ELSEVIER

Computer Communications 23 (2000) 149–161

computer communications

www.elsevier.com/locate/comcom

General weighted fairness and its support in explicit rate switch algorithms[☆]

B. Vandalore*, S. Fahmy[1], R. Jain, R. Goyal[2], M. Goyal

Department of Computer and Information Science, The Ohio State University, 2015 Neil Avenue Mall, Columbus, OH 43210-1277, USA

Received 24 February 1999; received in revised form 15 July 1999; accepted 30 July 1999

Abstract

This paper gives a new definition of general weighted (GW) fairness and shows how this can achieve various fairness definitions, such as those mentioned in the ATM Forum TM 4.0 specifications. The GW fairness can be achieved by calculating the *ExcessFairshare* (weighted fairshare of the left over bandwidth) for each VC. We show how a switch algorithm can be modified to support the GW fairness by using the *ExcessFairshare* term. We use ERICA+ as an example switch algorithm and show how it can be modified to achieve the GW fairness. For simulations, the weight parameters of the GW fairness are chosen to map a typical pricing policy. Simulation results are presented to demonstrate that, the modified switch algorithm achieves GW fairness. An analytical proof for convergence of the modified ERICA+ algorithm is given in the appendix. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: ATM switch algorithms; Pricing policy; ABR service; Traffic management

1. Introduction

The asynchronous transfer mode (ATM) is the chosen technology to implement Broadband Integrated Services Digital Network (B-ISDN). Different traffic characteristics ranging from non-real-time to real-time are supported in ATM through its various service categories (CBR: constant bit rate, rt-VBR: real-time variable bit rate, nrt-VBR: non-real-time VBR, ABR: available bit rate, UBR: unspecified bit rate). The ATM Forum is currently in the process of standardizing the Guaranteed Frame Rate (GFR) service category. The International Telecommunication Union (ITU-T) defines similar service categories for ATM.

The ABR service category is the only service category which uses closed-loop feedback for flow control. All other service categories have open loop flow control. In ABR, one Resource Management (RM) cell is sent for every $Nrm - 1$ (value of Nrm parameter is usually 32) data cells by the source. The source indicates its current source rate in the RM cell. The RM cell is turned around at the destination and sent back to the source (Fig. 1). The switches along the RM cell path indicate the current maximum rate, which they can support in the explicit rate field of the RM cell. The sources adjust their rates accordingly.

To get a guarantee for the minimum amount of service the user can specify a minimum cell rate (MCR) in ATM ABR service. The ABR service guarantees that the allowed cell rate (ACR) is never less than MCR. When MCR is zero for all sources, the available bandwidth can be allocated equally among the competing sources. This allocation achieves max–min fairness. When MCRs are non-zero, ATM Forum TM 4.0 specification [1] recommends, other definitions of fairness that allocate the excess bandwidth (which is, available ABR capacity less the sum of MCRs) equally among sources, or proportional to MCRs. In this paper, we give a different definition of sharing the excess bandwidth using predetermined weighted than the one recommended in ATM Forum specifications [1]. In the real world, the users prefer to get a service, which reflects the amount they are paying. The pricing policy requirements can be realized by appropriately mapping the policy to the weights associated with the sources.

The specification of the ABR feedback control algorithm (switch algorithm) is not yet standardized. The earliest algorithms used binary feedback techniques [2]. Distributed algorithms [3] that emulated a centralized algorithm were proposed in Refs. [4,5]. Improved, simpler distributed

[☆] This paper is an expanded version of a paper which appeared in the Proceedings of ICNP’98.

* Corresponding author. Tel.: +1-614-688-4482; fax: +1-614-292-2911. *E-mail address:* vandalor@cis.ohio-state.edu (B. Vandalore).

[1] Now with Department of Computer Sciences, Purdue University, IN, USA.

[2] Now with Nexabit Networks, MA, USA.

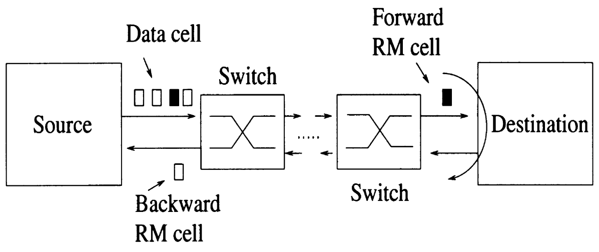


Fig. 1. ABR flow control. RM cells are sent periodically by the source. The RM cell is turned around at the destination. The RM cells in the forward direction are called FRM cells and those in the backward direction are called BRM cells. The switches along the RM cell path indicate the rate which they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

algorithms which achieved max–min fairness were proposed in Refs. [6–11]. Recently, a discussion on generalized definition of max–min fairness and its distributed implementation is given in Refs. [12,13]. A weight-based max–min fairness policy and its implementation in ABR service is given in Ref. [14]. The fairness in the presence of MCR guarantees is discussed in Refs. [15,16].

In this paper, we generalize the definition of the fairness, by allocating the excess bandwidth proportional to weights associated with each source. We show how a switch scheme can support non-zero MCRs and achieve the GW fairness. As an example, we show how the ERICA+ [6] switch scheme can be modified to support the GW fairness.

The modified scheme is tested using simulations with various network configurations. The simulations test the performance of the modified algorithm, with different weights, using a simple configuration, a transient source configuration, a link bottleneck configuration, and a source bottlenecked configuration. Scalability and robustness are tested using a configuration with hundred TCP sources and a background VBR connection carrying long range dependent traffic. These simulations show that the scheme realizes various fairness definitions in ATM TM 4.0 specification that are special cases of the generalized fairness.

Section 2 discusses the GW fairness definition and shows how the various other definitions of fairness can be realized using this general definition. Then, we show how a switch scheme can achieve general fairness. As an example, we show how ERICA+ is modified to support the GW fairness. An analytical proof of convergence for the modified algorithm is given in Appendix A. Simulation configurations and the results for the modified algorithm are given next. Finally, we give our conclusions and discuss some future work.

2. General weighted fairness: definition

We first define the following parameters:

A_l	total available bandwidth for all ABR connections on a given link l
A_b	sum of bandwidth of under-loaded connections that 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
μ_i	MCR of connection i
μ	$\sum_{i=1}^{n} \mu_i$, sum of MCRs of active connections bottlenecked at this link
w_i	preassigned weight associated with the connection i
g_i	GW fair allocation for connection i

The general weighted fair allocation is defined as follows:

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

Note that this definition of fairness is different from the weighted allocation given as an example fairness criterion in ATM TM 4.0 specifications. In the above definition, only the excess bandwidth is allocated proportional to weights. The above definition ensures the allocation is at least MCR.

2.1. Mapping TM 4.0 fairness to general weighted fairness

Here we show how the different fairness criteria mentioned in ATM TM 4.0 specification, can be realized using the above fairness definition.

1. *Max–Min:* in this case MCRs are zero and the bandwidth is shared equally.

   $$g_i = A/n$$

   This is a special case of general weighted fairness with $\mu_i = 0$, and $w_i = c$, where c is a constant.
2. *MCR plus equal share:* the excess bandwidth is shared equally.

   $$g_i = \mu_i + (A - \mu)/n$$

   By assigning equal weights, we achieve the above fairness.
3. *Proportional to MCR:* The allocation is proportional to its MCR.

   $$g_i = \frac{A \times \mu_i}{\mu} = \frac{(\mu + A - \mu)\mu_i}{\mu} = \mu_i + \frac{(A - \mu)\mu_i}{\mu}$$

   By assigning $w_i = \mu_i$, we can achieve the above fairness.

3. Relationship to pricing/charging policies

In real world users expect a service related to the price they are paying for the service. In this section we discuss a simple pricing policy and arrive at a weight function to support such a policy.

Consider a very small interval T of time. The charge C that a customer pays for using a network during this interval is a function of the number of bits W that the network transported successfully:

$$C = f(W, R)$$

where $R = W/T$ is the average rate.

