A Simple Approach to Congestion Control of ABR Traffic with Weighted Max-Min Fairness

ROHAN DE SILVA
School of Information Systems, Technology & Management
University of New South Wales
Sydney 2052
AUSTRALIA
http://www.sistm.unsw.EDU.AU/people/rdesilva/home/

Abstract: This paper describes a simple algorithm called SAFARI (Simple Approach to Fair Allocation of Rate Indication) for the congestion control of available bit rate (ABR) service of asynchronous transfer mode (ATM) networks. The ATM Forum specifies a category of switches known as explicit rate (ER) switches that computes an ER for each source to satisfy a fairness criterion. To compute ER, an ATM switch needs to measure a number of traffic parameters on-line. There are a number of congestion control schemes for ABR service but SAFARI scheme is less complex compared to both the load factor-based schemes that require the measurement of many variables and the queue-level-based schemes that require much computation. As opposed to other load factor-based schemes, we measure only the number of virtual circuits, target rate and the load on the switch. This eliminates the need for the maintenance of any table of rates or the reading of rates off the forward resource management cells. Furthermore, as opposed to our method, the computational complexity of other schemes increases considerably when used for weighted max-min fairness criterion with non-zero minimum cell rate (MCR). Performance of this scheme has been studied via simulations on a GFC-II network.

Key-Words: ATM networks, ABR service, Congestion control, Convergence

1 Introduction
Congestion control is important for the successful operation and optimum exploitation of the available resources of asynchronous transfer mode (ATM) networks. Without a proper congestion control mechanism in place, the aggregate traffic from different users can exceed the capacity of the network, leading to the buffer overflow in intermediate nodes or switches. This results in loss of cells and even the loss of a single cell in ATM can cause the retransmission of a large chunk of data.

Before sending traffic over an ATM network, the source has to establish a virtual circuit (VC) connection through a set of switches to the destination. At the time of connection establishment, the switches also agree to satisfy the traffic contract of the connection specified by the quality of service (QoS) parameters. Each switch has a first-in-first-out (FIFO) queue associated with each outgoing link. Generally, voice and video traffic have strict delay and jitter requirements, and are given higher priority for transmission at the network switches. Data traffic can experience some delay and this property enables us to store the data cells temporarily in the queue of an intermediate switch and forward them to the next switch when enough bandwidth is available in the outgoing link. However, the buffers of the intermediate switches have finite capacities and the number of cells that is stored at a given instant should be maintained below a certain limit.

ATM Forum has developed the Traffic Management Specification for available bit rate (ABR) service [1] which stipulates that the congestion control schemes for ABR service should be closed-loop and rate-based. The ATM Forum has defined special 53-octet control cells called the resource management (RM) cells to convey information to the sources about the state of congestion of the network. The source generates forward RM (FRM) cells which are turned around by the destination and sent back to the source as backward RM (BRM) cells. These BRM cells carry feedback information provided by the switches and/or the destination back to the source. RM cells contain a field called the explicit rate (ER) field and a field called the current cell rate (CCR) field, which is set by the source to its current cell delivery rate when it generates an FRM cell. The source regularly sends to the destination an FRM cell after sending a certain number of data cells.
Furthermore, the standard specifies a category of switches known as ER switches that computes an ER for each source to satisfy a fairness criterion. An ER switch calculates the ER using a certain congestion control algorithm and replaces the ER value in BRM cells if the former is less than the latter. To compute the ER, the switch may need to measure a number of traffic parameters on-line. The exact form of the congestion control algorithm that an ER switch should employ has not been specified by the ATM Forum but has been left to be determined by the vendors.

As a result, a countless number of rate-based congestion control schemes have mushroomed during the latter part of the last decade but only a few of them have taken new directions. As it is described in Section 2, these schemes involve either complex computations or require the measurement of many variables.

The scheme that we introduce and discuss in this paper takes a different approach. The primary goal in our design is to reduce the number of variables that should be measured and to minimise the computation burden. Thus, our scheme is less complex compared to both the load factor-based schemes that require the measurement of many variables and the queue-level-based schemes that require much computation.

The rest of the paper is organised as follows. After discussing background and the research problem in Section 2, we discuss the fairness criteria and develop our basic ABR controller in Section 3. Then, we extend the basic algorithm to the case of weighted max-min fairness criterion. In section 4, after discussing the closed-loop system of the basic controller, we introduce dynamic QCFs. Simulations and the analysis of the results is found in section 5. The paper is concluded in section 5.

