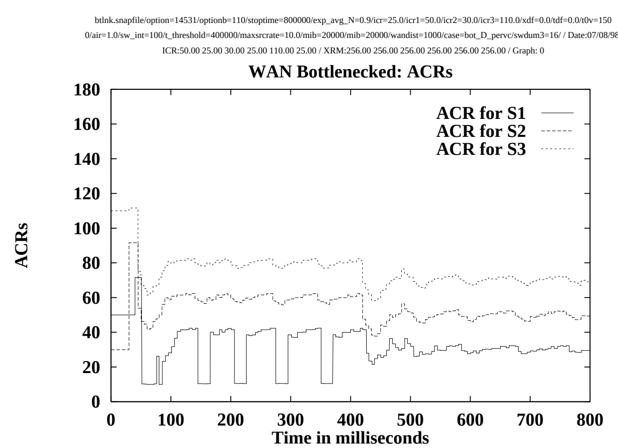
(a) Algorithm A



(b) Algorithm B

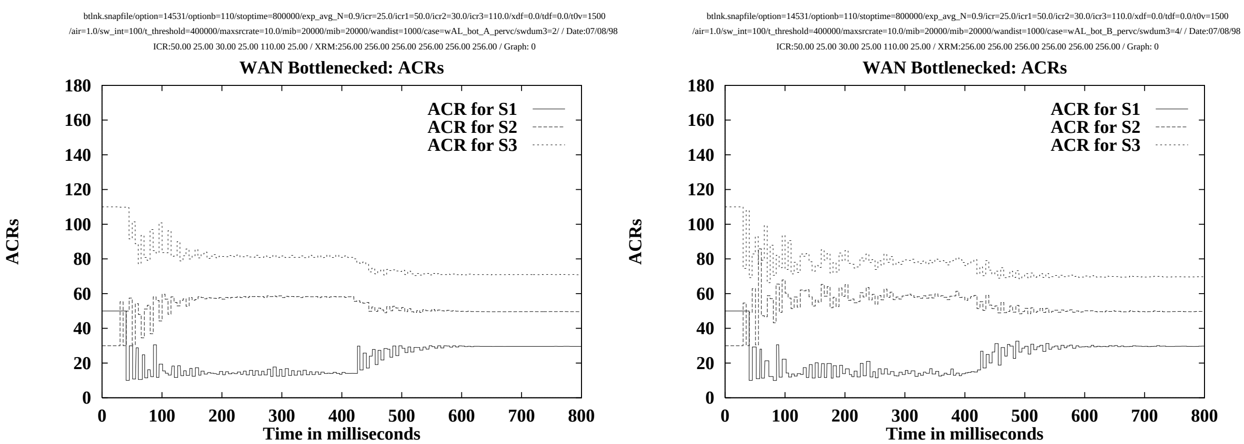


(c) Algorithm C



(d) Algorithm D

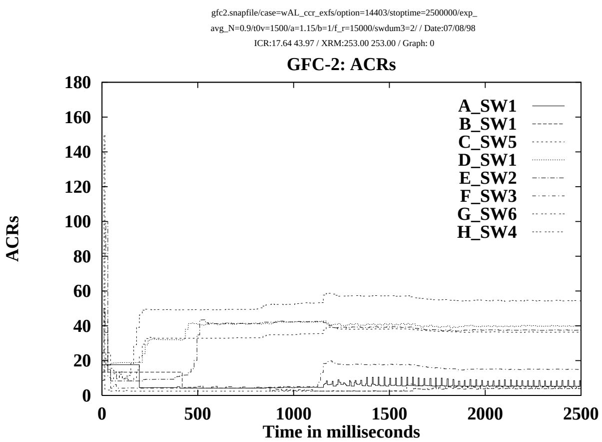
Figure 6: Source Bottleneck: ACR graphs for Algorithm A, B, C and D



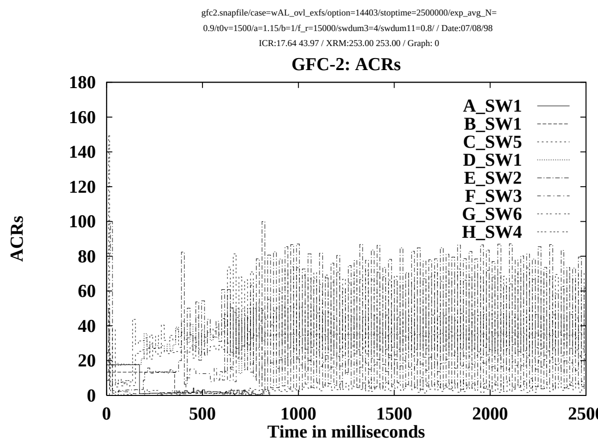
(a) Algorithm A

(b) Algorithm B

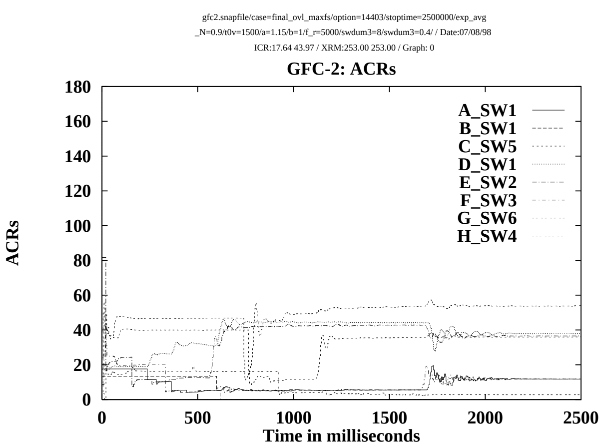
Figure 7: Source Bottleneck: ACR graphs for Algorithm A, B using measured source rate



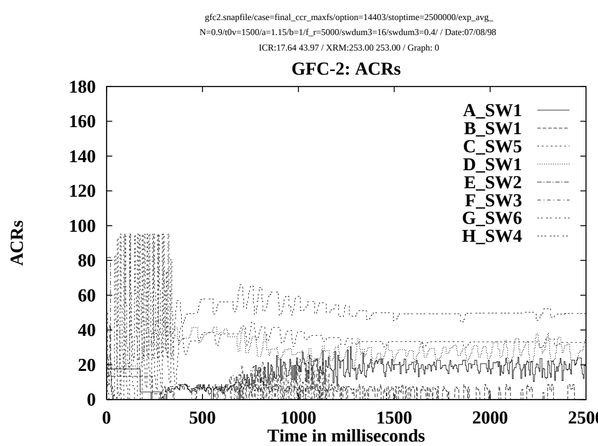
(a) Algorithm A



(b) Algorithm B



(c) Algorithm C



(d) Algorithm D

Figure 8: GFC-2 configuration: ACR graphs for algorithms A, B, C and D.

References

[1] Shivkumar Kalyanaraman, Raj Jain, Rohit Goyal, Sonia Fahmy, and Bobby Vandalore. [1] "The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks". Submitted to *IEEE/ACM Transactions on Networking*, November 1997,

[2] Bobby Vandalore, Sonia Fahmy, Raj Jain, Rohit Goyal, Mukul Goyal, "Generalized Fairness support in Switch Algorithms" ATM Forum/98-0151, February 1998,

[3] Sonia Fahmy, Raj Jain, Shivkumar Kalyanaraman, Rohit Goyal, and Bobby Vandalore. Determining the number of active ABR sources in switch algorithms. ATM Forum/98-0154, February 1998.

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

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

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

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

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