****

ATM Forum Document Number: ATM Forum/98-0154

****

Title: Determining the number of active ABR sources in switch algorithms

****

Abstract:
The ABR service is designed to fairly allocate the bandwidth unused by higher priority services. The network indicates to the ABR sources the rates at which they should transmit to minimize their cell loss. Switches must constantly measure the demand and available capacity, and divide the capacity fairly among the contending connections. In order to compute the fair and efficient allocation for each connection, a switch needs to determine the effective number of active connections. In this contribution, we propose a method for determining the number of active connections and the fair bandwidth share for each. We prove the efficiency and fairness of the proposed method analytically, and simulate it for a number of configurations.

****

Source:
Sonia Fahmy, Raj Jain, Shivkumar Kalyanaraman, Rohit Goyal, and Bobby Vandalore
The Ohio State University
Department of Computer and Information Science

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

This presentation of this work at the ATM Forum is sponsored by NASA Lewis Research Center.

****

Date: February 1998

****

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

****

Notice:
This contribution has been prepared to assist the ATM Forum. It is offered to the Forum as a basis for discussion and is not a binding proposal on the part of any of the contributing organizations. The statements are subject to change in form and content after further study. Specifically, the contributors reserve the right to add to, amend or modify the statements contained herein.

****

This contribution is a revised version of a paper to be presented at the IEEE International Conference on Communications (ICC) 1998 [5].

1 Introduction

Determining the fair bandwidth share for the active ABR connections is an extremely complex problem. This is because fairness is commonly measured by the max-min fairness criteria (defined in the next section). Intuitively, fairness means that if a connection is bottlenecked elsewhere, it should be allocated the maximum it can use, and the left over capacity should be fairly divided among the connections that can use it. The switch should indicate this fair bandwidth share to the sources, while also accounting for the load and queuing delays at the switch.

This contribution proposes a method to determine the fair bandwidth share for the active ABR connections, and analyzes the performance of this method using both simple mathematical proofs and simulations. The remainder of the contribution is organized as follows. In the next section, we describe the original ERICA switch algorithm which is employed in this study. Sections 3 and 4 point out some problems with the original ERICA algorithm, and describe how ERICA has solved these problems. We then describe our proposed method (which also overcomes those problems), and give a proof of its correctness, and a number of examples of its operation. Finally, we analyze the performance of the proposed method.

2 The Original ERICA Switch Algorithm

Several switch algorithms have been developed to compute the feedback to be indicated to ABR sources in RM cells [1]-[6],[10, 11]. The ERICA algorithm [8, 10] is one of these algorithms.

The ERICA algorithm aims at computing a fair and efficient allocation of the available bandwidth to all contending sources. Like any dynamic resource management algorithm, it requires monitoring the available capacity and the current demand on the resources. Here, the key "resource" is the available bandwidth at a queuing point (e.g., an output port). Hence, the ERICA switch algorithm is applied to each output port (or link). In this section, we present the basic features of the original algorithm and explain their operation. The next sections describe some problems and additions to the algorithm, and a new method to determine the number of active connections that tackles those problems. For a more complete description of the algorithm and its performance, refer to [10].

The ERICA switch periodically monitors the load on each link and determines a load factor, z, the available capacity, and the number of currently active virtual connections (VCs). The load factor is calculated as the ratio of the measured input rate at the port to the target capacity of the output link.

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

where:
ABR Capacity$\leftarrow$Target Utilization $\times$ Link Bandwidth $-$ VBR Usage $-$ CBR Usage.

The Input Rate is measured over an interval called the switch measurement interval. The above steps are executed at the end of the switch measurement interval.

Target utilization is a parameter which is set to a fraction (close to, but less than 100%) of the available capacity.

The load factor, z, is an indicator of the congestion level of the link. High overload values are undesirable because they indicate excessive congestion; so are low overload values which indicate link underutilization. The optimal operating point is at an overload value equal to one.

The fair share of each VC, $FairShare$, is also computed as follows:

$$\text{FairShare} \leftarrow \frac{\text{ABR Capacity}}{\text{Number of Active Connections}}$$

The switch allows each connection sending at a rate below the $FairShare$ to rise to $FairShare$. If the connection does not use all of its $FairShare$, then the switch fairly allocates the remaining capacity to the connections which can use it. For this purpose, the switch calculates the quantity:

$$\text{VCShare} \leftarrow \frac{CCR}{z}$$

If all VCs changed their rate to their $VCShare$ values then, in the next cycle, the switch would experience unit overload (z equals one). $VCShare$ aims at bringing the system to an efficient operating point, which may not necessarily be fair. A combination of the $VCShare$ and $FairShare$ quantities is used to rapidly reach optimal operation as follows:

$$\text{ER Calculated} \leftarrow \text{Max (FairShare, VCShare)}$$

Sources are allowed to send at a rate of at least $FairShare$ to ensure minimum fairness between connections. If the $VCShare$ value is greater than the $FairShare$ value, the source is allowed to send at $VCShare$, so that the link is not underutilized. This step also allows an unconstrained source to proceed towards its max-min rate.

The calculated ER value cannot be greater than the ABR Capacity which has been measured earlier. Hence, we have:

$$\text{ER Calculated} \leftarrow \text{Min (ER Calculated, ABR Capacity)}$$

To ensure that the bottleneck ER reaches the source, each switch computes the minimum of the ER it has calculated as above and the ER value in the RM cell, and indicates this value in the ER field of the RM cell.