It is reasonable to assume that $f(\ )$ is a non-decreasing function of W. That is, those sending more bits do not pay less. The function $f(\ )$ should also be a non-increasing function of time T or equivalently a non-decreasing function of rate R.

For economy of scale, it is important that the cost per bit does not increase as the number of bits goes up. That is, C/W is a non-decreasing function of W.

Mathematically, we have three requirements:

$$\partial C/\partial W \geq 0$$

$$\partial C/\partial R \geq 0$$

$$\partial (C/W)/\partial W \leq 0$$

One simple function that satisfies all these requirements is

$$C = c + wW + rR$$

Here, c is the fixed cost per connection; w the cost per bit; and r is the cost per Mbps. In general, c, w and r can take any non-negative value.

In the presence of MCR, the above discussion can be generalized to

$$C = f(W, R, M)$$

where M is the MCR. All arguments given above for R apply to M also except that the customers requesting larger M possibly pay more. One possible function is

$$C = c + wW + rR + mM$$

where m is dollars per Mbps of MCR. In effect, the customer pays $r + m$ dollars per Mbps up to M and then pays only r dollars per Mbps for all the extra bandwidth he/she gets over and above M.

Consider two users with MCRs M_1 and M_2. Suppose their allocated rates are R_1 and R_2 and, thus, they transmit W_1 and W_2 bits, respectively. Their costs are

$$C_1 = c + wW_1 + rR_1 + mM_1$$

$$C_2 = c + wW_2 + rR_2 + mM_2$$

Cost per bit (C/W) should be a decreasing function of bits W. Thus, if $W_1 \geq W_2$:

$$C_1/W_1 \leq C_2/W_2$$

$$c/W_1 + w + rR_1/W_1 + mM_1/W_1$$
$$\leq c/W_2 + w + rR_2/W_2 + mM_2/W_2$$

Since $R_i = W_i/T$, we have:

$$c/(R_1T) + w + r/T + mM_1/(R_1T)$$
$$\leq c/(R_2T) + w + r/T + mM_2/(R_2T)$$

$$c/R_1 + mM_1/R_1 \leq c/R_2 + mM_2/R_2$$

$$(c + mM_1)/(c + mM_2) \leq R_1/R_2$$

$$(a + M_1)/(a + M_2) \leq R_1/R_2$$

where a ($= c/m$) is the ratio of the fixed cost and cost per unit of MCR.

Note that the allocated rates should either be proportional to $a +$ MCR or be a non-decreasing function of MCR. We have chosen to use $a +$ MCR as the weight function in our simulations.

4. General weighted fair allocation problem

In this section we give the formal specification of the general weighted fair allocation problem, and give a motivation for the need of a distributed algorithm.

The following additional notation is necessary:

$\mathscr{L}$	set of links, $\mathscr{L}_s$ set of links that session s goes through
$\mathscr{S}$	set of sessions, $\mathscr{S}_l$ set of sessions that go through link l. $N = \|S\|$
$\mathscr{A}$	$(\mathscr{A}_l, l \in \mathscr{L})$ set of available capacity
$\mathscr{M}$	$(\mu_s, s \in \mathscr{S})$, where μ_s is the minimum cell rate (MCR) for session s
$\mathscr{W}$	$(w_1, w_2, \ldots, w_N)$ denotes the weight vector
$\mathscr{R}$	$(r_1, r_2, \ldots, r_N)$ the current allocation vector (or rate vector)
$\mathscr{G}$	$(g_1, g_2, \ldots, g_N)$ the general fair allocation; $\mathscr{G}_{\mathscr{S}_l}$ denotes the set of allocations of sessions going over link l

Definition 1 (General weighted fair allocation problem). The GW fair problem is to find the rate vector equal to the GW fair allocation, i.e. $\mathscr{R} = \mathscr{G}$., where $g_i \in \mathscr{G}_{\mathscr{S}_l}$ is calculated for each link l as defined in Section 2.

Note the 5-tuple $(\mathscr{S}, \mathscr{L}, \mathscr{C}, \mathscr{W}, \mathscr{R})$ represents an instant of the bandwidth sharing problem. When all weights are equal the allocation is equivalent to the general max–min fair allocation as defined in Refs. [12,13]. A simple centralized algorithm for solving the above problem would be to first find the correct allocation vector for the bottleneck links. Then, solve the same problem of smaller size after deleting bottleneck links. A similar kind of centralized, recursive algorithm is discussed in Ref. [13]. Centralized algorithm implies that all information is known at each switch, which is not feasible, hence a distributed algorithm is necessary.

5. Achieving general fairness

A typical ABR switch scheme calculates the excess bandwidth capacity available for best effort ABR after reserving bandwidth for providing MCR guarantee and higher priority classes such as VBR and CBR. The switch fairly divides the excess bandwidth among the connections bottlenecked at that link. Therefore, the ACR can be represented by the following equation:

$$ACR(i) = \mu_i + ExcessFairshare(i)$$

ExcessFairshare is the amount of bandwidth allocated over the MCR in a fair manner.

In the case of GW fairness, the *ExcessFairshare* term is given by

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

If the network is near steady state (input rate = available capacity), then the above allocation enables the sources to attain the GW fairness. The ATM TM 4.0 specification mentions that the value of $(ACR - MCR)$ can be used in the switch algorithms. We use this term to achieve the GW fairness. We have to ensure the $(ACR - MCR)$ term converges to *ExcessFairshare* value. We use the notion of *activity level* to achieve the above objective [17]. A connection's *excess activity level* $(EAL(i))$ is defined as follows:

$$EAL(i) = minimum\left(1, \frac{max(0, SourceRate(i) - \mu_i)}{ExcessFairshare(i)}\right)$$

$SourceRate(i)$ is the rate at which the source is currently transmitting data. Note that $SourceRate(i)$ is the $ACR(i)$ given as the feedback rate earlier by the switch. The excess activity level indicates how much of the *ExcessFairshare* is actually being used by the connection. Excess activity level is zero if $SourceRate(i)$ is less than μ_i. The activity level attains the value of 1 when the *ExcessFairshare* is used by the connection. It is interesting to note that using activity level for calculating is similar to the *consistent marking* technique of Charny [18], where the switch marks connections which have lower rate than their *advertised rate*. The new advertised rate is calculated using the equation

$$\text{Advertised Rate} = \frac{\mathscr{A}_l - \sum \text{Rates of marked connections}}{|S_l| - \sum \text{Marked connections}}$$

The activity level inherently captures the notion of marking, i.e. when a source is bottlenecked elsewhere, then activity level times the fairshare (based on available left over capacity) is the actual fairshare of the bottleneck source. The computation of activity level can be done locally and is an O(1) operation, compared to O(n) computations required in consistent marking [18].

We expect that the links use their *ExcessFairshare*, but this might not be the case. By multiplying the weights by the activity level, and using these as the weights in calculating the *ExcessFairshare* we can make sure that the rates converge to the GW fairness allocation. Therefore, the *ExcessFairshare* share term is defined as

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

Note that w_i is not multiplied by $EAL(i)$ in the numerator, since we desire to attain a value of $EAL(i) = 1$ for all sources and give excess bandwidth in proportion to the weights. Due to this, sources which have not yet achieved their fairshare are asked to increase their rate to *ExcessFairShare*. Rate of sources which are bottlenecked elsewhere are not affected. The rate of such a source depends only on the explicit feedback rate, which it receives from switches at which it is bottlenecked. Connections which are bottlenecked at sources also receive the correct amount of *ExcessFairShare*.

There is a possibility that the denominator becomes zero in the above expression, when all the sources are inactive $(SourceRate(i) < \mu_i)$. In this case the *ExcessFairshare* evaluates to a infinite value. This means that since there is no load on the link, each source can have the whole available capacity as the fairshare. So in our simulations we handle this special case by taking the minimum of available bandwidth $(A - \mu)$ and value calculated by the above expression.