2 Background

Fundamentally, two pioneer research contributions that took different directions attempted to solve the rate-based congestion control problem by calculating the ER at the switches. One group used the queue level [2] and most of the initial ABR rate-based congestion control schemes were based on this parameter. In this approach, the queue level is measured and the switch calculates the ER using this queue level information. These schemes were designed using formal control theory, and as a result, the convergence of the queue level to a desired level could be proved analytically. Early versions and designs of this approach used a single buffer and included the case of a single congested node [2], multiple congested nodes [3] and the Smith predictor [4]. These designs assumed that the feedback delay from the source to the switch is known and fixed, but the feedback delay changes with the queue build up in upstream switches. A second problem with these schemes was that they required the careful selection of the controller parameters. To eliminate the problem of changing feedback delay, hop-by-hop control [5] was introduced. Some designs of this category employed per-VC queuing and scheduling [5, 6]. Per-VC queuing and scheduling requires the cells from each VC to be stored in a logically separate partial buffer. However, this approach requires expensive switches and has the problems of scalability and the selection of controller parameters. The number of additions required in the computation of ER is usually in the order of the number of VCs sharing the outgoing link that carry the ABR traffic and therefore, these schemes are not scalable. There are many queue level-based congestion control schemes that employ advanced control techniques such as estimation techniques to tune parameters on-line [7] and non-linear and adaptive control to handle the problems of saturation of the source rates and the queue levels [8]. But unfortunately, they increase the computation burden and in some cases, reduce the scalability as well. An interesting variant of queue level-based control is presented in [9]. This method uses a proportional controller where the fair share is equal to a fraction of the error in queue level plus the previous fair share divided by the load factor. The problem with this controller is that at steady state, the fair share may not be equal to the correct fair share if the controller parameter is not selected dynamically.

The second approach of explicit rate calculation for the rate-based congestion control of ABR traffic is based on the load factor [10]. The load factor is a quantity that is defined as the ratio between the aggregate input rate and the target rate at the switch of all ABR virtual circuits that pass through the switch and share the same outgoing link. The target rate is the remaining capacity for ABR connections in the outgoing link after all other VCs carrying higher priority video and voice traffic have been allocated their required capacity. The interesting property of the load factor is that when it is equal to one, the input rate is equal to the target rate. The load factor-based schemes also have the ability to compute the explicit rate
according to a prescribed max-min fair share criterion and the early versions [10, 11] of these schemes did not need the selection of any parameter. However, it was later recognised that when the load factor is equal to one and the prescribed max-min fair share criterion is satisfied, the buffer level may be too high. This problem has been solved [12] by introducing a function called the queue control function (QCF) [13]. The idea of using a QCF is to reduce or increase the target rate when the queue level deviates from a desired operating point. Thus, the queue level is monitored and the QCF is used to calculate a target utilisation factor which modifies the target rate. If the queue level is higher than the desired level, this increases the load factor resulting in a decreased explicit rate. This process continues and finally when the load factor reaches one, it is expected that the queue level would reach the desired level. However, the use of a QCF has introduced the need for the measurement of queue level (an additional measurement) and the selection of measurement intervals and parameters in prescribed ranges. The other representative schemes that use the load factor-based approach are [14 - 19].

3 Development of the ABR Controller
We call our basic algorithm SAFARI (Simple Approach to Fair Allocation of Rate Indication). As opposed to other schemes, we measure only the number of VCs, target rate and the load on the switch. This eliminates the maintenance of any table or reading of CCR or ER values off all the FRM cells as, for example, with explicit rate indication algorithm (ERICA), fast max-min rate allocation algorithm (FMMRA) [14] and other load factor-based schemes [17, 18]. Our scheme relies on solving the max-min fairness equation (eq (1) of section 2.1) iteratively. The computational burden of this iterative solution has the complexity of order one (O(1)). This iteration on max-min fair share converges to the correct max-min fair share that is obtained by directly solving eq (1). Since the iteration has global convergence properties irrespective of the initial values and change of parameters such as the number of VCs and the target rate, the algorithm is robust. Since it can converge to the correct max-min fair share value from any initial condition, it can be employed in switches in heterogeneous environments as well.

