A Definition of General Weighted Fairness and its Support in Explicit Rate Switch Algorithms

Bobby Vandalore, Sonia Fahmy, Raj Jain, Rohit Goyal, Mukul Goyal
The Ohio State University
Department of Computer and Information Science
2015 Neil Avenue Mall, Columbus, OH 43210-1277

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

Abstract

In this paper we give a general definition of weighted fairness and discuss how a pricing policy can be mapped to general weighted (GW) fairness. 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 general fairness. Simulations 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.

1. Introduction

To guarantee a minimum amount of service the user can specify a MCR (minimum cell rate) in ATM ABR (available bit rate) service. The ABR service guarantees that the ACR (allowed cell rate) 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 [12] 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 one recommended in [12]. It can also be easily shown that our definition achieves all the recommended fairness definitions of [12] when appropriate weight functions are used. 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 mapping appropriately 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 [22]. Distributed algorithms [10] that emulated a centralized algorithm were proposed in [5, 17]. Improved, simpler distributed algorithms which achieved max-min fairness were proposed in [13, 4, 9, 15, 18, 11]. Recently, [20, 2] discussed a generalized definition of max-min fairness and its distributed implementation. [19] discussed a weight-based max-min fairness policy and its implementation in ABR service. [7, 21] discussed the fairness in the presence of MCR guarantees.

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 schemes can support non-zero MCRs and achieve the GW fairness. As an example, we show how the ERICA+ switch scheme can be modified to support 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 simple configuration, transient source configuration, link bottleneck configuration, and source bottlenecked configuration. These simulations show that the scheme realizes various fairness definitions in ATM TM 4.0 specification, that are special cases of the generalized fairness. We present an analytical proof of convergence for the modified algorithm in the appendix.

2. General weighted fairness: definition

The following is the definition of some 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. This above definition ensures the allocation is at least MCR.

3. Relationship to pricing/charging policies

Consider a very small interval T of time. The cost C to the customer 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: a) $\partial C/\partial W \geq 0$ b) $\partial C/\partial R \geq 0$ c) $\partial(C/W)/\partial W \leq 0$.

One simple function that satisfies all these requirements is: $C = c + wW + rR$. Here, c is the fixed cost per connection; w is 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: $C_1 = c + wW_1 + rR_1 + mM_1$ and $C_2 = c + wW_2 + rR_2 + mM_2$

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

$C_1/W_1 \leq C_2/W_2 \rightarrow c/W_1 + w + rR_1/W_1 + mM_1/W_1 \leq c/W_2 + w + rR_2/W_2 + mM_2/W_2$

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

$c/(R_1T) + w + r/T + mM_1/(R_1T) \leq c/(R_2T) + w + r/T + mM_2/(R_2T) \rightarrow c/R_1 + mM_1/R_1 \leq c/R_2 + mM_2/R_2 \rightarrow (c + mM_1)/(c + mM_2) \leq R_1/R_2 \rightarrow (a + M_1)/(a + M_2) \leq R_1/R_2$

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. This is the weight policy we have chosen to use 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:

$\mathcal{L}$ = Set of links, $\mathcal{L}_s$ set of links that session s goes through.

$\mathcal{S}$ = Set of sessions, $\mathcal{S}_l$ set of sessions that go through link l. $N = \mid S \mid$.

$\mathcal{A}$ = $(\mathcal{A}_l, l \in \mathcal{L})$ set of of available capacity.

$\mathcal{M}$ = $(\mu_s, s \in S)$, where μ_s is the minimum cell rate (MCR) for session s.

$\mathcal{W}$ = $(w_1, w_2, \ldots, w_N)$ denotes the weight vector.

$\mathcal{R}$ = $(r_1, r_2, \ldots, r_N)$ the current allocation vector (or rate vector).

$\mathcal{G}$ = $(g_1, g_2, \ldots, g_N)$ the general fair allocation. $\mathcal{G}_{\mathcal{S}_l}$ 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., $\mathcal{R} = \mathcal{G}$. Where $g_i \in \mathcal{G}_{\mathcal{S}_l}$ is calculated for each link l as defined in the section 2.