A switch algorithm can use the above *ExcessFairshare* term to achieve the GW fairness. In Section 6, we show how the ERICA+ switching algorithm is modified to achieve the GW fairness.

6. Example modifications to a switch algorithm

The ERICA+ algorithm operates at each output port of a switch. The switch periodically monitors the load on each link and determines a load factor (z), the available ABR capacity, and number of currently active sources or VCs. The measurement period is the “Averaging Interval”. These measurements are used to calculate the feedback rate which is indicated in the backward RM (BRM) cells. The measurements are done in the forward direction and the feedback is given in the backward direction. The complete description of the ERICA+ algorithm can be obtained from Ref. [6].

The ERICA+ algorithm uses the term *FairShare*, which is the bottleneck link capacity divided by the active number of VCs. It also uses a *MaxAllocPrevious* term, which is the maximum allocation in the previous “Averaging Interval”. This term is used to achieve max–min fairness. We modify the algorithm by replacing the *FairShare*

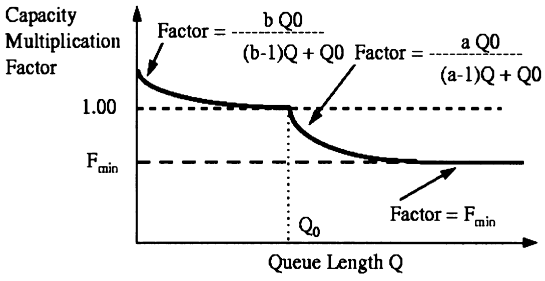


Fig. 2. The dynamic queue control function used in ERICA + . $F_{\min}$ thresholds the amount of capacity used for queue draining. Q_0 is the target queue length, its value is dependent on the "Target delay" parameter and the link capacity. A value of $a = 1.15$ and $b = 1$ is used in our simulations.

term by *ExcessFairshare*(i) and adding the MCR (μ_i). The key steps in ERICA+ which are modified to achieve the GW fairness are shown as follows:

Algorithm A.

```
At the end of Averaging Interval
    Total ABR Cap ← Link Cap − VBR Cap − $\sum_{i=0}^{n}$ min (SourceRate(i),$\mu_i$)
    Target ABR Cap ← Factor × Total ABR Cap
    Input Rate ← ABR Input Rate − $\sum_{i=0}^{n}$ min (SourceRate(i),$\mu_i$)
    z ← Input Rate/Target ABR Cap
foreach VC_i
    EAL(i) ← min(1,max(0,SourceRate(i) − $\mu_i$)/ExcessFairshare(i))
    SumOfWts ← SumOfWts + $w_i$EAL(i)
endfor
foreach VC_i
    ExcessFairshare(i) ← $w_i$(Target ABR Cap)/{SumOfWts}
endfor
```

The *Factor* term is dependent on the queue length [19].When the *Factor* is less than 1, $(1 - Factor) \times$ Total ABR Cap is used to drain the queues. A simple choice is to use a constant queue control function (CQF), where the *Factor* is set to a value less than 1, say 0.95. The remaining 5% of the link capacity is used for queue draining. Another option is to use a dynamic queue control function (DQF). In DQF, the *Factor* value is 1 for small queue lengths and drops sharply as queue length increases. ERICA+ uses a hyperbolic function for calculating value of the *Factor* (Fig. 2).

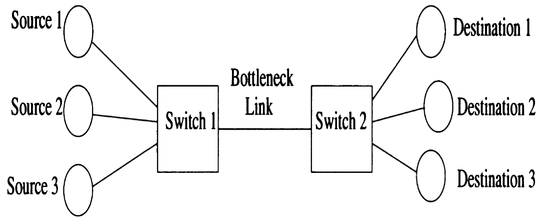


Fig. 3. N sources–N destinations configuration.

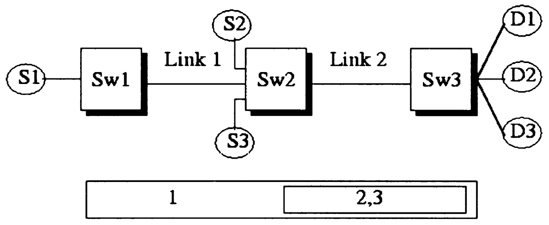


Fig. 4. Three sources–bottleneck configuration. S1 is bottlenecked at 10 Mbps.

When a BRM is received

$$VCShare \longleftarrow \frac{\max(0, SourceRate(i) - \mu_i)}{z}$$

$$ER \longleftarrow \mu_i + \max(ExcessFairshare(i), VCShare)$$

$$ER_{RM\_Cell} \longleftarrow \min(ER_{RM\_Cell}, ER, \text{Target ABR Cap})$$

The *VCShare* is used to achieve a unit overload. When the network reaches steady state the *VCShare* term converges to *ExcessFairshare*(i), achieving the generalized fairness criterion. The complexity of the computations done at the switching interval is O(number of VCs). The update operation when the BRM cell arrives is an O(1) operation. Proof of convergence of Algorithm A is given in Appendix A.

7. Simulation configurations

We use different configurations to test the performance of the modified algorithm. We assume, unless specified otherwise, that the sources are greedy, i.e. they have infinite amount of data to send, and always send data at ACR. In all configurations, the data traffic is unidirectional, from source to destination. If bidirectional traffic is used, similar results will be achieved, except that the convergence time will be longer since the RM cells in the backward direction will travel along with the data traffic from destination to source. All the link bandwidths are 149.76 (155.52 less than the SONET overhead), expect in the GFC-2 configuration.

7.1. Three sources

This is a simple configuration in which three sources send data to three destinations over two switches and a bottleneck link (Fig. 3). This configuration is used to demonstrate that the modified switch algorithm can achieve the general fairness for different set of weight functions.

7.2. Source bottleneck

In this configuration (Fig. 4), the source S1, is bottlenecked at 10 Mbps, which is below its fairshare

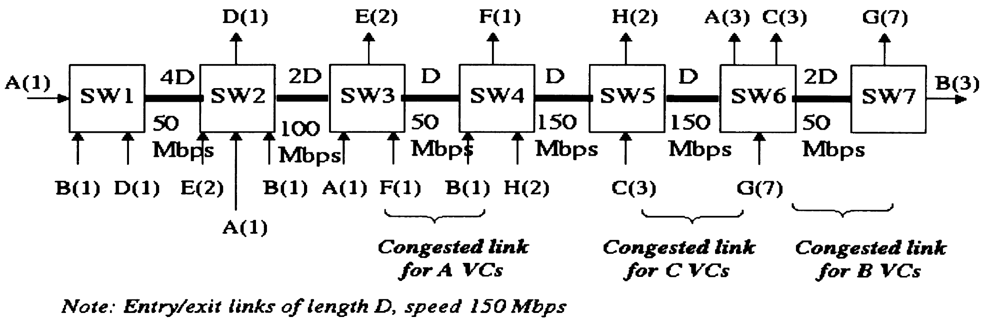


Fig. 5. Generic fairness configuration-2. X(*n*) indicates that there *n* number of VCs of type X.

(50 Mbps). This configuration tests whether the GW fairness can be achieved in the presence of source bottleneck.

7.3. Generic fairness configuration-2 (GFC-2)

This configuration (explained in detail in Ref. [20]) is a combination of upstream and parking lot configuration (Fig. 5). In this configuration, all the links are bottlenecked links, round trip times are different for different type of VCs.

7.4. TCP sources with VBR background

This configuration is used to test the robustness and scalability of the algorithm (Fig. 6). In this configuration 100 infinite TCP sources (large file transfers) transmit data continuously through a bottleneck link to 100 destinations. One VBR connection carrying multiplexed MPEG traffic, which is long-range dependent, is used as background traffic [21]. The mean bandwidth of VBR traffic is 45 Mbps. The VBR traffic is generated with hurst parameter (*H*) value of 0.9, hence it has high degree of self-similarity.