The algorithm described above is the main algorithm, but several other steps are carried out to avoid transient overloads and variations in measurement, and drain the transient queues. Moreover, the algorithm is modified to achieve max-min fairness as described in sections 4 and 5.

3 The Measurement Interval

ERICA measures the required quantities over consecutive intervals and uses the measured quantities in each interval to calculate the feedback in the next interval. The length of the measurement interval limits the amount of variation which can be eliminated. It also determines how quickly the feedback can be given to the sources, because ERICA gives the same feedback value per source during each measurement interval. Longer intervals produce better averages, but slow down the rate of feedback. Shorter intervals may result in more variation in measurements, and may consistently underestimate or overestimate the measured quantities.

The ERICA algorithm estimates the number of active VCs to use in the computation of the fair share by considering a connection active if the source sends *at least one cell during the measurement interval.* This can be **inaccurate** if the source is sending at a low rate and the measurement interval is short. Exponentially averaging the number of active connections over successive intervals produces more accurate estimates, but may still underestimate the number of connections if the measurement interval is short. In this contribution, we propose a better method for estimating the number of active connections. The new method is not so sensitive to the length of the measurement interval. It also eliminates the need to perform some of the steps of the ERICA algorithm, as described in the next section.

4 The Fairness Problem and ERICA Solution

Assuming that the measurements do not exhibit high variation, and the measurement interval is long enough to estimate the number of VCs, the load factor and the available capacity, the original ERICA algorithm converges to efficient operation in all cases. The convergence from transient conditions to the desired operating point is rapid, often taking less than a round trip time.

However, we have discovered cases in which the original algorithm does not converge to max-min fair allocations. This happens if all of the following three conditions are met:

The load factor z becomes one
There are some connections which are bottlenecked upstream
The source rate for all remaining connections is greater than the $FairShare$

If this happens, then the system remains in its current state, because the term CCR/z is greater than $FairShare$ for the non-bottlenecked connections. This final state may or may not be fair in the max-min sense.

This problem was overcome in ERICA as follows. The algorithm is extended to remember the highest allocation made during each measurement interval, and ensure that all eligible connections can also get this high allocation. To do this, we add a variable $MaxAllocPrevious$ which stores the maximum allocation given in the previous interval, and another variable $MaxAllocCurrent$ which accumulates the maximum allocation given during the current switch measurement interval. Basically, for $z > 1 + \delta$, where δ is a small fraction, we use the basic ERICA algorithm and allocate Max (FairShare, VCShare). But, for $z \leq 1 + \delta$, we attempt to make all the rate allocations equal. We calculate the ER as Max (FairShare, VCShare, MaxAllocPrevious).

The key point is that the $VCShare$ is only used to achieve efficiency. The fairness can be achieved only by giving the contending connections equal rates. Our solution attempts to give the connections equal allocations during underload and then divide the (equal) CCRs by the same z during the subsequent overload to bring them to their max-min fair shares. The system is considered to be in a state of overload when its load factor, z, is greater than $1 + \delta$. The aim of introducing the quantity δ is to force the allocation of equal rates when the overload is fluctuating around unity, thus avoiding unnecessary rate oscillations.

The remainder of this contribution proposes a more accurate method to compute the max-min fair shares for all the contending connections, while avoiding the excessive sensitivity to the length of the measurement interval discussed in the previous section.

5 An Accurate Method to Determine the Fair Bandwidth Share

As previously discussed, ERICA determines the number of active connections by considering a source as active if at least one cell from this source is sent during the measurement interval. A more accurate method to compute activity and eliminate the need for the proposed solution to the fairness problem is to compute a quantity that we call the "effective number of active VCs" and use this quantity to compute the $FairShare$, as described next.

5.1 Basic Idea

We redefine the $FairShare$ quantity to be ***the maximum share a VC could get at this switch under max-min fairness criteria***. Hence, the $FairShare$ is calculated as follows:

$$\text{FairShare} = \frac{\text{ABR capacity}}{\text{Effective number of active VCs}}$$

The main innovation is the computation of the effective number of active VCs. The value of the effective number of active VCs depends on the activity level of each of the VCs. The activity level of a VC is defined as follows:

$$\text{Activity level} = \text{Min}(1, \frac{\text{Source Rate}}{FairShare})$$

Thus, VCs that are operating at or above the $FairShare$ are each counted as one. The VCs that are operating below the $FairShare$ (and are probably not bottlenecked at this switch) only contribute a fraction. The VCs that are bottlenecked at this switch are considered fully active while other VCs are considered partially active.

The effective number of active VCs is the sum of the activity levels for all VCs:

$$\text{Effective number of active VCs} = \sum_i \text{Activity level of VC}_i$$

Note that the definition of activity level depends upon the $FairShare$, and the definition of the $FairShare$ depends upon the activity levels. Thus, the definitions are recursive. Ideally, we would need to iterate several times given the source rates of various VCs.

5.2 Examples of Operation

Example 1 (stability):

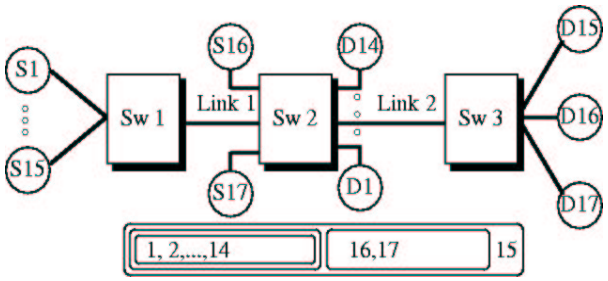


Figure 1: Upstream Configuration

Consider the upstream bottleneck case with 17 VCs shown in figure 1. It has been shown in [9] that this configuration demonstrates the unfairness of the original ERICA algorithm as described in section 2, which necessitates the addition described in section 4.