Note the 5-tuple $(\mathcal{S}, \mathcal{L}, \mathcal{C}, \mathcal{W}, \mathcal{R})$ 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 [20, 2]. 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 [20]. 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: $ACR(i) = \mu_i + ExcessFairshare(i)$, where $ExcessFairshare$ 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 GW fairness. We have to ensure the $(ACR - MCR)$ converges to the $ExcessFairshare$. We use the notion of *activity level* to achieve the convergence [16]. A connection's *activity level* ($AL(i)$) is defined as follows.

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

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

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

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 [1].

We expect that the links use their $ExcessFairshare$, but this might not be 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:

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

A switch algorithm can use the above $ExcessFairshare$ term to achieve general fairness. In the next section we show how the ERICA+ switching algorithm is modified to achieve 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 BRM (backward RM) 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 [13].

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 $FairShare$ term by $ExcessFairshare(i)$ and adding the μ_i. The keys steps in ERICA+ which are modified to achieve the GW fairness are shown below:

Algorithm A

At the end of Averaging Interval:

$$\text{Total ABR Cap} \leftarrow \text{Link Cap} - \text{VBR Cap} - \sum_{i=0}^{n} \min(SourceRate(i), \mu_i)$$

$$\text{Target ABR Cap} \leftarrow Fraction \times \text{Total ABR Cap}$$

$$\text{Input Rate} \leftarrow \text{ABR Input Rate} - \sum_{i=0}^{n} \min(SourceRate(i), \mu_i)$$

$$z \leftarrow \frac{\text{Input Rate}}{\text{Target ABR Cap}}$$

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

The $Fraction$ term is dependent on the queue length [3]. Its value is one for small queue lengths and drops sharply as queue length increases. When the $Fraction$ is less than one, $(1 - Fraction) \times TotalABRCap$ is used to drain the queues. ERICA+ uses an hyperbolic function for calculating value of the $Fraction$.

When a BRM is received:

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

$$\text{ER} \leftarrow \mu_i + \max(\text{ExcessFairshare(i)}, \text{VCShare})$$

$$\text{ER}_{RM\_Cell} \leftarrow \text{Min}(\text{ER}_{RM\_Cell}, \text{ER}, \text{Target ABR Cap})$$

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

7. Simulation configurations

We use different configurations to test the performance of the modified algorithm. We assume that the sources are greedy, i.e., they have infinite amount of data to send, and always send data at ACR. Poisson or self-similar sources were not used, since in the presence of these sources (which have varying rates) the GW fair allocation varies dynamically. 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 larger 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 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 a two switches and a bottleneck link. See figure 1.

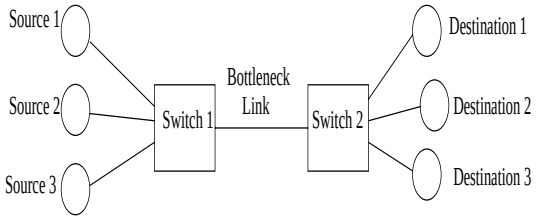


Figure 1. N Sources - N Destinations Configuration

7.2. Source bottleneck

In this configuration, the source S1, is bottlenecked at 10 Mbps, which is below its fairshare (50 Mbps). This configuration tests whether the GW fairness can be achieved in the presence of source bottleneck.

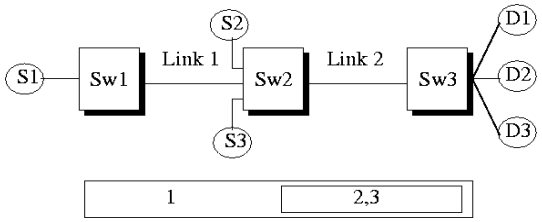


Figure 2. 3 Sources - Bottleneck Configuration

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

This configuration (explained detailedly in [14]) is a combination of upstream and parking lot configuration (See Figure 3). In the configuration all the links are bottlenecked links.