As ABR congestion controllers are designed to achieve a prescribed max-min fairness, before introducing the development of our controller, we present the available max-min fairness criteria in the next subsection.

3.1 Max-min Fairness Criteria
ATM Forum [1] defines a number of max-min fair share criteria for the ABR service. In the basic max-min fair share criterion with all MCR values equal to zero, the available bandwidth B of n connections that are bottlenecked only at the link under consideration is equally shared among these connections. The fair share FS(i) of the i-th connection is given by:

\[
FS(i) = \frac{B}{n}.
\]

For the case where the fair share is equal to the MCR plus weighted max-min share, FS(i) is given by:

\[
FS(i) = MCR(i) + (B - M) \times \left(\frac{w(i)}{\sum w(j)}\right)
\]

where M, MCR(i) and w(i) are the sum of MCRs of active connections belong to n, MCR of connection i and weight of connection i respectively.

All other fair share criteria are special cases of the criterion described by eq (2).

3.2 Proposed Basic SAFARI Scheme
We present the iterative solution to max-min fairness criterion in eq (1) formally as Theorem I below.

Theorem I
Consider N ABR flows, with all MCR values equal to zero, that share a common FIFO buffer at an ABR switch. Some of these flows may be bottlenecked at upstream ATM switches or constrained at the sources. Assume that the target ABR rate of the outgoing link, the bottleneck rates of the upstream switches and the constrained rates of the constrained sources do not change. Assume also that the cells delivered by the sources at the CCR values equal to max-min fair share calculated in the previous iteration or at the constrained rates arrive at the switch before the next iteration. Under these conditions, the max-min fair share FS(t) calculated using the iteration in eq (3) converges to the correct max-min fair share FS*.

\[
FS(t+1) = FS(t) + AV(1 - z(t))
\]

with the initial condition

\[
FS(0) = C_{in}(0)/N
\]

and

\[
FS^* = \lim_{t \to \infty} FS(t)
\]
where \( C_{in}(t) \), \( C \) are the input rate and the target rate respectively and \( z(t) \) is the load factor given by
\[
z(t) = \frac{C_{in}(t)}{C} \quad (6)
\]
and \( AV \) is the average target rate of \( N \) flows of ABR traffic given by
\[
AV = \frac{C}{N} \quad (7)
\]
An interesting property of this iteration is that when the fair share converges to the correct max-min fair share, the load factor converges to 1. We formally state this property in a companion theorem:

**Theorem II**

If the iteration in eq (3) is performed on any number of \( N \) fixed ABR flows in the scenario specified in Theorem I, the load factor \( z(t) \) converges to 1.

We omit providing the analytical proofs of these two theorems here as they are fairly long. But they can be found in [20]

### 3.3 Modified SAFARI Scheme

Inspection of the iteration described by eq (3) shows that this basic algorithm cannot be used in the form given above in an ER switch unless the feedback delay is equal to one iteration step. To see this, let us assume that the fair share is computed and written into a BRM cell that passes through the switch in the backward direction. If the feedback delay (the delay from the time of calculation of fair share to the receiving of cells at this new rate by the switch) is greater than the length of the iteration cycle, the switch may go through many iteration cycles instead of just one during this time. As a result, the calculated fair share will not be correct.

One solution to this problem is to update the fair share calculated using eq (3) by only a fraction of \( AV(1-z(t)) \). The value of this fraction should be adjusted according to the feedback delay between the switch and the farthest source and the length of the measurement interval to obtain optimum performance. We formally state the modified iteration of the SAFARI algorithm as a corollary below:

**Corollary I**

For the scenario described in Theorem I, if
\[
FS(t+1) = \max \{FS(t) + kAV(1-z(t)), AV\}(8)
\]
with
\[
FS(0) = AV
\]
where \( 0 < k < 1 \), the calculated fair share converges to the correct max-min fair share. The proof of Corollary I can also be found in [20]. Thus, to stabilise the queue at a desired value, we have to use an explicit QCF together with this controller.