Assume that the target capacity is 150 Mbps. For the second switch, when the rates for ($S1$, $S16$, $S17$) are (10, 70, 70):

Iteration 1: FairShare = 70 Mbps
Activity = (10/70, 70/70, 70/70) = (1/7, 1, 1)
Effective number of active VCs = 1 + 1 + 1/7 = 15/7
Iteration 2: FairShare = Target capacity/Effective number of active VCs = 150/2.14 = approximately 70 Mbps

Hence, this example shows that the system is stable at the allocation of (10, 70, 70). At any other allocation, the scheme will calculate the appropriate $FairShare$ that makes the allocation eventually reach this point, as seen in the next two examples.

Example 2 (rising from a low FairShare):

For the same configuration, when the rates are (10, 50, 90):

Assume that the Effective number of active VCs = 3
Iteration 1: FairShare = 150/3 = 50 Mbps
Activity = (10/50, 50/50, 1) = (0.2, 1, 1)
Effective number of active VCs = 0.2 + 1 + 1 = 2.2
Iteration 2: FairShare = 150/2.2 = approximately 70 Mbps

Again, the scheme reaches the optimal allocation within a few round trip times.

Example 3 (dropping from a high FairShare):

For the same configuration, when the rates are (10, 50, 90), suppose that the effective number of active VCs is initially 2:

Iteration 1: FairShare = 150/2 = 75 Mbps
Activity = (10/75, 50/75, 1) = (0.13, 0.67, 1)
Effective number of active VCs = 0.13 + 0.67 + 1 = 1.8
Iteration 2: FairShare = 150/1.8 = 83.33 Mbps

Suppose the sources start sending at the new rates, except for the first one which is bottlenecked at 10 Mbps. Also assume that FairShare is still at 83.33 Mbps.

Activity = (10/83.33, 83.33/83.33, 83.33/83.33) = (0.12, 1, 1)
Effective number of active VCs = 0.12 + 1 + 1 = 2.12
FairShare = 150/2.12 = approximately 70 Mbps

Again, the scheme reaches the optimal allocation after the sources start sending at the specified allocations, which is within a few round trip times.

5.3 Derivation

The following derivation shows how we have verified the correctness of our method of calculation of the number of active connections. The new algorithm is based upon some of the ideas presented in

the MIT scheme [2, 3, 4]. However, this algorithm does not suffer from the known drawbacks of the MIT scheme (the high complexity, possible underutilization, and insensitivity to queuing delay).

The derivation depends on classifying active VCs as either underloading VCs or overloading VCs. A VC is *overloading* if it is bottlenecked at this switch; otherwise the VC is said to be *underloading*. In the MIT scheme, a VC is determined to be overloading by comparing the computed $FairShare$ value to the desired rate indicated by the VC source. In our scheme, we classify a VC as overloading if its source rate is greater than the $FairShare$ value. Our algorithm only performs one iteration every measurement interval, and is not of the complexity of the order of the number of VCs, as with the MIT scheme.

The MIT scheme has been proved to compute max-min fair allocations for connections within a certain number of round trips (see the proof in [3]). According to the MIT scheme:

$$FairShare = \frac{\text{ABR Capacity} - \sum_{i=1}^{N_u} Ru_i}{N - N_u}$$

where:

Ru_i = Rate of i^{th} underloading source ($1 \leq i \leq N_u$)
N = Total number of VCs
N_u = Number of underloading VCs

Substituting N_o for the denominator term, this becomes:

$$\text{FairShare} = \frac{\text{ABR Capacity} - \sum_{i=1}^{N_u} Ru_i}{N_o}$$

where:

N_o = Number of overloading VCs ($N_u + N_o = N$)

Multiplying both sides by N_o, we get:

$$FairShare \times N_o = \text{ABR Capacity} - \sum_{i=1}^{N_u} Ru_i$$

Adding $\sum_{i=1}^{N_u} Ru_i$ to both sides produces:

$$FairShare \times N_o + \sum_{i=1}^{N_u} Ru_i = \text{ABR Capacity}$$

Factoring $FairShare$ out in the left hand side:

$$FairShare \times (N_o + \sum_{i=1}^{N_u} \frac{Ru_i}{FairShare}) = \text{ABR Capacity}$$

Or:

$$FairShare = \frac{\text{ABR Capacity}}{N_o + \sum_{i=1}^{N_u} \frac{Ru_i}{FairShare}}$$

Substituting N_{eff}, we get:

$$FairShare = \frac{\text{ABR Capacity}}{N_{eff}}$$

where:

$$N_{eff} = N_o + \sum_{i=1}^{N_u} \frac{Ru_i}{FairShare}$$

This means that the effective number of active VCs is equal to the number of overloading sources, plus the fractional activity of underloading sources. This is the key equation we have proposed above, and implemented as discussed in the next subsection.

5.4 Algorithm Pseudo-code

This section explains how the new algorithm was implemented and incorporated into the ERICA switch algorithm.