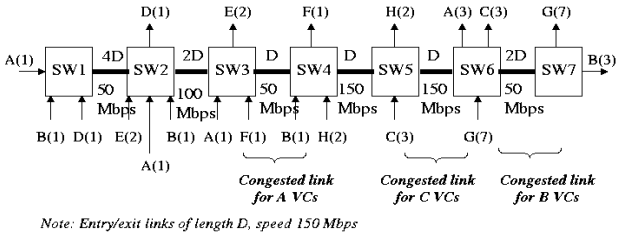


Figure 3. Generic Fairness Configuration - 2

Table 1. Simulation Parameter Values

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

Table 2. 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	18.53	16.64
	2	30	5	35	49.92	49.92
	3	50	5	55	81.30	81.30

7.4. Simulation parameters

The simulations were done on an extensively modified version of NIST simulator [6]. The parameter values used in the different configurations are given in Table 1. The "Averaging Interval" is the period for which the 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 remains constant. The "Target Delay" parameter specifies the desired delay due to this constant queue length at steady state.

8. Simulation results

In this section we give the simulation results for the different configurations.

8.1. Three sources

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

The allocations of these cases are given in Table 2. The following can be observed from the Table 2

- Case 1: a = ∞, MCRs = 0. All weights are equal so the allocation (149.76/3) = 49.92 Mbps for each connection. This is allocation is the same as max-min fair allocation.
- Case 2: a = ∞, 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) = (29.92,39.92,69.92) Mbps.
- Case 3: a = 5, MCRs $\neq$ 1. Hence, the weight function is 5 + MCR. The left over 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) = (16.64, 49.92, 83.2) Mbps.

The Figure 2 shows the ACRs of the three sources for the above three cases. From the figure one can observe that the sources achieve the GW fairness. Steady state queues were of constant length.

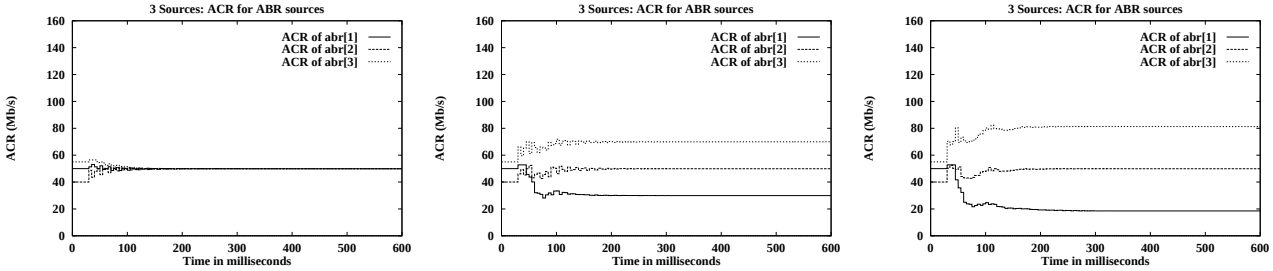
8.2. Three sources: transient

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 parameters values from the cases 1, 2 and 3 of the previous section were used in these simulations. The results of these simulations are given in Table 3. The non-transient (ntr) column give the allocation when transient source 2 is not present, i.e., between 0ms to 400ms and between 800 ms to 1200 ms. The transient (tr) columns give allocation when the transient source 2 is present, i.e., between 400 ms to 800 ms.

The ACR values of the sources for these three simulations are shown in figure 5. It can be seen both from the Table 3 and the graphs that the switch algorithm does converge to the general fairness allocation even in the presence of transient sources. We observed that the algorithm had a good response time from the utilization graph (not shown here due to lack of space).

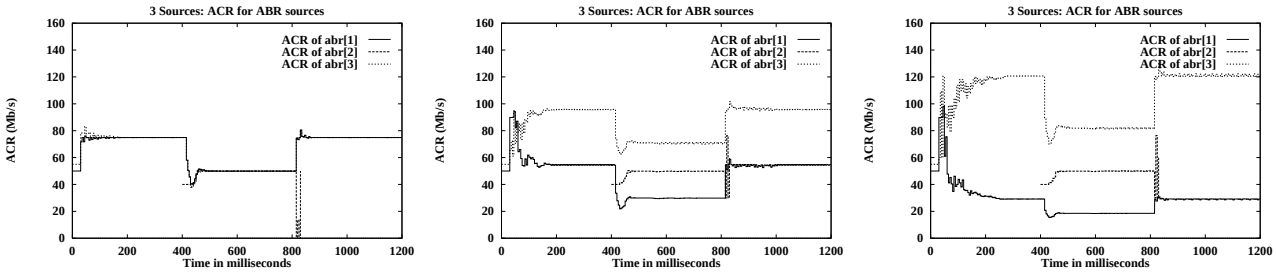
8.3. Source bottleneck

Cases 1, 2 and 3 of section 8.1 were simulated using the three sources bottleneck configuration. The total simulation time 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 rate of at most 10 Mbps, irrespective of its ACR (and ICR).