The QCFs that are found in the literature [13] assume that the queue level is measured. On the other hand, the load factor-based congestion control methods do not inherently require the measurement of queue level. A further problem with these QCFs is that they do not take into account the delay in the feedback. We introduce a predictive QCF (PQCF) that can be deduced from the rate control using one-step ahead predictor approach that has been developed and discussed in [6]. This approach takes into account the feedback delays from the switch to the sources and considers that the queue dynamics are the result of the source rates calculated at a time equal to one feedback delay earlier. In this predictive congestion control, when the queue level \( Q(k) \) converges to a desired queue level \( Q^* \), the input rate converges to the target rate. For a target rate \( C \), the slot time \( T \) and the quotient of maximum feedback delay divided by \( T \) equals to \( n \), we calculate \( f(k) \), the value of QCF (the utilisation factor) during \( k \)-th time slot as
\[
f(k) = f(k-n-1)+\beta f \left[ Q(k-1) - Q(k) \right]/ (T*C) \quad (9)
\]
where the parameter \( \beta f \) (\( 0 < \beta f < 1 \)) provides some damping.

If the queue level during a certain time slot is greater than the time integral of the target rate minus the input rate during the next time slot, the throughput is equal to the target rate, otherwise the throughput is less than the target rate. Thus, the load factor \( z(k) \) can be written as
\[
\begin{align*}
z(k) & \geq \frac{\text{input rate}}{\text{target rate}} \\
& \geq 1+\frac{Q(k)-Q(k-1)}{T*C} \quad (10)
\end{align*}
\]
From eqs (9) and (10), assuming that the queue has enough cells, we can write
\[
f(k) = f(k-n-1)-\beta f \left[ z(k) \right]/ (T*C) \quad (11)
\]

The PQCF introduced above needs the storage of the \( n+1 \) past values of the utilisation factor to compute the current utilisation factor. This slightly increases the memory required.

Fig 1 shows the closed-loop ABR control system incorporating the SAFARI controller. In Fig 1, \( q^{-1} \) is the backward shift operator and \( \tau_f \) is the feedback delay. The SAFARI algorithm described above acts as the controller and it is indicated in Fig 1 as the part inside the dotted rectangle.
3.4 Extension of SAFARI Algorithm to the Weighted Fair Share Criterion

Extension of the SAFARI algorithm to the weighted fair share case in eq (2) is straightforward. At the connection setup, the switch has to agree that it can provide the user with the requested MCR (MCR$_i$) and the weight (w$_i$) of the traffic contract. Each switch keeps a list of MCR values and the weights of all the VCs that pass through it. The switch calculates the total MCR by adding these individual MCR values. After determining the input rate and the available ABR capacity, the switch modifies these values by subtracting the total MCR from both of them before determining the load factor. The fair share calculation iteration is then performed at the arrival of a BRM cell. The calculated fair share is modified using the MCR of the particular VC and the weight of the VC to obtain the explicit rate which is written into the ER field of the BRM cell if it is less than the ER value in the cell.

In contrast, let us see how this modification affects the other algorithms such as ERICA or FMMRA where the CCR or ER values of each VC is needed for the calculation of fair share. Inspection of these algorithms show that each CCR or ER value read from the FRM cells have to be modified if these algorithm are to be adapted for the weighted fair share case. This modification includes the subtraction of the individual MCR value of each VC from the corresponding CCR or ER value before calculating the VC share. Obviously, when there is a large number of ABR VCs passing through the switch, this computation takes a considerable amount of processing time. But in the case of the SAFARI algorithm, we have to subtract only the total MCR from the input rate.

The pseudocode of SAFARI algorithm is given below:

**Initialisation:**
- total MCR ← Sum of all the MCRs
- w ← Sum of all weights w$_i$
- k ← measurement interval/maximum feedback delay
- j ← 0

**End of measurement Interval:**
- target rate ← Link Capacity – Non-ABR Capacity
- z ← (input rate – total MCR)/(target rate – total MCR)
- AV ← (target rate – total MCR)/Number of ABR VCs
- Calculate f using the equation of QCF
- FS ← max [{FS + kAV(f-z)}, AV]
- IF (j = 0)
- {FS ← (target rate – total MCR)/ Number of ABR VCs
- j ← j+1
- }

**A BRM cell arrives via the i-th VC:**
- ER in the cell ← min {ER in the cell, (MCR$_i$ + w$_i$*FS/w)}

4 Simulations

In this section, we discuss the simulation environment and the results. We investigated the performance of the controller to achieve the basic max-min fairness as well as weighted max-min fairness with non-zero MCR.