7.5. Simulation parameters

The simulations were done on an extensively modified version of NIST simulator [22]. The following parameter values were used in all our simulations: link distance 1000 km; averaging interval 5 ms; target delay 1.5 ms; exponential decay factor 0.1 (when using dynamic queue control function).

The "Averaging Interval" is the period for which the

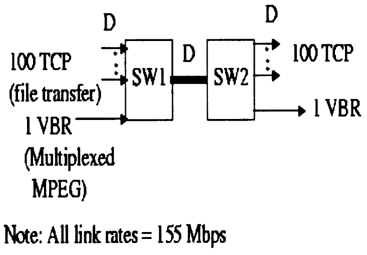


Fig. 6. 100 TCP sources + VBR background. TCP sources are infinite sources. VBR connection carries multiplexed MPEG traffic which exhibits long range dependency.

switch monitors various parameters. Feedback is given based on these monitored values. The ERICA+ algorithm uses dynamic queue control to vary the available ABR capacity dependent on queue size. At steady state the queue length of constant value can be obtained. The "Target Delay" parameter specifies the desired delay due to this constant queue length at steady state. When using dynamic queue control function we exponentially average *Excess-Fairshare* term. This is done so that effectively only one feedback is given in each feedback interval and to absorb the variation in "Target ABR Cap" value due to the queue control function. For convergence, the feedback delay, averaging interval and exponential averaging decay factor should obey the following equation:

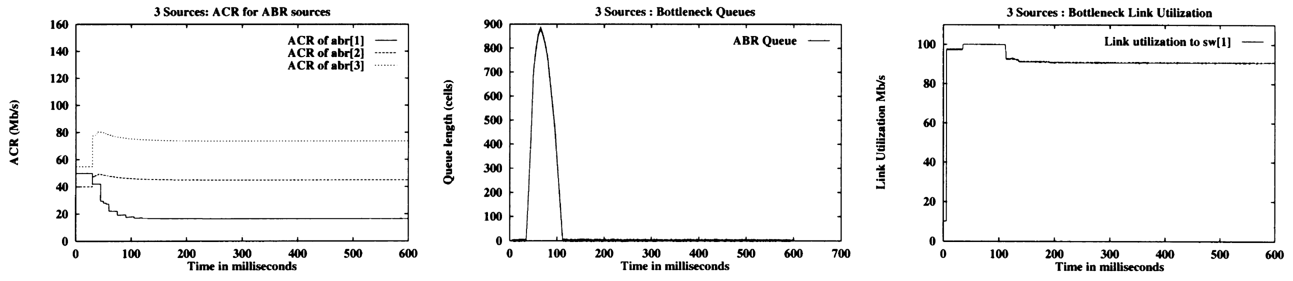
$$\frac{\text{Averaging interval}}{\text{Exponential decay factor}} \geq \text{Feedback delay}$$

8. Simulation results

In this section, we give the simulation results for the different configurations. The simulation results using both constant queue control function (shown in graphs as configuration name and CQF) and dynamic queue control function (shown in graphs as configuration and DQF) are given. For the CQF the value of *Factor* used is 0.9. The tabular results are those obtained from simulations using the dynamic queue control function.

Table 1
Three sources configuration simulation results

Case	Src	MCR	*a*	Wt func	Expected fair share	Actual share
1	1	0	∞	1	49.92	49.92
	2	0	∞	1	49.92	49.92
	3	0	∞	1	49.92	49.92
2	1	10	∞	1	29.92	29.92
	2	30	∞	1	49.92	49.92
	3	50	∞	1	69.92	69.92
3	1	10	5	15	18.54	18.53
	2	30	5	35	49.92	49.92
	3	50	5	55	81.31	81.30



(a) ACRs (b) Queue length (c) Utilization

Fig. 7. Three sources: case 3 + CQF simulation results: (a) ACRs; (b) queue length; (c) utilization.

8.1. Three sources

Simulations using a number of weight functions were done using the simple three sources configuration to demonstrate that GW fairness is achieved in all these cases. The ICRs (Initial Cell Rates) of the sources were set to (50,40,55) Mbps in all the simulations.

The allocations of these cases using DQF are given in Table 1. The following can be observed from Table 1.

- Case 1: $a = \infty$, MCRs = 0. All weights are equal so the allocation (149.76/3) = 49.92 Mbps for each connection. This allocation is the same as max–min fair allocation.
- Case 2: $a = \infty$, MCRs $\neq 0$. The left over capacity 149.76 − (10 + 30 + 50) = 59.76 Mbps is divided equally among the three sources. So the allocation is (10 + 19.92, 30 + 19.92, 50 + 19.92) = (29.92, 49.92, 69.92) Mbps.
- Case 3: $a = 5$, MCRs $\neq 1$. Hence, the weight function is 5 + MCR. The leftover capacity, 59.76 Mbps, is divided proportional to (15,35,55). Hence the allocation is $(10 + 15/105 \times 59.76, 30 + 35/105 \times 59.76, 50 + 55/105 \times 59.76)$ = (18.54,49.92,81.31) Mbps.

Fig. 7 shows the ACRs, queue and utilization graphs of the three sources for case 3 using constant queue control function. Fig. 8 shows the corresponding graphs using dynamic queue control function. From the figures, one can observe that the sources achieve the GW fairness rate and queues are controlled in steady state. When using DQF, queue length values oscillate before reaching steady state values. The utilization achieved at steady state is 100% when using DQF and 90% (same as *Factor* value) when using CQF.

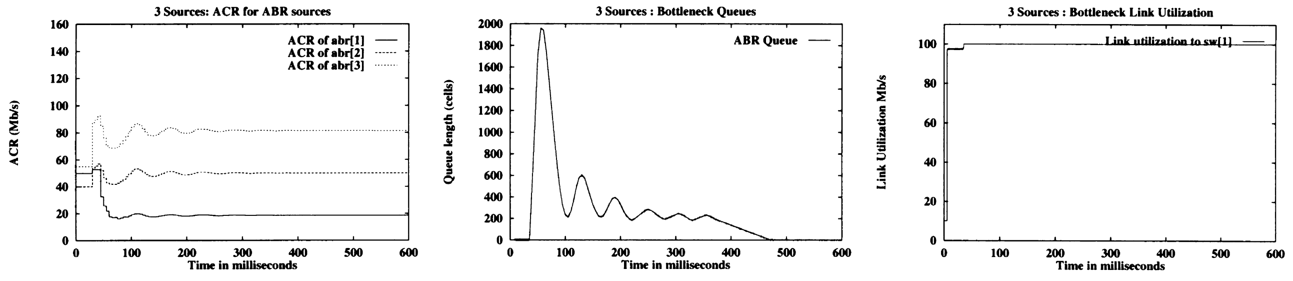
8.2. Three sources: transient

In these simulations, the same simple three source configuration is used. Source-1 and source-3 transmit data throughout the simulation period. Source-2 is a transient source, which starts transmitting at 400 ms and stops at 800 ms. The total simulation time is 1200 ms. Same parameter values from the cases 1–3 of Section 8.1 were used in these simulations. The results of these simulations are given in Table 2. The non-transient (ntr) columns give the allocation when transient source-2 is not present, i.e. between 0 and 400 ms and between 800 and 1200 ms. The transient (tr) column give allocation when the transient source-2 is present, i.e. between 400 and 800 ms.

The ACR values of the sources and the utilization of the bottleneck link for case 2 are shown in Fig. 9. It can be seen both from Table 2 and the graphs that the switch algorithm does converge to the general fairness allocation even in the presence of transient sources. The algorithm has a good response time, since there is only a small dip in the utilization graph when the transient source stops sending traffic (at 800 ms).

8.3. Source bottleneck

Cases 1–3 of Section 8.1 were simulated using the three sources bottleneck configuration. The total simulation time



(a) ACRs (b) Queue length (c) Utilization

Fig. 8. Three sources: case 3 + DQF simulation results: (a) ACRs; (b) queue length; (c) utilization.

Table 2
Three sources transient configuration simulation results (ntr, non-transient period; tr, transient; NC, not converged)

No.	Src	Wt func	Exp frshr (ntr)	Actual (ntr) share	Exp frshr (tr)	Actual share
1	1	1	74.88	74.83	49.92	49.92
	2	1	NC	NC	49.92	49.92
	3	1	74.88	74.83	49.92	49.92
2	1	1	54.88	54.88	29.92	29.83
	2	1	NC	NC	49.92	49.92
	3	1	94.88	95.81	69.92	70.93
3	1	15	29.92	29.23	18.53	18.53
	2	35	NC	NC	49.92	49.92
	3	55	119.84	120.71	81.30	81.94

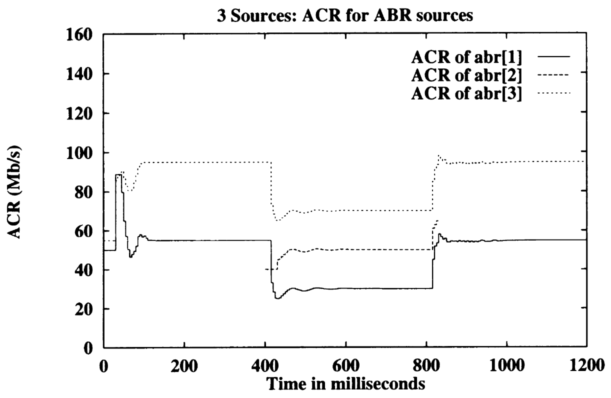
was 800 ms. In these simulations the source S1 is bottlenecked at 10 Mbps for first 400 ms, i.e. it always transmits data at the rate of at most 10 Mbps, irrespective of its ACR (and ICR). After 400 ms, source S1 behaves like an infinite source and sends data at ACR.

The initial ICRs were set to 50, 30, 110 Mbps. The load on the bottleneck link is near unity. If the switch algorithm uses the CCR (Current Cell Rate) value indicated in the RM cell as the source rate, the switch cannot estimate the correct value of source rate of the bottleneck source. However, if the switch uses measured source rate then it can correctly estimate the bottlenecked source's rate. Table 3 shows the results both when the switch uses the CCR field and when it measures the source rate during the presence of source bottleneck (i.e. before 400 ms). The correct fairness is achieved only when the measured source rates are used. When the source bottleneck disappears after 400 ms, the sources achieve the GW fairness (fairshare value same as in the simple configuration), both when CCR is used as source rate and when source rates are measured.

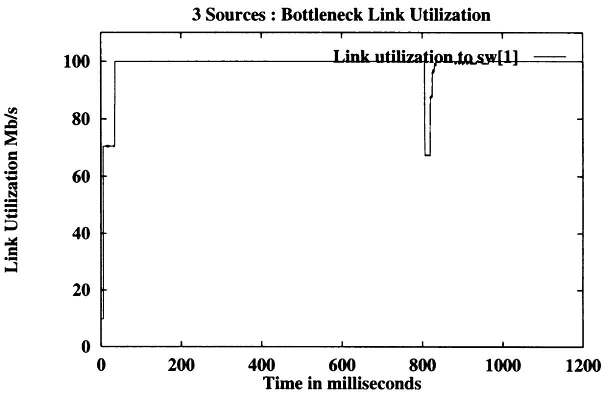
Fig. 10(a) shows the ACR graph for the simulation of case 1 using source rate from the CCR field of RM cell. Fig. 10(b) shows the same case using measures source rates. When the CCR value from the RM cells is used as source rate, the algorithm is not able to estimate the actual rate at which the source is sending data. So, it does not estimate the correct GW fairshare values in the presence of source bottlenecks. When measured source rate is used it calculates correct fairshare even in the presence of source bottlenecks.

8.4. Link bottleneck: GFC-2

In this configuration, each link is a bottleneck link. An MCR value of 5 was used for all A type VCs. All other VCs have MCR of 0. The MCR plus equal share of excess bandwidth was chosen as the fairness criteria. Dynamic queue control function was used in this simulation. The expected share for VCs of type A, B, C, D, E, F, G, H are 11.25, 5, 33.75, 33.75, 33.75, 6.25, 5 and 50.625 Mbps, respectively. The actual allocation for these VCs in the simulation was 11.25, 5, 35.67, 35.75, 35.75, 6.25, 5 and 50.5 Mbps, respectively, which agree well with the expected allocations. Fig. 11(a) shows the ACR graphs for each type of VCs. Fig. 11(b) shows the queue length graph at various bottleneck links between the switches. From the figure and actual allocations, it can be seen that the VCs converge to their expected fairshare. The queue length graphs show that initial queue buildup occurs before convergence and its maximum queue length depends on ICR and round trip time. This simulation demonstrates that the algorithm works in the presence of multiple link bottlenecks and different round trip times.



(a) ACRs



(b) Utilization

Fig. 9. Three sources (transient): (a) ACR and (b) utilization graphs.

Table 3
Three sources bottleneck configuration simulation

Case	Src	Wt	Exp fishr func	Using CCR in RM cell	Using measured CCR
1	1	1	69.92	51.50	69.29
	2	1	69.88	51.80	69.29
	3	1	69.88	85.94	69.29
2	1	1	39.88	43.98	39.58
	2	1	59.88	52.06	59.57
	3	1	79.88	85.85	79.76
3	1	15	19.96	42.72	19.19
	2	35	53.32	51.62	53.28
	3	35	86.64	86.16	86.37

8.5. 100 TCP sources with VBR background

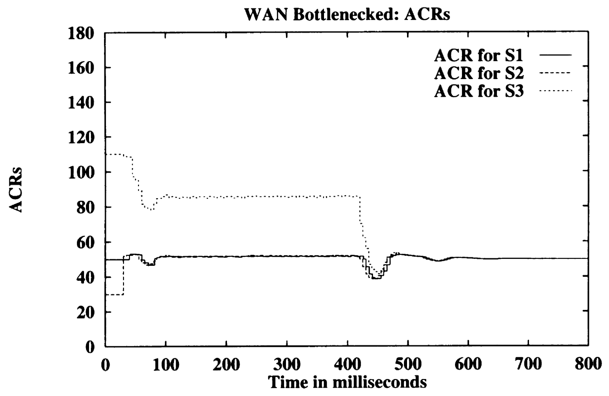
The VBR VC carrying multiplexed MPEG source traffic has higher priority over TCP sources running over ABR. The VBR traffic generated is highly variable as shown in Fig. 12(a). The TCP sources are infinite TCP sources. During initial period, the TCP traffic is bursty since its congestion window is limited by ACR and slow start protocol. Once the congestion window reaches the maximum value the TCP sources become equivalent to persistent source. All TCP sources start sending data at the same time, so the load phases (active and idle periods) of multiple sources coincide. Source-25 has MCR value of 1 Mbps, source-50 has MCR of 1.5 Mbps and source-100 has MCR value of 2 Mbps. All other TCP sources have an MCR value of 0.5 Mbps. A value of 10 was used for parameter a of the weight function (a + MCR). Hence the GW fairness criteria here is MCR plus proportional to MCR.

Fig. 12(b)–(d) shows ACRs, queue length and link utilization, respectively, which are ATM level metrics. Fig. 12(e) and (f) shows congestion window and average throughput, respectively, which are TCP level metrics. Though the system does not have a steady state the queues are controlled and utilization is high. The expected throughput received by the TCP sources when congestion window is maximum is 1.02 Mbps for source-1, 1.54 Mbps for source-25, 2.07 Mbps for source-50 and 2.59 Mbps for source-100 according to the GW fairness criteria (MCR plus proportional MCR in this case). The average throughput values as shown in Fig. 12(f) is slightly different from the expected throughputs. This is due to the varying VBR capacity and since the average throughputs include measurement during initial burstiness of TCP sources, where the congestion windows have not yet reached the maximum value. This simulation demonstrates that the algorithm is robust and scalable.

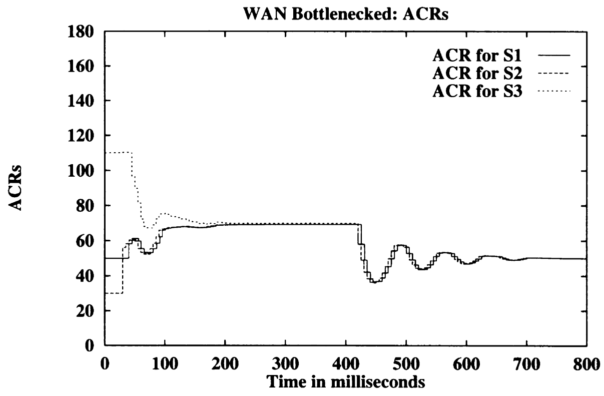
9. Conclusion

In this paper, we have given a general definition of fairness, which inherently provides MCR guarantee and divides the excess bandwidth proportional to predetermined weights. Different fairness criterion such as max–min fairness, MCR plus equal share, proportional MCR can be realized as special cases of this general fairness. We showed how to realize a typical pricing policy by using appropriate weight function. The GW fairness can be achieved by using the *ExcessFairshare* term in switch algorithms. The weights are multiplied by the activity level when calculating the *ExcessFairshare* to reflect the actual usage of the source.

We have shown how ERICA+ switch algorithm can be modified to achieve this general fairness. The proof of convergence of Algorithm A is given in Appendix A. The



(a) Case 3 + DQF + source rate from CCR field



(b) Case 3 + DQF + measured source rate

Fig. 10. Three sources bottleneck: ACR graphs: (a) case 3 + DQF + source rate from CCR field; (b) case 3 + CCR + measured source rate.

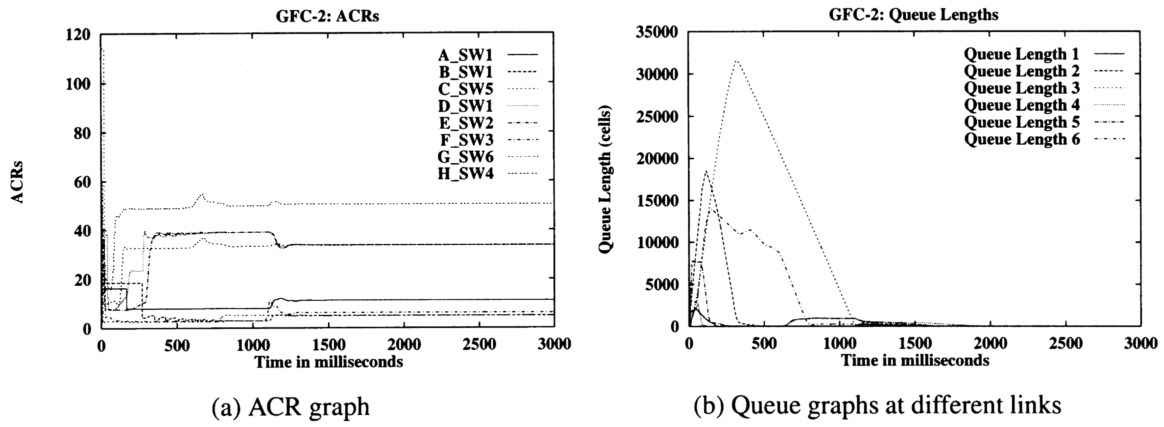


Fig. 11. GFC-2 configuration: (a) ACR graph; (b) queue graphs at different links.

simulation results show that the modified algorithm achieves the general fairness in all configurations. In addition, the results show that the algorithm converges in the presence of both source and link bottleneck and is quick to respond in the presence of transient sources. In source bottlenecked configuration the value of the CCR (source rate) from the RM cells may be incorrect. Hence, it is necessary to use the measured source rate in the presence of source bottlenecks. The algorithm is robust and scalable as demonstrated by simulation results using the hundred TCP sources plus VBR background configuration. Future work includes, extending the GW fairness criterion to multipoint ABR connections and designing a robust and scalable switch algorithm for such connections.

Acknowledgements

This research was sponsored in part by Rome Laboratory/C3NC Contract F30602-96-C-0156.

Appendix A. Proof of convergence of Algorithm A

We make the following assumptions:

- Synchronous update of source rates.
- Queue control function is a constant function.
- Infinite (greedy) sources, which always have data to send. Though there might be source or link bottleneck present.
- If a source bottleneck is present, it does not change its bottleneck rate during convergence.
- $\sum_{s \in S_l} \mu_i \leq A_l$, i.e. sum of MCRs is less than available ABR capacity (connection admission policy).
- Load factor $z > 0$ and $ER < A_l < LinkRate$.

Lemma A1. *Algorithm A converges to the GW fair allocation, for a session bottlenecked by a link.*

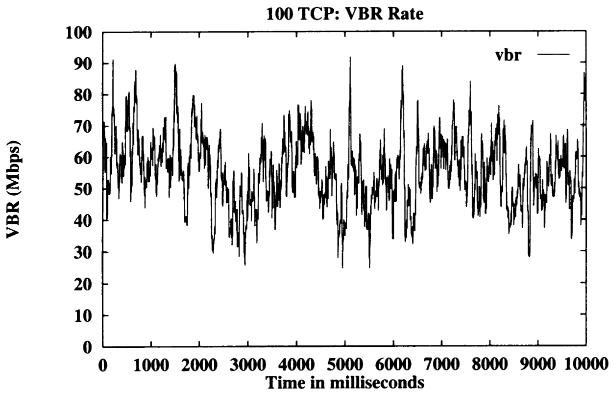
Proof. The proof technique used here is similar to the one used in Ref. [6]. Let l_b be the link which is bottlenecked. Without loss of generality assume that first k sessions through the link l_b are bottlenecked (either link bottlenecked or source bottlenecked) elsewhere. Let $n = |S_{l_b}| - k$. Let $r_{b1}, r_{b2}, \ldots, r_{bk}$ be the bottleneck rates and $r_1, r_2, \ldots, r_n$ be the rates of non-bottlenecked (under-loaded) sources. Let $A_b = \sum_{i=1}^{k} r_{bi}$ be total capacity of bottlenecked links. These non-bottlenecked sources are bottlenecked at the current link l_b. According to the GW fairness definition, fair allocation rates g_i is given by

$$g_i = \mu_i + \frac{w_i(A_l - A_b)}{\sum_{J=1}^{n} w_j}$$

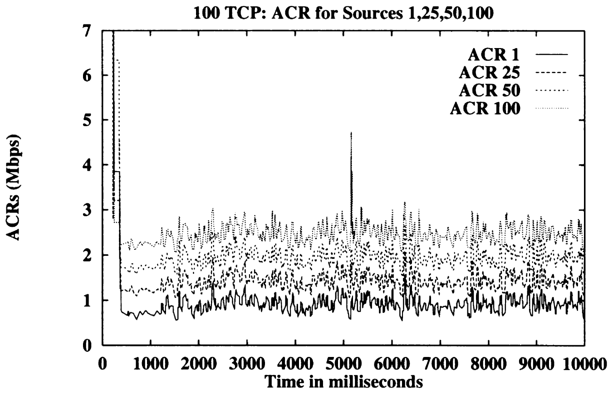
Assume that the bottlenecks elsewhere have been achieved, therefore the rates $r_{b1}, r_{b2}, \ldots, r_{bk}$ are stable. For simplicity, assume that the MCRs of these sources are zero. Proof for the bottlenecks having non-zero MCRs is a simple extension.We show that rates allocated at this switch converges to $r_{b1}, r_{b2}, \ldots, r_{bk}$ and $g_1, g_2, \ldots, g_n$ and load factor converges to $z = 1$.

Case 1: load factor $z < 1$. Here the link is under-loaded, hence due to the *VCShare* term $(SourceRate(i) - \mu_i)/z$, all the rates increase. If $n = 0$, i.e. all the sessions across this link are bottlenecked elsewhere, there are no non-bottlenecked sources, the GW fair allocation is trivially achieved. Assume that $n \geq 1$, now because of the *VCShare* term (in step for calculating *ER* in Algorithm A), the rates of non-bottlenecked sources increase. This continues until the load factor reaches a value greater than or equal to 1. Hence we have shown that if load factor is less than 1, the rates increase till the load factor becomes greater than 1.

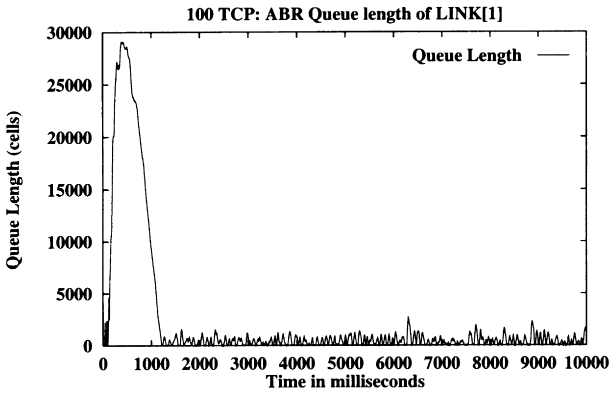
Case 2: load factor $z > 1$. In this case if the link is not getting its *ExcessFairshare* then, its rate increases, which might further increase z. This continues till all the sessions achieve at least their *ExcessFairshare*. At this point the allocation rates are decreased proportional to $1/z$ due to the first term. As in the previous case z decreases, until it reaches a value of 1 or less.



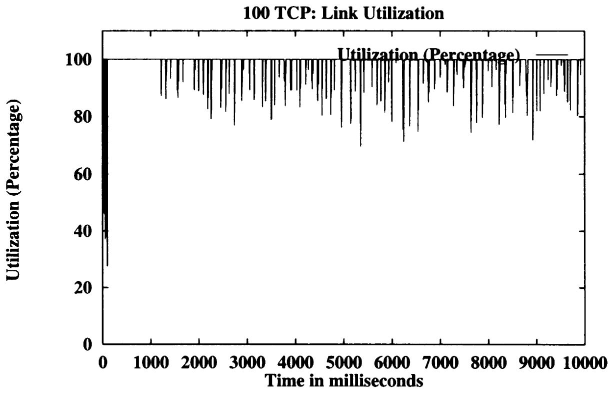
(a) VBR capacity



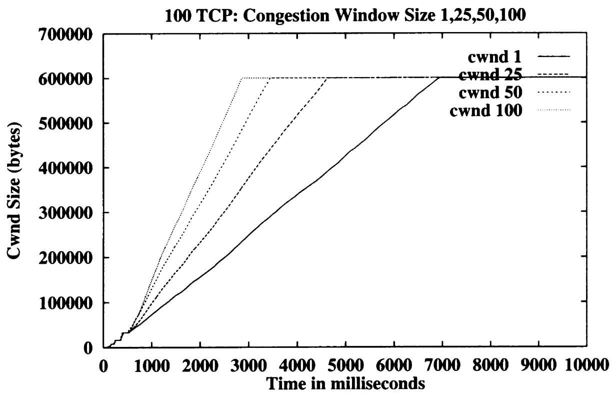
(b) ACRs



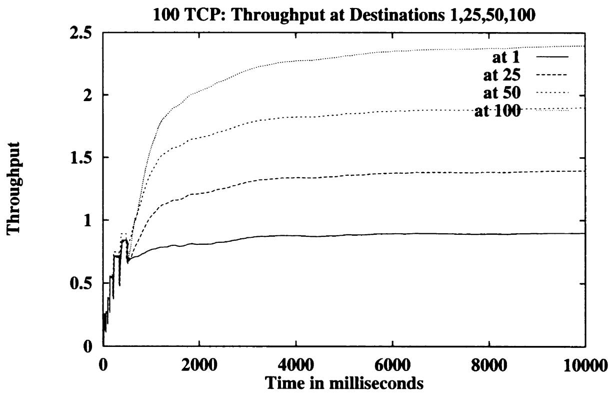
(c) Queue length



(d) Link utilization



(e) TCP congestion window



(f) Average TCP throughput

Fig. 12. 100 TCP + VBR background simulation graphs: (a) VBR capacity; (b) ACRs; (c) queue length; (d) link utilization; (e) TCP congestion window; (f) average TCP throughput.

From the above two cases it can be seen that load factor oscillates around 1 and converges to the value of 1. Assume that load factor is $z = 1 + \delta$, then the number round trip times for it to converge to 1 is given by $\log_{1+\delta}|S_l|$. Henceforth, in our analysis we assume that the network is near the steady state that is load factor is near 1. This implies that

$$\sum_{i=1}^{k} r_{bi} + \sum_{i=1}^{n} r_i = A_l \rightarrow \sum_{i=1}^{n} r_i = A_l - A_b$$

Let $A_m = \sum_{i=1}^{n} \mu_i$ be the total allocation for MCRs of the non-bottlenecked sources. Define $\alpha_i = r_i - \mu_i$, then we have

$$\sum_{i=1}^{n} \alpha_i = A_l - A_b - A_m = A$$

We have to show that

$$\alpha_i = \frac{w_i A}{\sum_{j=1}^{n} w_j}$$

Case A: $k = 0$, i.e. there are no bottleneck sources. From the step for calculating *ER* in Algorithm A, we have

$$\alpha_i = \max(ExcessFairshare(i), \alpha_i/z)$$

We observe that this equation behaves like a differential equation in multiple variables [23]. The behavior is like that of successive values of root acquired in the Newton–Ralphson method for finding roots of an equation. Hence the above equation converges, and the stable values of α_i is given by

$$\alpha_i = ExcessFairshare(i) = \frac{w_i A}{\sum_{j=1}^{n} w_j EAL(i)}$$

Since we have assumed greedy sources and no bottlenecks in this case, the excess activity level is 1 for all sessions. Hence

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

which is indeed the desired value for α_i.

Case B: $k \neq 0$, i.e. there are some bottleneck sources. Let β_i be the allocated rate corresponding to r_{bi}. Let w_{bi} be the weight for session s_{bi}. Let $W_b = \sum_{i=1}^{k} w_{bi} EAL(b_i)$ and $W = \sum_{i=1}^{n} w_i$. We know that the equation for the rate allocation behaves as a stabilizing differential equation. In the steady state all the above terms such as W, W_b and rates stabilize. For sources bottlenecked elsewhere the algorithm calculates a rate β_i which is greater than r_{bi}, otherwise the bottlenecked session would be bottlenecked at the current link. For non-bottlenecked source the rate at steady state is given by

$$\alpha_i = \frac{w_i(A_l - A_m)}{W_b + W}$$

Since the link has an overload of one at steady state, we have

$$\sum_{i=1}^{n} \alpha_i = A_l - A_m - A_b$$

which implies that

$$\frac{(A_l - A_m)\sum_{i=1}^{n} w_i}{W_b + W} = A_l - A_m - A_b$$

Substituting W for $\sum_{i=1}^{n} w_i$ we get

$$W_b = \frac{WA_b}{A_l - A_m - A_b}$$

Using the above value for W_b we get

$$\alpha_i = \frac{w_i(A_l - A_m)}{WA_b/(A_l - A_m - A_b) + W} = \frac{w_i(A_l - A_m - A_b)}{W}$$

which is the desired value for the α_i. Hence, the sessions bottlenecked at the link l_b do indeed achieve the GW fairness. □

Theorem A1. *Starting at any arbitrary state of the network, if only greedy sources and source bottlenecked or link bottlenecked sources are present Algorithm A converges to GW fair allocation.*

Proof. The convergence of the distributed algorithm is similar to the centralized algorithm. Assume that the centralized algorithm converges in M iterations. At each iteration there are set of links $\mathscr{L}_i$ which are bottlenecked at the current iteration $\cup_{i=1}^{M} \mathscr{L}_i = \mathscr{L}$.

Using Lemma A1, we know that each link $l \in \mathscr{L}_i$ does indeed converge to the general fair allocation $\mathscr{G}_l$. The distributed algorithm converges in the above order of links until the whole network is stable and allocation is $\mathscr{G}$. The number of round trips taken to converge is bounded by $M \times O(\log S)$ since each link takes $O(\log S_l)$ round trips for convergence. □

References

[1] S.S. Sathaye, ATM Forum Traffic Management Specification Version 4.0. ftp://ftp.atmforum.com/pub/approved-specs/af-tm-0056.0000.pdf, 1996.
[2] N. Yin, M.G. Hluchyj, On closed-loop rate control for ATM cell relay networks, Proceedings of the IEEE INFOCOM, 1994, pp. 99–108.
[3] J. Mosley, Asynchronous distributed flow control algorithms, PhD Thesis, Department of Electrical Engineering, MIT, Cambridge, 1984.
[4] A. Charny, An algorithm for rate allocation in a packet-switching network with feedback, Master's Thesis, MIT, Cambridge, 1994.
[5] D.H.K. Tsang, W.K.F. Wong, A new rate-based switch algorithm for ABR traffic to achieve max–min fairness with analytical approximation and delay adjustment, Proceedings of IEEE Globecom'96, 1996.
[6] S. Kalyanaraman, R. Jain, R. Goyal, S. Fahmy, B. Vandalore, The ERICA switch algorithm for ABR traffic management in ATM networks, Transactions on Networking, October 1999 (in press).
[7] C. Fulton, S.-Q. Li, C.S. Lim, UT: ABR feedback control with tracking, Preprint, 1997.
[8] L. Kalampoukas, A. Varma, K.K. Ramakrishnan, An efficient rate allocation algorithm for ATM networks providing max–min fairness, Proceedings of the Sixth IFIP International Conference on High Performance Networking, September, 1995.
[9] K. Siu, T. Tzeng, Intelligent congestion control for ABR service in ATM networks, Computer Communication Review 24 (5) (1995) 81–106.
[10] Y. Afek, Y. Mansour, Z. Ostfeld, Phantom: a simple and effective

flow control scheme, Proceedings of the ACM SIGCOMM, August, 1996.

[11] L. Roberts, Enhanced PRCA (proportional rate control algorithm), ATM Forum/AF-TM 94-0735R1, 1994.

[12] S.P. Abraham, A. Kumar, A stochastic approximation approach for a max–min fair adaptive rate control of ABR sessions with MCRs, Proceedings of INFOCOM, April, 1998.

[13] Y.T. Hou, H.H.-Y. Tzeng, S.S. Panwar, A generalized max–min rate allocation policy and its distributed implementation using the ABR flow control mechanism, Proceedings of INFOCOM, April, 1998.

[14] Y.T. Hou, H. Tzeng, S.S. Panwar, A simple ABR switch algorithm for the weighted max–min fairness policy, Proceedings of IEEE ATM'97 Workshop, May, 1997, pp. 329–338.

[15] D. Hughes, Fair share in the context of MCR, ATM Forum/AF-TM 94-0977, 1994.

[16] N. Yin, Max–min fairness vs. MCR guarantee on bandwidth allocation for ABR, Proceedings of IEEE ATM'96 Workshop, August, 1996.

[17] S. Fahmy, R. Jain, S. Kalyanaraman, R. Goyal, B. Vandalore, On determining the fair bandwidth share for ABR connections in ATM networks, Proceedings of the IEEE International Conference on Communications (ICC), June, 1998.

[18] A. Charny, D. Clark, R. Jain, Congestion control with explicit rate indication, Proceedings of IEEE ICC'95, 1995, pp. 1954–1963.

[19] B. Vandalore, R. Jain, R. Goyal, S. Fahmy, Design and analysis of queue control functions for explicit rate switch schemes, Proceedings of the IC3N'98, October, 1998, pp. 780–786. All our papers and ATM Forum contributions are available through http://www.cis.ohio-state.edu~jain/.

[20] R.J. Simcoe, Test configurations for fairness and other tests, ATM Forum/AF-TM 94-0557, 1994.

[21] S. Kalyanaraman, B. Vandalore, R. Jain, R. Goyal, S. Fahmy, S. Kota, Performance of TCP over ABR with long-range dependent VBR background traffic over terrestrial and satellite ATM networks, Proceedings of LCN, October, 1998.

[22] N. Golmie, Netsim: network simulator, http://www.hsnt.nist.gov/misc/hsnt/prd_atm-sim.html, 1998.

[23] H.J. Kushner, D.S. Clark, Stochastic Approximation Methods for Constrained and Unconstrained Systems, Springer, Berlin, 1978.

Bobby Vandalore *received his BTech degree in 1993 from Indian Institute of Technology, Madras and MS degree in 1995 from The Ohio State University, both in Computer Science. He is currently a PhD candidate at The Ohio State University. His main research interests are in the areas of multimedia communications, traffic management, and performance analysis. He is the author of several papers and ATM Forum contributions. He is a student member of the ACM, the IEEE, and the IEEE Communications and Computer societies.*

Sonia Fahmy *is an assistant professor at the Department of Computer Sciences, Purdue University, IN. She received her PhD degree in Computer and Information Science in 1999 from The Ohio State University. Her primary research interests are in the areas of network architectures and protocols, multicasting, traffic management and quality of service provision. She is the author of several journal and conference papers and ATM Forum contributions. She is a member of Phi Kappa Phi, Sigma Xi, Upsilon Pi Epsilon, the ACM, ACM SIGCOMM, the IEEE, and the IEEE Communications and Computer societies.*

Raj Jain *is very active in the areas of traffic management and quality of service in data networks. His research has influenced the directions of Traffic Management and Testing working groups of ATM Forum. He is an active participant in several other industry forums including Internet Engineering Task Force (IETF), Institute of Electrical and Electronic Engineering (IEEE), American National Institute (ANSI), and Telecommunications Institute of America (TIA). He is a Fellow of IEEE, a fellow of ACM, and serves on the editorial boards of Computer Networks, Computer Communications (UK), and the Journal of High Speed Networks. He is the author of two popular books: "FDDI Handbook: High Speed Networking using Fiber and Other Media" published by Addison-Wesley and "The Art of Computer Systems Performance Analysis" published by Wiley. Dr Jain is on the Board of Directors of MED-I-PRO Systems, LLC, Pamona, CA, and on the Board of Technical Advisors to Nexabit Networks Westboro, MA. He is also a consultant to several networking companies. His publications can be found at http://www.cis.ohio-state.edu/~jain/*

Rohit Goyal *is a senior software engineer at Nexabit Networks. He received a PhD in Computer Science from The Ohio State University in 1999. His primary research interests are traffic management, quality of service, and performance analysis for high-speed networks. He is an active participant in the ATM Forum, IETF, and TIA, and has published several conference and journal papers. He received a BS in Computer Science from Denison University, Granville, OH, and an MS in Computer and Information Science from The Ohio State University.*

Mukul Goyal *is currently a PhD student in CIS Department at The Ohio State University. He received his MS from CIS Department at The Ohio State University in March 1999. He worked as a software engineer in Siemens Communication Software Ltd., Bangalore, India from 1995 to 1997.*