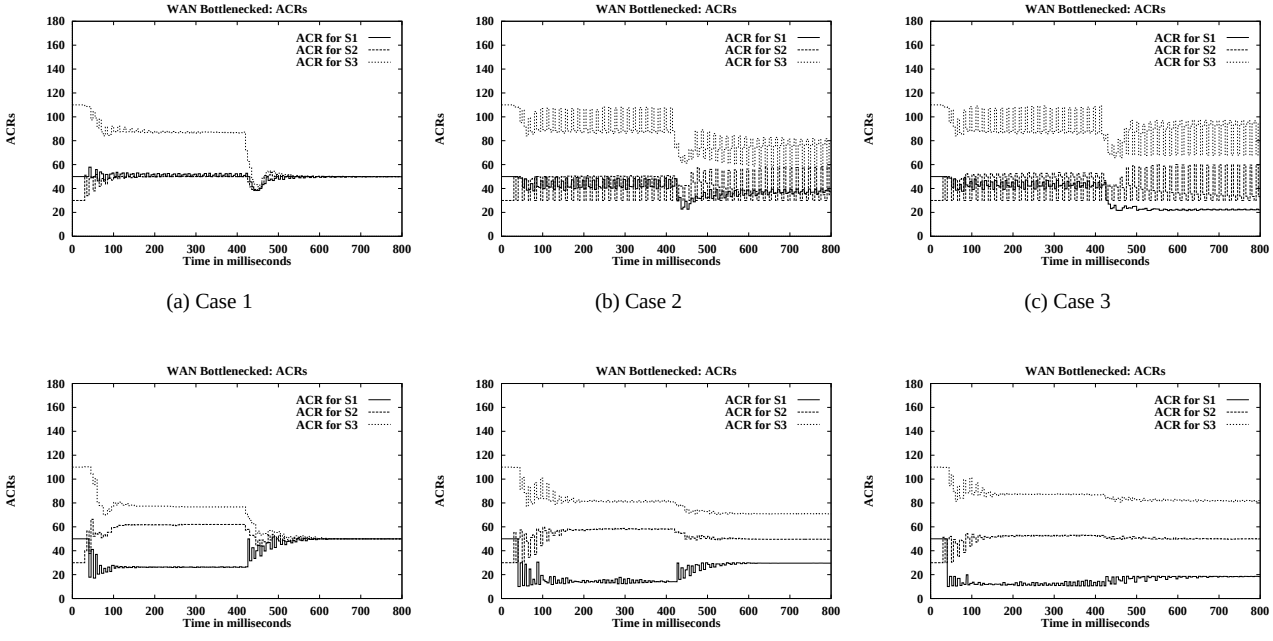
(a) Case 1 (b) Case 2 (c) Case 3

Figure 4. Three Sources: ACR graphs



(a) Case 1 (b) Case 2 (c) Case 3

Figure 5. Three Sources (Transient) : ACR graphs



(a) Case 1 (b) Case 2 (c) Case 3

(d) Case 1 (e) Case 2 (f) Case 3

Figure 6. Three Sources Bottleneck: ACR graphs (with and without measuring Source Rate)

Table 3. Three sources transient configuration simulation results

#	Src #	wt. func.	Exp frshr (ntr)	Actual (ntr) share	Exp frshr (tr)	Actual (tr) 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

nt - non-transient period, tr - transient - NC - not converged

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. But if the switch uses measured source rate then it can correctly estimate the bottlenecked source's rate. Table 4 shows the results both when the switch uses the CCR field and when it measures the source rate. The correct fairness is achieved when the measured source rates are used.