4.1 Simulation on a Generic Flow Control II (GFC-II) Network

Performance of the SAFARI algorithm was investigated using the GFC-II configuration used by many authors [12] and recommended by the ATM Forum [1]. The GFC-II configuration is shown in Fig 3. The simulations were performed using OPNET simulation tool. Because of the memory limitations, we had to scale down the link delay by a factor of 10. For the simulation purposes, we also selected a time slot equal 2.826 µs (transmission time of a cell). The sources are persistent and the theoretical CCR values at steady state are shown in Table 1.
The maximum feedback delay of this network is 3065.4 time slots. Since the minimum feedback delay is 235.8 time slots, the measurement interval (T) was selected as 235.8 time slots. The gain parameter k was selected as 0.08 (approximately 235.8/3065.4). Parameter $\beta_f$ of the QCF of eq (11) was selected as 1/T.

Table 1: Theoretical steady state CCR values for the GFC-II network

<table>
<thead>
<tr>
<th>Source#</th>
<th>Steady state CCR (cells/time slot)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{0}, S_{4}, S_{7}$</td>
<td>0.0666</td>
</tr>
<tr>
<td>$S_{1}, S_{6}, S_{9}$</td>
<td>0.0333</td>
</tr>
<tr>
<td>$S_{12}, S_{13}, S_{14}$</td>
<td>0.2333</td>
</tr>
<tr>
<td>$S_{2}$</td>
<td>0.2333</td>
</tr>
<tr>
<td>$S_{3}, S_{4}$</td>
<td>0.2333</td>
</tr>
<tr>
<td>$S_{8}$</td>
<td>0.0666</td>
</tr>
<tr>
<td>$S_{15}-S_{21}$</td>
<td>0.0333</td>
</tr>
<tr>
<td>$S_{10}, S_{11}$</td>
<td>0.35</td>
</tr>
</tbody>
</table>

The maximum queue level of this network is 4520 cells. Since the minimum feedback delay is 235.8 time slots, the measurement interval (T) was selected as 235.8 time slots. The gain parameter k was selected as 0.08 (approximately 235.8/3065.4). Parameter $\beta_f$ of the QCF of eq (11) was selected as 1/T.

Table 2: Maximum queue levels and CCR values for the simulation results shown in Fig 3

<table>
<thead>
<tr>
<th>Maximum Queue Levels (Cells)</th>
<th>Steady state CCR (Cells/time slot)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW0 456</td>
<td>$S_0$ 0.066</td>
</tr>
<tr>
<td>SW1 410</td>
<td>$S_1$ 0.033</td>
</tr>
<tr>
<td>SW2 286</td>
<td>$S_2$ 0.231</td>
</tr>
<tr>
<td>SW3 96</td>
<td>$S_3$ 0.231</td>
</tr>
<tr>
<td>SW4 336</td>
<td>$S_4$ 0.066</td>
</tr>
<tr>
<td>SW5 478</td>
<td>$S_{10}$ 0.347</td>
</tr>
<tr>
<td>SW7 116</td>
<td>$S_{12}$ 0.231</td>
</tr>
<tr>
<td>SW8 120</td>
<td>$S_{15}$ 0.033</td>
</tr>
</tbody>
</table>

Fig 3 shows that the CCRs of the sources converged to the correct max-min fair share in about 6000 time slots (approximately two round trip times). The queue levels dropped to near zero in about 22000 time slots (approximately seven round trip times). Maximum queue level (see Table 2) was around 4520 (452 scaled up by a factor of 10). The utilisation of all the links between the switches were hundred percent after one round trip time. For comparison, the simulation results reported in [12] with the ERICA algorithm for the same GFC-II network was two round trip times for the convergence of the fair share and seven round trip times for the convergence of the queue levels. However, the maximum queue level reported with ERICA was much higher and was about 28,000 cells.

Predictive control is better than proportional integral control or proportional control if the system has a time delay (see [21]). The only disadvantage of this QCF against the QCF of [43] is that we have to store the past values of f. The number of past values that should be stored is one plus the quotient of the feedback delay to the farthest source from the switch divided by the measurement interval T.
4.2 Simulation on the GFC-II network with MCRs and weights

To investigate the performance of the SAFARI algorithm for the case MCR plus weighted max-min share criterion, we simulated it on the modified GFC-II network. Since the GFC-II network source parameters do not include weights and MCR values, we selected a set of values for them. These values are shown in Table 3. The theoretical CCR values are shown in Table 4.