The following variables are introduced:

- N_{last}: Effective number of active VCs in the last measurement interval.
- $N_{current}$: Effective number of active VCs being accumulated for the current measurement interval.
- Activity: This array is maintained for each VC. It is set to one for overloading sources (an overloading source is a source whose CCR exceeds its $FairShare$ value). The activity of a VC is set to the fraction obtained from dividing the CCR of the VC by the $FairShare$ value in the case of underloading sources.
- FirstCellSeen: This is also maintained for each VC, and is only used to avoid the initialization effects of the VC. It is one bit that is set to one if the VC has shown any sign of activity; otherwise, it is set to zero.
- VCsSeen: The sum of the VCs whose FirstCellSeen flag is set. Also used to avoid initialization effects.

INITIALIZATION:

1. N_{last} = number of VCs set up
2. $FairShare$ = ABR Capacity/N_{last}
3. $N_{current} = 0$
4. VCsSeen = 0
5. FOR ALL VCs DO
   Activity [VC] = 0
   FirstCellSeen [VC] = 0
   END (* FOR *)
6. Initialize other ERICA variables

END OF MEASUREMENT INTERVAL:

1. IF (VCsSeen >= N_{last})
   N_{last} = max (1, $N_{current}$)
   END (* IF *)

2. $N_{current} = 0$

3. $FairShare$ = ABR Capacity/N_{last}

4. FOR ALL VCs DO
   Activity [VC] = min (1, CCR [VC]/$FairShare$)
   $N_{current} = N_{current}$ + Activity [VC]
   END (* FOR *)

5. Update Overload Factor, and update or reset other ERICA variables

CELL IS RECEIVED IN FORWARD DIRECTION:

1. Do NOT update $N_{current}$ as used to be done with ERICA

2. IF (NOT FirstCellSeen [VC]) THEN
   FirstCellSeen [VC] = 1
   VCsSeen = VCsSeen + 1
   END (* IF *)

3. Update CCR [VC]

BRM CELL TO BE SENT IN REVERSE DIRECTION:
ER in BRM cell = Max (FairShare, CCR [VC]/Overload Factor)

Observe that the FirstCellSeen array and the VCsSeen counter are only used for the purpose of removing initialization effects from the simulation, and will not exist in a real implementation. Thus, in a real implementation, no steps (other than source rate estimation) will be carried out when a cell is seen, which means that the algorithm will have a low complexity.

6 Performance Analysis

The new algorithm has been tested for a variety of networking configurations using several performance metrics. The results were similar to the results obtained with the ERICA algorithm [8], except that the new algorithm is max-min fair (without executing the max-min fairness step described in section 4 above), and also the algorithm is less sensitive to the length of the measurement interval. A sample of the results is described in this section.

6.1 Parameter Settings

Throughout our experiments, the following parameter values are used:

1. All links have a bandwidth of 155.52 Mbps.

2. All links are 1000 km long.

3. All VCs are bidirectional.

4. The source parameter Rate Increase Factor (RIF) is set to one, to allow immediate use of the full explicit rate indicated in the returning RM cells at the source.

5. The source parameter Transient Buffer Exposure (TBE) is set to large values to prevent rate decreases due to the triggering of the source open-loop congestion control mechanism. This was done to isolate the rate reductions due to the switch congestion control from the rate reductions due to TBE.

6. The switch target utilization parameter was set to 90%. This factor is used to scale down the ABR capacity term used in the ERICA algorithm.

7. The switch measurement interval was set to the minimum of the time to receive 100 cells and 1 ms.

8. All sources are deterministic, i.e., their start/stop times and their transmission rates are known.

6.2 Simulation Results

The simulations performed focus on two main aspects of the new scheme: its fairness, and its transient response.

6.2.1 Fairness

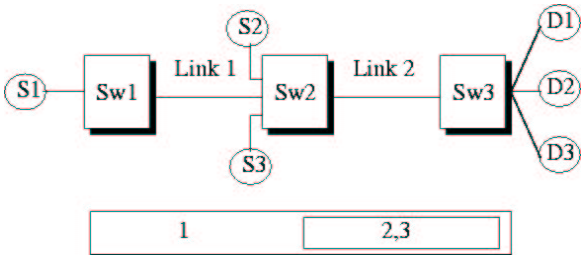


Figure 2: Three source configuration

In order to test fairness, we simulated a three source configuration where one of the sources is bottlenecked at a low rate (10 Mbps). Hence, even though the network gives that source feedback to increase its rate, it never sends at a rate faster than 10 Mbps. The other two sources start transmission at different ICR values. The aim of the configuration is to examine whether the two non-bottlenecked sources will reach the same ACR values, utilizing the bandwidth left over by the first source.

Figure 2 illustrates the configuration simulated. Note that the round trip time for the $S2$ and $S3$ connections is 30 ms, while that for the $S1$ connection is 40 ms. This configuration is almost identical to the one used in the examples in section 5 (figure 1), except that connection $S1$ to $D1$ is bottlenecked at the source $S1$ itself, and not at "Link 1." The reason we chose to demonstrate a source bottleneck situation here (and not a link bottleneck situation like figure 1) is to demonstrate the effect of using the CCR field in the RM cells versus measuring the source rate.

The results are presented in the form of three graphs for each configuration:

1. Graph of allowed cell rate (ACR) in Mbps over time for each source.
2. Graph of ABR queue lengths in cells over time at the bottleneck port.
3. Graph of the effective number of active VCs N_{eff} at the bottleneck port.

Figure 3 illustrates the performance of the original ERICA algorithm without the fairness step discussed in section 4. Source $S1$ is the bottlenecked source. Sources $S2$ and $S3$ start sending at different ICR (and hence ACR) values. Their ICR values and that of $S1$ add up to little more than the the link rate, so there is little overload. Observe that the rates of $S2$ and $S3$ remain different, leading to unfairness. The number of active VCs is counted using the original ERICA method, so the switch sees 3 sources (see figure 3(c)), and the $FairShare$ value remains at around 50 Mbps. Hence, the source $S2$ never increases its rate to make use of the bandwidth left over by $S1$ and only $S3$ utilizes this bandwidth.

Figure 4 illustrates how the fairness problem was overcome in ERICA by the change described in section 4. In this case, the sources are given the maximum allocation in case of underload or unit load, and hence all sources get an equal allocation. The modified algorithm is max-min fair.

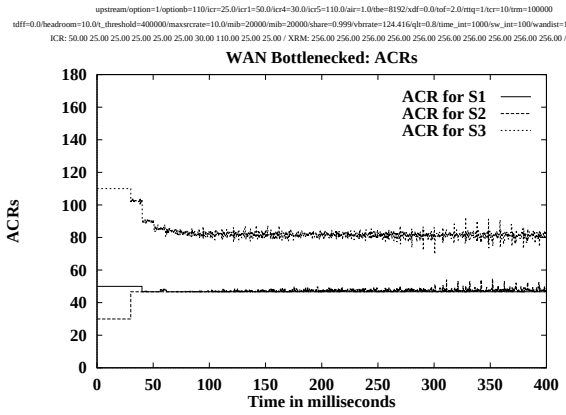
Figure 5 illustrates the results with the new method to calculate the fair share of the bandwidth. Observe that the allocations are max-min fair in this case, without needing to apply the maximum allocation algorithm as in the previous case. This is because the method of calculation of the effective number of active connections is different. Figure 5 shows that after the initialization period, the effective number of active VCs stabilizes at 1 (for $S2$), plus 1 (for $S3$), plus 10/50 (for $S1$), which gives $1+1+0.2 = 2.2$ sources. The method also stabilizes to the correct number even *if the length of the measurement interval is short*, unlike the original method where the length of the measurement interval must be long enough to detect cells from all sources, even low-rate sources.

The proposed method works correctly for all cases when there are *link bottlenecks* at various locations (e.g., the configuration in figure 1), since it correctly calculates the activity level of each connection based on its CCR value. However, observe that in *source bottleneck* cases, the CCR value cannot be simply obtained from the forward RM cells, but must be measured by the switches. This is because, in source bottleneck situations, the source indicates its ACR value in the CCR field of the RM cell, but the source may actually be sending at a much lower rate than its ACR.

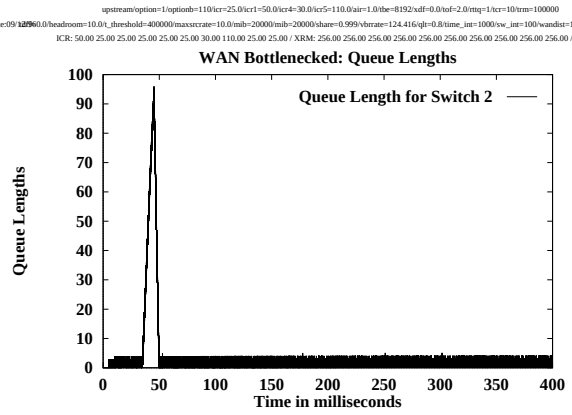
For example, for the configuration discussed above (figure 2), assume that we were relying on the CCR values in the RM cells. Figure 6 shows that the new method is not fair in this case, since source $S1$ indicates an ACR of 50 Mbps so the effective number of active connections stabilizes at 3 (see figure 6(c)), and the $FairShare$ remains at 50 Mbps. But source $S1$ is only sending at 10 Mbps. CCR measurement at the switch detects this, and hence arrives at the correct allocation as seen in figure 5.

6.2.2 Transient Response

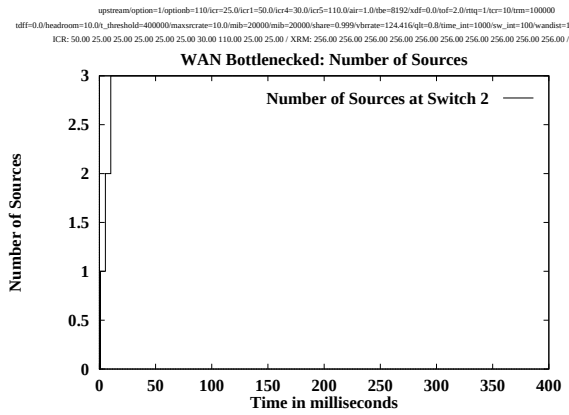
Figure 7 illustrates a two-source configuration. The round trip time for each connection is 30 ms. The new algorithm was simulated for this configuration, where the first source is active throughout the simulation period, while the second source starts sending after 60 ms and stops sending data at



(a) Allowed Cell Rate

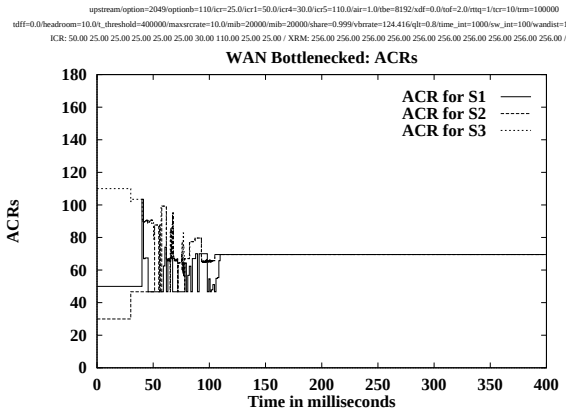


(b) Queue Length

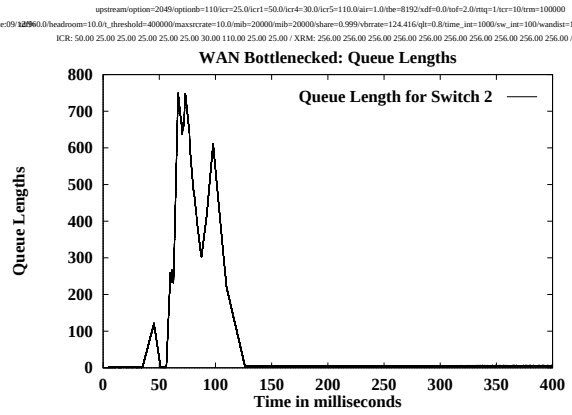


(c) Number of Active Connections

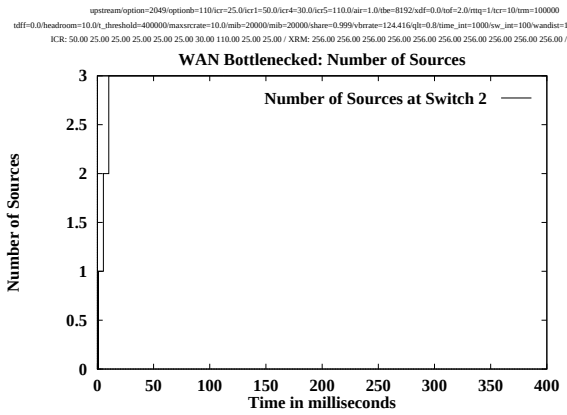
Figure 3: Results for a WAN three source bottleneck configuration with the original ERICA



(a) Allowed Cell Rate

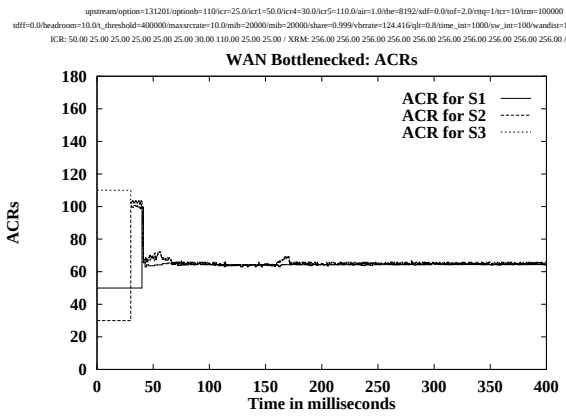


(b) Queue Length

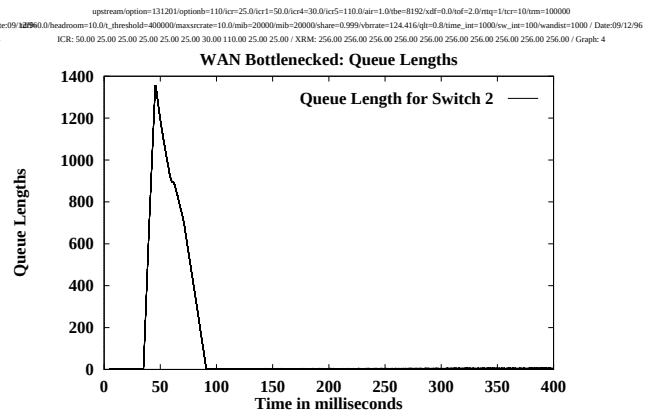


(c) Number of Active Connections

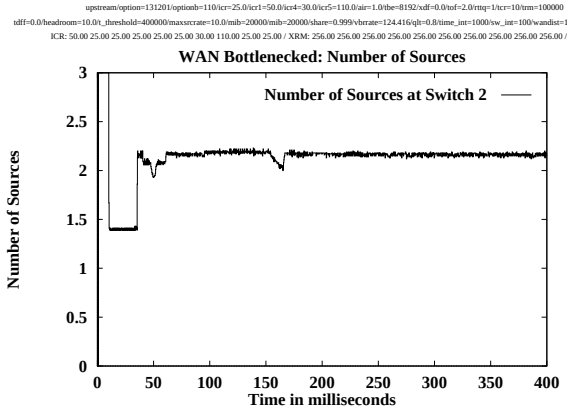
Figure 4: Results for a WAN three source bottleneck configuration with ERICA



(a) Allowed Cell Rate

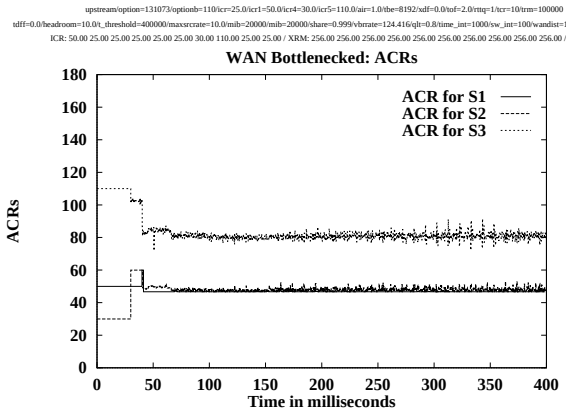


(b) Queue Length

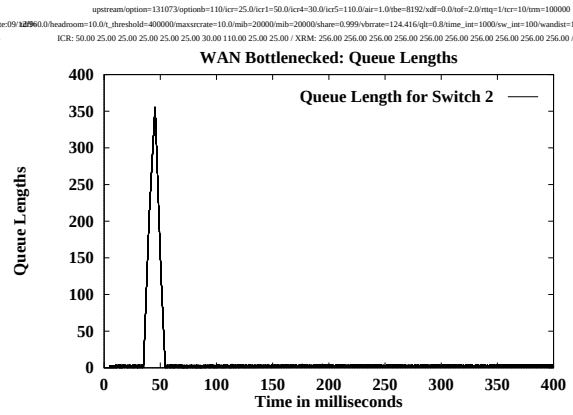


(c) Number of Active Connections

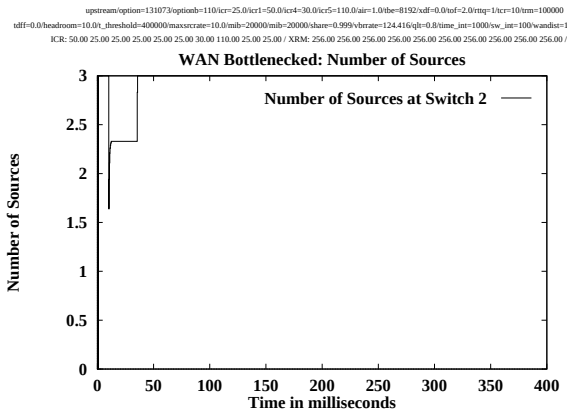
Figure 5: Results for a WAN three source bottleneck configuration with the proposed ERICA and source rate measurement at the switch



(a) Allowed Cell Rate



(b) Queue Length



(c) Number of Active Connections

Figure 6: Results for a WAN three source bottleneck configuration with the proposed ERICA

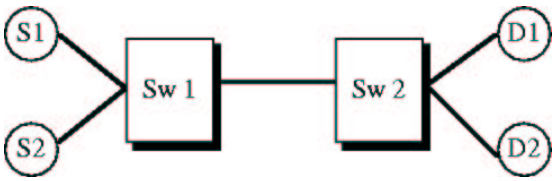
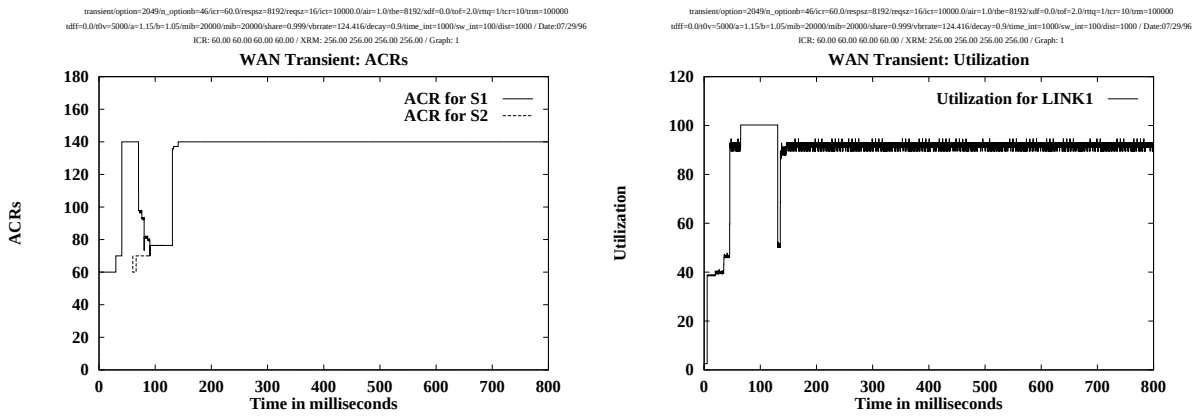


Figure 7: Two source configuration



(a) Allowed Cell Rate

(b) Link Utilization

Figure 8: Results for a WAN transient configuration with ERICA

120 ms. Both sources are persistent sources while they are on. The purpose of this configuration is to see if the proposed method has a slower transient response due to its recursive operation.

The results are presented in the form of two graphs for each configuration:

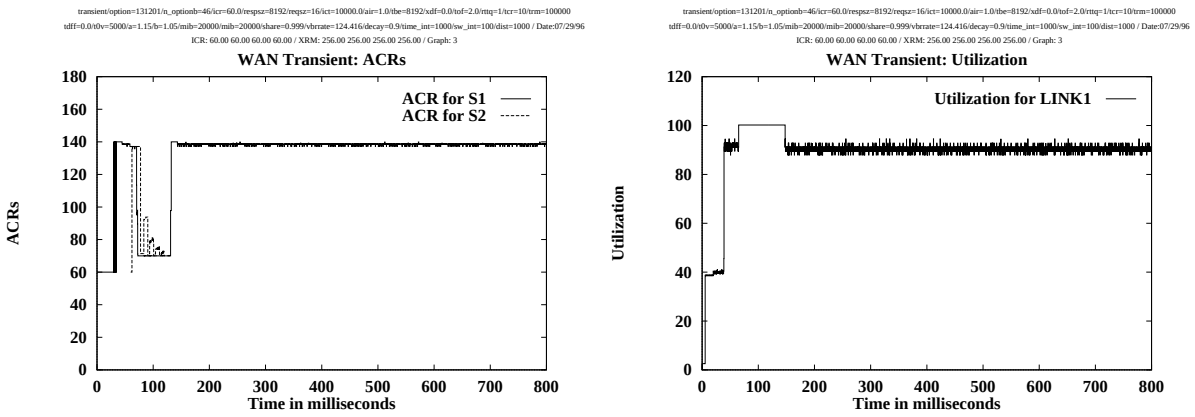
1. Graph of allowed cell rate (ACR) in Mbps over time for each source.
2. Graph of link utilization (as a percentage) over time for the bottleneck link.

Figures 8 and 9 show the performance of the ERICA algorithm (with the fairness modification) versus the performance of the proposed algorithm. It is clear that the transient response of both methods is comparable. The new method is slightly slower to reduce the rates in the start-up period of the second source, due to the recursive nature of the algorithm. However, the difference is small, and the benefits of the method far outweigh the slower response.

6.3 Observations on the Results

From the simulation results, we can make the following observations about the performance of the proposed algorithm:

- During transient phases, if the $FairShare$ value increases, the N_{eff} value decreases (since it uses the $FairShare$ value in the denominator), and $FairShare$ further increases (since it uses N_{eff} in the denominator), so N_{eff} further decreases, and so on, until the correct values



(a) Allowed Cell Rate

(b) Link Utilization

Figure 9: Results for a WAN transient configuration with the proposed ERICA and source rate measurement at the switch

of source rate, N_{eff} and $FairShare$ are reached. Then the proposed scheme is provably fair and efficient in steady state (see figure 5(a) and (c)).

- Using very small measurement interval values results in more problems for the original ERICA scheme than with the proposed scheme, because the proposed scheme does not measure the effective number of active connections by observing if cells are received from that connection during the measurement interval. Hence, even if the measurement interval is so short such that no cells are seen from many low-rate sources, the proposed method can compute the $FairShare$ of the bandwidth correctly.

- Without source rate measurement at the switch for each VC, the value of N_{eff} depends on the source ACR, which is not the same as the source rate for source bottleneck cases. Thus, N_{eff} is too large in those cases, and the $FairShare$ term is less than the CCR by Overload term, leading to unfairness. With per-VC source rate measurement, the value of N_{eff} is correct.

7 Summary

This contribution has proposed and demonstrated a new method to compute the fair bandwidth share for ABR connections in ATM networks. The method relies on distinguishing among underloading connections and overloading connections, and computing the value of the "effective number of active connections." The available bandwidth is divided by the effective number of active connections to obtain the fair bandwidth share of each connection.

The method is provably max-min fair, and can be used to ensure the efficiency and fairness of bandwidth allocations. Integrating this method into ERICA tackles the fairness and measurement interval problems of ERICA, while maintaining the fast transient response, queuing delay control, and simplicity of the ERICA scheme.

Analysis and simulation results were used to investigate the performance of the method. From the results, it is clear how the method overcomes the fairness problem with the original ERICA, as well as its excessive sensitivity to the length of the measurement interval.

References

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

[2] A. Charny, D. Clark, and R. Jain. Congestion Control with Explicit Rate Indication. In *Proceedings of ICC'95*, page 10 pp, June 1995.

[3] Anna Charny. An algorithm for rate allocation in a cell-switching network with feedback. Master's thesis, Massachusetts Institute of Technology, May 1994.

[4] Anna Charny, David D. Clark, and R. Jain. Congestion Control with Explicit Rate Indication. ATM Forum/94-0692, July 1994.

[5] Sonia Fahmy, Raj Jain, Shivkumar Kalyanaraman, Rohit Goyal, and Bobby Vandalore. On determining the fair bandwidth share for ABR connections in ATM networks. In *Proceedings of the IEEE International Conference on Communications (ICC) '98*, June 1998.

[6] Y. Gong and I. Akyildiz. Dynamic traffic control using feedback and traffic prediction in ATM networks. In *Proceedings of the IEEE INFOCOM '94*, pages 91–98, June 1994.

[7] Jeffrey M. Jaffe. Bottleneck flow control. *IEEE Transactions on Communications*, COM-29(7):954–962, July 1981.

[8] R. Jain, S. Fahmy, S. Kalyanaraman, and R. Goyal. ABR Switch Algorithm Testing: A Case Study with ERICA. ATM Forum/96-1267, October 1996.

[9] R. Jain, S. Fahmy, S. Kalyanaraman, and R. Goyal. The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks, Part II: Requirements and Performance Evaluation. Submitted to IEEE/ACM Transactions on Networking, January 1997.

[10] R. Jain, S. Kalyanaraman, R. Goyal, S. Fahmy, and R. Viswanathan. ERICA Switch Algorithm: A Complete Description. ATM Forum/96-1172, August 1996.

[11] N. Yin and M. G. Hluchyj. On closed-loop rate control for ATM cell relay networks. In *Proceedings of the IEEE INFOCOM '94*, pages 99–108, June 1994.

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