The graphs for these simulations are given in Figure 6. Graphs in Figure 6 (a), (b) and (c) correspond to cases 1, 2 and 3 without using measured source rates. Graphs in Figure 6 (c), (d) and (e) are for the same cases using measured source rates. The switch algorithm uses queue control, to dynamically use part of available ABR capacity to drain the queues. When the queue is large the available ABR capacity is only a fraction of actual capacity. So, the algorithm takes sometime before converging to the correct fairness values. When the CCR value from the RM cells is used, the algorithm is not able to estimate the actual rate at which the source is sending data. So it does not converge in case 2 and case 3 (Figures 6(b) and 6(c)). In case 1 (Figure 6(a), it converged since the bottleneck source's rate (CCR) had the correct value of 50 which is the same allocation it would get in the fair allocation.

8.4. Link bottleneck: GFC-2

In this configuration each link is a bottleneck link. The Figure 7 (a) shows the ACR graphs for each type of VCs. The expected share for VCs of type A, B, C, D, E, F, G, H are 10, 5, 35, 35, 35, 10, 5, and 52.5 Mbps respectively. The actual allocation for these VCs in the simulation was

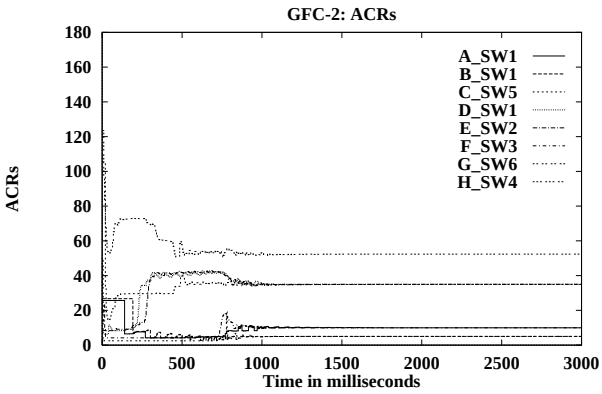


Figure 7. GFC-2 configuration: ACRs of A through H, VCs

9.85, 4.97, 35.56, 35.71, 35.34, 10.75, 5, and 51.95 Mbps respectively. From the Figure and actual allocations it can be seen that the VCs converge to their expected fairshare. This shows that the algorithm works in the presence of multiple link bottlenecks and varying round trip times.

Table 4. Three sources bottleneck configuration simulation results

Case #	Src #	wt.	Exp frshr func.	Using CCR in RM cell	Using Measured CCR
1	1	1	49.92	49.85	49.92
	2	1	49.92	49.92	49.92
	3	1	49.92	49.92	49.92
2	1	1	29.92	NC	29.62
	2	1	49.92	NC	49.60
	3	1	69.92	NC	71.03
3	1	15	18.53	NC	18.42
	2	35	49.92	NC	49.92
	3	35	81.30	NC	81.93

NC - not converged

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 general fairness can be achieved by

using the $ExcessFairshare$ term in the 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 the appendix. The modified algorithm has been tested under different configuration using persistent sources. The simulations 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 maybe incorrect. Hence, it is necessary to used the measured source rate in the presence of source bottlenecks.

Acknowledgments

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

References

[1] A. Charny, D. Clark and R. Jain. Congestion control with explicit rate indication. In *Proc. IEEE ICC'95*, pages 1954–1963, 1995.

[2] S. P. Abraham and A. Kumar. A stochastic approximation approach for a max-min fair adaptive rate control of abr sessions with mcrs. In *Proc. of INFOCOM*, April 1998.

[3] Bobby Vandalore, Raj Jain, Rohit Goyal and S. Fahmy. *Design and Analysis of Queue Control Function for Switch Schemes*[1]. ATM Forum/AF-TM 97-1087, 1998.

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

[5] A. Charny. *An algorithm for rate allocation in a packet-switching, network with feedback. Master's Thesis*, MIT, Cambridge, 1994.

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

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

[8] H. J. Kushner and D. S. Clark. *Stochastic approximation methods for constrained and unconstrained systems*. Springer-Verlag, 1978.

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

[10] J. Mosley. *Asynchronous distributed flow control algorithms. Ph.D Thesis*, Dept. Electrical Engg., MIT, Cambridge, 1984.

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

[12] S. S. Sathaye. *ATM Forum Traffic Management Specification Version 4.0*. ftp://ftp.atmforum.com/pub/approved-specs/af-tm-0056.0000.pdf, 1996.

[13] Shivkumar Kalyanaraman, Raj Jain, Rohit Goyal, Sonia Fahmy and B. Vandalore.

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

[15] K. Siu and T. Tzeng. Intelligent congestion control for abr service in atm networks. *Computer Communication Review*, 24(5):81–106, October 1995.

[16] Sonia Fahmy, Raj Jain, Shivkumar Kalyanaraman, Rohit Goyal and B. Vandalore. On determining the fair bandwidth share for abr connections in atm networks. In *Proc. of the IEEE International Conference on Communications (ICC) 1998*, June 1998.

[17] D. H. K. Tsang and W. K. F. Wong. A new rate-based switch algorithm for abr traffic to achieve max-min fairness with analytical approximation and delay adjustment. In *Proc. IEEE Globecom'96*, 1996.

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

[19] Y. T. Hou, H. Tzeng and S. S. Panwar. A simple abr switch algorithm for the weighted max-min fairness policy. In *Proc. IEEE ATM'97 Workshop*, pages 329–338, May 1997.

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

[21] N. Yin. Max-min fairness vs. mcr guarantee on bandwidth allocation for abr. In *Proc. of IEEE ATM'96 Workshop*, August 1996.

[22] N. Yin and M. G. Hluchyj. On closed-loop rate control for atm cell relay networks. In *Proc. of IEEE INFOCOM*, pages 99–108, 1994.

Appendix: proof of convergence

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 it bottleneck rate during convergence.
- $\sum_{s \in S_l} \mu_i \leq A_l$
- Load factor $z > 0$ and $ER < A_l < LinkRate$

Lemma 1 *The Algorithm A converges to the GW fair allocation, for a session bottlenecked by a link.*

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

Proof: The proof technique used here is similar to the one used in [13]. 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 $n = \mid S_{l_b} \mid -k$. 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 till load factor reaches a value greater than or equal to one. Hence we have shown that if load factor is less than one, the rates increase till the load factor becomes greater than one.

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 the z decreases, till it reaches a value of 1 or less.

From the above two cases it can be seen that load factor oscillates around one and converges to the value of one. Assume that load factor is $z = 1 + \delta$, then the number round trip times for it to converge to one is given by $log_{1+\delta} \mid S_l \mid$. Henceforth, in our analysis we assume that the network is near the steady state that is load factor is near one. 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: $n = 0$, 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 [8]. The behavior is like that of successive values of root acquired in the Newton-Ralphson method for finding roots of a equation. Hence the above equation converges, and the stable values of α_i is given by:

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

Since, we have assumed greedy sources and no bottlenecks in this case, the activity level is one for all sessions. Hence, $\alpha_i = \frac{w_i A}{\sum_{j=1}^{n} w_j}$, which is indeed the desired value for α_i.

Case B: $B \neq 0$, i.e., there are some bottleneck sources. Let β_i be the allocated rate corresponding to r_{b_i}. Let w_{b_i} be the weight for session s_{b_i}. Let $W_b = \sum_{i=1}^{K} w_{bi} AL(b_i)$ 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 link the algorithm calculates a rate β_i which is greater than r_{b_i}, 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 $\frac{\sum_{i=1}^{n} w_i(A_l - A_m)}{W_b + W} = A_l - A_m - A_b \rightarrow W_b = \frac{W A_b}{A_l - A_m - A_b}$

Using the above value for W_b we get:

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

Therefore, $\alpha_i = \frac{w_i(A_l - A_m - A_b)}{W}$ which is the desired values for the α_i. Hence, the sessions bottlenecked at the link l_b do indeed achieve the GW fairness. □

Theorem 1 *Starting at any arbitrary state of the network, if only greedy sources and source bottlenecked or link bottlenecked sources are present the Algorithm A converges to GW fair allocation.*

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

Using lemma 1, we know that each link $l \in \mathcal{L}_i$ does indeed converge to the general fair allocation $\mathcal{G}_l$. The distributed algorithm converges in the above order of links until the whole network is stable and allocation is $\mathcal{G}$. 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. □