<table>
<thead>
<tr>
<th>Source#</th>
<th>MCR (cells/time slot)</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0, S4, S7</td>
<td>0.016667</td>
<td>1</td>
</tr>
<tr>
<td>S1, S6, S9</td>
<td>0.016667</td>
<td>2</td>
</tr>
<tr>
<td>S12, S13, S14</td>
<td>0.05</td>
<td>2</td>
</tr>
<tr>
<td>S2</td>
<td>0.05</td>
<td>3</td>
</tr>
<tr>
<td>S3, S4</td>
<td>0.033333</td>
<td>1</td>
</tr>
<tr>
<td>S8</td>
<td>0.033333</td>
<td>2</td>
</tr>
<tr>
<td>S15-S21</td>
<td>0.016667</td>
<td>3</td>
</tr>
<tr>
<td>S10, S11</td>
<td>0.066667</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3: MCR values and weights of the sources

<table>
<thead>
<tr>
<th>Source#</th>
<th>Steady state CCR (cells/time slot)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0, S4, S7</td>
<td>0.0565</td>
</tr>
<tr>
<td>S1, S6, S9</td>
<td>0.0253</td>
</tr>
<tr>
<td>S12, S13, S14</td>
<td>0.2516</td>
</tr>
<tr>
<td>S2</td>
<td>0.2513</td>
</tr>
<tr>
<td>S3, S4</td>
<td>0.2513</td>
</tr>
<tr>
<td>S8</td>
<td>0.1130</td>
</tr>
<tr>
<td>S15-S21</td>
<td>0.0253</td>
</tr>
<tr>
<td>S10, S11</td>
<td>0.3775</td>
</tr>
</tbody>
</table>

Table 4: Theoretical CCR values with MCR and weights of Table 4

The simulation results are shown in Fig 4.

In contrast to the basic max-min fairness case, here the maximum queue levels are lower (compare Tables 3 and 5). This may be due to the weights and the selected MCR values. The above simulation results show that we can use SAFARI controller to achieve any type of max-min fairness specified in [1].

5 Conclusion

There are two classes of ER ABR congestion control schemes, namely the load factor-based ER control schemes and the queue level-based ER control schemes. The latter require relatively more steps of computation than the former. The research discussed in this paper deals mainly with the load factor-based schemes.

The reading and the storage of CCR or ER values are needed in all well-known load factor-based explicit rate control schemes, such as in ERICA and FMMRA that use FIFO queuing. This requires considerable processing time and memory thus increasing the cost. In this paper, we have presented a load factor-based congestion control scheme that does not need to read and store the
CCR or ER values of incoming FRM cells. This elimination of storing CCR or ER values and also the development of a suitable iterative solution for fair share that globally converges to the correct fair share have greatly reduced the computational complexity and memory requirements of our scheme.

The load-factor based schemes have evolved over the years from their fundamental approach by including a static QCF to handle the queue level. However, this inclusion requires the value of queue level which is an additional measurement. By making the QCF to be dependant only on the load factor, we have eliminated this additional measurement. We have introduced a predictive QCFs that take into account the feedback delay unlike the static QCFs, which do not consider this factor. However, as the past values of the utilisation factor have to be stored, the implementation of this QCF slightly increases the memory requirements.

We have used OPNET simulation tool to verify the performance of this new load factor-based algorithm on a complicated GFC II network recommended by the ATM Forum. The simulation results are encouraging and demonstrate that the queue levels are much smaller than that have been reported with the ERICA algorithm for the same GFC II network.

We have also extended our algorithm to handle the case of weighted max-min fair share criterion with non-zero MCR values. It has been shown that with other schemes such as ERICA and FMMRA, adaptation to this case will involve per-VC calculations and hence increase the computational burden considerably. But in our algorithms the change can be accommodated with a few simple calculations.

Though there are several congestion controllers developed for the per-VC queueing and scheduling switches that are queue level-based they have the disadvantage of being more costly and complex than the single FIFO queueing. On the other hand, if only single FIFO queue switches are used to reduce this cost and complexity, we have doubts if any simpler and cheaper scheme than SAFARI can be constructed. The ABR congestion controller presented in this paper can also be adapted to the medium access (MAC) level flow control of 10 Gigabit Ethernet (10 GbE) easily if its standard allows us to include a field in the PAUSE frame (control frame) to carry ER. This idea of including an ER field in the PAUSE frame has already been proposed in [22].

References:


