Max-Min Fair Rate Control of ABR Connections with Nonzero MCRs*

Santosh P. Abraham and Anurag Kumar
Dept. of Electrical Communication Engg.
Indian Institute of Science
Bangalore 560 012, INDIA
Ph. (+91)-80-334 0855
Fax (+91)-80-334 7991
e-mail: aspaul, anurag@ece.iisc.ernet.in

Abstract

Traffic sources that do not have intrinsic temporal characteristics are expected to be transported over ATM networks using the Available Bit Rate (ABR) service. These sources are amenable to reactive flow control and are expected to use bandwidth left over after servicing the guaranteed QoS services (CBR and VBR). Fair allocation of the available bandwidth to competing ABR connections is based on the concept of Max-Min fairness. The ABR service definition allows sources to specify a Minimum Cell Rate (MCR) that is acceptable to them. Most studies of Max-Min fair rate allocation assume zero MCRs. In this paper, we first develop a natural extension of the concept of Max-Min fair rate allocation to the case of ABR sessions with nonzero MCR values. Then we present a centralised algorithm and discuss the construction of distributed algorithms for obtaining the Max-Min allocation. We show that the Max-Min allocation can be obtained as the solution of a certain vector equation, and discuss how such a perspective can help in designing stochastic approximation type algorithms for obtaining the Max-Min allocation when the available capacity is randomly varying.

Technical Area: Congestion Control in ATM Networks

*Report based on research supported by a grant under the IISc-Nortel Memorandum of Understanding, November 1995.

1 Introduction

Data traffic services (such as file transfer, image transfer, hypermedia document access, etc.) are expected to be carried via the ABR (Available Bit Rate) service in ATM networks. Network bandwidth left over after handling the guaranteed quality of service (QoS) sources (CBR and VBR) will be shared by the ABR connections in an ATM network. Such bandwidth is not guaranteed, and will be variable, hence the ABR sources will need to be controlled [8]. Since the sources of traffic (such as those listed above) that will use the ABR service do not have intrinsic temporal characteristics, they are particularly amenable to reactive flow control in the network.

On the establishment of an ABR connection (i.e., a session), the user specifies to the network both a maximum required bandwidth and a minimum usable bandwidth. These are designated as the Peak Cell Rate (PCR) and the Minimum Cell Rate (MCR); the MCR may be zero. The bandwidth available from the network may vary, but if the ABR connection is accepted, the rate at which it is allowed to send cells must not become less than the negotiated MCR.

A considerable amount of work [17] [12] has been done on binary feedback based control algorithms for congestion control of ABR traffic. However, the feedback schemes they yield are often of a "bang-bang" type. Such schemes, though easy to implement, yield oscillatory rate allocations. These schemes, in general, do not yield a fair rate allocation when implemented in a network.

The problem of fair allocation of bandwidth resources to sessions that can adjust their source rates was first studied in the context of voice networks with variable rate coding (see [6] for a textbook treatment). The notion developed there was called "Max-Min" fairness. Max-Min fair allocation is based on the idea that no session should get extra bandwidth at the expense of a session that has lesser than or equal bandwidth. This notion of Max-Min fair allocation was subsequently adopted for ABR traffic. Charny [9], L.Benmohammed et al [5], Raj Jain et al [14] have developed fair allocation algorithms for the zero MCR case based on this notion of fairness. However this notion does not incorporate a minimum rate requirement for the sessions.

Our work, reported here, focuses on the fair allocation of resources available for ABR connections with nonzero MCR requirements. In the context of fair rate allocation with nonzero MCRs, ITU-T's Recommendation I.371 ([2]) has the following to say:

> "The bandwidth available on an ABR connection is the sum of an MCR, which could be 0, and of a variable cell rate that results from sharing the available bandwidth among ABR connections via a defined allocation policy. A defined allocation policy means that the allocation that the user receives above the minimum cell rate is not only determined by what the user requests or submits, but also by network policy. Defined allocation policies are not subject to standardisation. However, network stability requires that for a given configuration of ABR user bandwidth requests, the allocation policy should support convergence to a stable allocation of bandwidth within the network."

Various ideas for extending the conventional Max-Min fair paradigm to the nonzero MCR case have been discussed by the ATM forum [1]. The notion of fairness that we adopt is a natural generalisation of the zero MCR Max-Min fair notion to the case where sessions have nonzero MCRs. In this paper we present the theoretical basis required for this notion of Max-Min fairness. We note that Kolarov et al [13] have briefly mentioned, without any theoretical justification, the idea of Max-Min fairness that we develop here.

After having developed the theory for Max-Min allocation for the nonzero MCR case, we present a centralised algorithm for computing the Max-Min allocation given a network topology and session configuration. However, centralised algorithms are of little use in practice. To be able to effectively implement this fairness criteria, we need distributed algorithms that work at the switches associated with the links of the network. Such algorithms should work with minimum exchange of control information and yield the Max-Min allocation. We motivate and discuss the theoretical principles involved in the design of distributed algorithms by casting the problem of finding a Max-Min allocation into one of solving a vector equation. We also indicate that this perspective is useful for the introduction of stochastic algorithms.

This paper is organised as follows: Section 1.1 sets down some of the notation that is used throughout the paper; other notation used locally is defined where it arises. In Section 2 we develop the theory for rate allocation with MCR≥ 0 and present a centralised algorithm. In Section 3 we motivate and discuss distributed algorithms for computing the Max-Min solution. We conclude in Section 4 and discuss the possibility of using stochastic algorithms for solving the Max-Min problem when the available link capacity is varying.

1.1 The Model and Notation

We assume that a session comprises a source and a destination node; cells from the source node traverse a fixed sequence of links to reach the destination node. Thus the network topology, the link capacities, the sessions and their routes are all given and static. The cell stream from each source is viewed as a fluid. Each source has an infinite backlog of fluid, and can transfer it to the network at any specified rate (note that a maximum transfer rate from a source can be easily incorporated by augmenting the network topology with a source access link with capacity equal to the source transfer rate limit). Every link has a fixed capacity to be shared among the sessions that use that link. We now describe the notation used.

We first describe some generic notation

- If A is a set, then $\mid A \mid$ denotes the size of, or the number of elements in, the set A.
- ϕ denotes the empty set.
- If $(x_1, x_2, \ldots, x_n)$ is a real valued vector, then $(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$ denotes the elements of the vector ordered in ascending order.

The following is a list of specific symbols that we have used

$\mathcal{S}$ the set of sessions

$\mathcal{L}$ the set of links

C_l the capacity of link $l \in \mathcal{L}$

$\mathcal{C}$ denotes the ordered set $(C_l, l \in \mathcal{L})$

$\mathcal{L}_s$ the set of links used by session $s \in \mathcal{S}$

$\mathcal{S}_l$ the set of sessions through link $l \in \mathcal{L}$

r_i the rate of the ith session, $1 \leq i \leq \mid \mathcal{S} \mid$; $r = (r_1, r_2, \ldots, r_{|\mathcal{S}|})$ denotes the *rate vector*

μ_s the minimum cell rate for session $s \in \mathcal{S}$

$\mathcal{M}$ the set $\{\mu_s : s \in \mathcal{S}\}$

For a rate vector r, and $l \in \mathcal{L}$ denote the total flow through link l by

$$f_l(r) = \sum_{s \in \mathcal{S}_l} r_s$$

Note that the 3-tuple $(\mathcal{L}, \mathcal{C}, \mathcal{S}, \mathcal{M})$ characterises an instance of the bandwidth sharing problem. Thus we will say, for example, that the rate vector r is feasible for $(\mathcal{L}, \mathcal{C}, \mathcal{S}, \mathcal{M})$, or that r is the Max-Min fair rate vector for $(\mathcal{L}, \mathcal{C}, \mathcal{S}, \mathcal{M})$, etc.

2 Max-Min Fair Bandwidth Sharing with Nonzero MCR

In this section we develop the theory of Max-Min allocation for non-zero MCR. To motivate the Max-Min solution with nonzero MCR consider the fair allocation problem for N sessions with MCRs $(\mu_i, i = 1, \ldots, N)$, on a single link with capacity C $(> \sum_{i=1}^{N} \mu_i)$. To solve this problem we consider the following approach. Define the function

$$\gamma(\eta) = \sum_{i=1}^{N} \max(\mu_i, \eta)$$

and solve for η^* such that

$$\gamma(\eta^*) = C$$

Let the allocation r_i for each session $i, i = 1, \ldots, N$ be given by

$$r_i = \max(\mu_i, \eta^*)$$

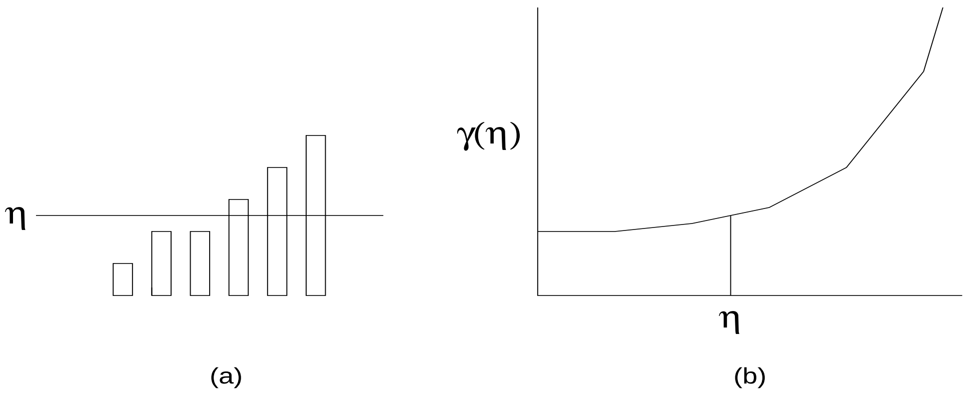

Figure 1: Two depictions of the function $\gamma(\eta)$; the vertical bars in (a) are the MCR values arranged in ascending order; the break points in the piecewise linear curve in (b) are these MCR values; $\gamma(0) = \sum_{i=1}^{N} \mu_i$.

It is useful to picture the function $\gamma(\eta)$ as shown in Figure 1. In part (a) of the figure, the μ_i values are arranged in ascending order and shown as vertical bars of increasing height; the value of η is shown as a line cutting across these bars; for this value of η each session with $\mu_i \leq \eta$ takes the value η, and every other session takes the value μ_i; the total flow is $\gamma(\eta)$. With part (a) of the figure in mind, we plot the function $\gamma(\eta)$ vs. η in part (b); the function is piecewise linear and the slope of the function at an η is the number of sessions with MCR less than η. Observe that with η as shown in part (a) of Figure 1, and $r_i = \max(\mu_i, \eta)$ $1 \leq i \leq N$, η is the minimum flow over all the sessions. This minimum flow is maximised by setting $\eta = \eta^*$. Observe that this allocation has the property that a session can increase its rate only at the expense of some other session.

It is interesting to observe that the session rates r_i, $1 \leq i \leq N$, obtained above, solve the following least squares rate balancing problem.

$$\text{Minimize} \quad \sum_{i=1}^{N} \left(r_i - \frac{C}{N} \right)^2$$

$$\begin{aligned} \text{s.t.} \quad r_i &\geq \mu_i \quad 1 \leq i \leq N \\ \sum_{i=1}^{N} r_i &= C \end{aligned}$$

This again emphasises that our approach extends the approach for zero MCRs in a natural way. We do not, however, pursue this least squares rate balancing view further in this paper.

To make the above idea precise for a network of links and sessions with arbitrary paths we need the following definitions. The development of the theory here follows closely the development of of the theory of Max-Min allocation for zero MCRs presented in [6].

Definition 2.1 *We call a rate vector r* **feasible** *for the problem* $(\mathcal{L}, \mathcal{C}, \mathcal{S}, \mathcal{M})$ *if*

$$\text{for all} \quad s \in \mathcal{S}, \quad r_s \geq \mu_s, \text{ and for all} \quad l \in \mathcal{L}, \quad f_l(r) \ = \ \sum_{s \in \mathcal{S}_l} r_s \leq C_l.$$

Note that the set of feasible vectors is non-empty iff

$$\forall l \in \mathcal{L} \qquad \sum_{s \in \mathcal{S}_l} \mu_s \leq C_l$$

We will assume that this is so, *with strict inequality*, in all the following discussions. Note that such feasibility will be ensured by an admission control procedure for ABR connections.

Definition 2.2 *A feasible rate vector r is* **Max-Min fair** *for* $(\mathcal{L}, \mathcal{C}, \mathcal{S}, \mathcal{M})$ *if it is not possible to increase the rate of a session s, while maintaining feasibility, without reducing the rate of some session p with $r_p \leq r_s$.*

Note that the above definition precisely captures the idea of Max-Min fairness outlined in the beginning of this section. However, since we are now interested in a network, the implications are a little more complicated. The following definitions and the subsequent theorem provide a more tractable characterisation of the Max-Min fair rate vector.

Definition 2.3 *Given a rate vector r, a link l is said to be a* **bottle-neck link** *for a session j if*

(i) link l is saturated, i.e., $f_l(r) = C_l$, and

(ii) for all the sessions $s \in \mathcal{S}_l$, such that $r_s > \mu_s$, $r_s \leq r_j$; i.e., every session in l, that is not at its MCR, has flow no more than that of session j, or equivalently

$$r_s \leq \max(\mu_s, r_j)$$

Recalling the notation in Section 1.1, we give the following definition.

Definition 2.4 *Let* $x = (x_1, x_2, \cdots, x_n), y = (y_1, y_2, \cdots, y_n) \in \mathcal{R}^n$. *Then* y *is defined as* **lexicographically larger** *than* x *(denoted* $>_{lex}$*) if* $\tilde{y}_1 > \tilde{x}_1$*, or if* $\tilde{y}_1 = \tilde{x}_1$ *then* $\tilde{y}_2 > \tilde{x}_2$*, etc.*

The following theorem gives two equivalent characterisations of the Max-Min rate vector.

Theorem 2.1 *If* r *is a feasible rate vector, then the following statements are equivalent:*
(i) r *is Max-Min fair.*
(ii) Every session $s \in \mathcal{S}$ *has a bottle-neck link.*
(iii) r *is lexicographically the largest among all feasible rate vectors.*

Proof: See Appendix A.
□

Theorem 2.1 states that the Max-Min vector is the lexicographic maximum in the feasible set of rate vectors. This implies that every rate in the Max-Min vector is greater than or equal to the maximum of the minimum rate in any feasible rate vector. To obtain the minimum rate in the Max-Min rate vector consider the following linear program.

Define the $\mid \mathcal{L} \mid \times \mid \mathcal{S} \mid$ matrix A with 0-1 elements, whose element in row l and column s is 1 if the session s flows through link l; otherwise that element is 0. Let $\underline{C}$ denote a column vector of link capacities with row indices corresponding to those of the matrix A. Let $\underline{\mu}$ denote the row vector of MCR values with column indices corresponding to those of the matrix A. Let $\underline{x}$ denote the row vector of the amounts by which the flows in a feasible vector r are more than the corresponding MCR values. In terms of this notation, the required linear program can be written as

$$\begin{aligned} \max \quad & \eta \\ A(\underline{x} + \underline{\mu}) \quad & \leq \quad \underline{C} \end{aligned}$$

$$\begin{aligned} \forall s \in \mathcal{S} \quad x_s + \mu_s &\geq \eta \\ \underline{x}, \eta &\geq 0 \end{aligned}$$

Consider increasing η in the above linear program while allocating to each session the rate as the maximum of η and the session's MCR. Increase η till the capacity of at least one link is fully utilised (call such a link as *saturated*). In other words, defining

$$\gamma_l(\eta_l) = \sum_{s \in \mathcal{S}_l} \max(\mu_s, \eta_l)$$

solve

$$\gamma_l(\eta_l) = C_l$$

for every $l \in \mathcal{L}$, to yield $\eta_l, l \in \mathcal{L}$; pick the smallest of these $\eta_l, l \in \mathcal{L}$. The value η so obtained can be shown to be the solution of the above linear program. Notice that the saturated link (or $\text{argmin}_{l \in \mathcal{L}} \eta_l$) will be the bottleneck link for all the sessions through it. Consider now the network with all saturated links removed and the bandwidth utilised by their sessions in the other links subtracted from the links' capacities and form the same linear program in the reduced network. Continue this till all sessions have at least one saturated link. Notice that the rate allocation of the sessions so obtained is such that every session has at least one bottleneck link and hence the allocation is Max-Min fair.

2.1 A Centralised Algorithm

We now present a formal algorithm for computing the Max-Min solution. Each link l solves a "single link allocation problem" (as described earlier) on the sessions through it. The algorithm operates by creating bottlenecks for a subset of the sessions at each iteration. It terminates when all sessions have at least one bottleneck link. The algorithm can be seen to be an extension of the one for Max-Min rate allocation with zero MCR [6].

Algorithm 2.1

The iterations are indexed by $k, k \geq 1$. At the end of the kth iteration, the following variables are defined

$\mathcal{S}(k)$: the set of unbottlenecked sessions

$\mathcal{L}(k)$: the set of unsaturated links

$f_l(r(k))$: the total flow in link l

$F_l(k)$: the total flow in link l due to the bottlenecked sessions

$n_l(k)$: the number of unbottlenecked sessions through link l

$\eta_l(k)$: result obtained by distributing the residual capacity of link l (after removing flow due to bottlenecked sessions) among the unbottlenecked sessions; for links with no unbottlenecked sessions, $\eta_l(k) = \eta_l(k-1)$.

Initialisation:
$k = 0, \mathcal{S}(0) = \mathcal{S}, \mathcal{L}(0) = \mathcal{L}$, and $\forall l \in \mathcal{L}, n_l(0) = \mid \mathcal{S}_l \mid, \gamma_l(0) = 0, \eta_l(0) = 0, F_l(0) = 0$.

While $\mathcal{S}(k)$ is not empty, do

1. $k \leftarrow k+1$
2. Calculate $\eta_l(k)$

   if $l \in \mathcal{L}(k-1)$ and $\mathcal{S}_l \cap \mathcal{S}(k-1) \neq \phi$, then compute $\eta_l(k)$ by solving

   $$C_l - F_l(k-1) - \sum_{s \in \mathcal{S}_l \cap \mathcal{S}(k-1)} \max(\eta_l(k), \mu_s) = 0$$

   Otherwise

   $$\eta_l(k) = \eta_l(k-1)$$

3. Set the virtual rate of each unbottlenecked session to the minimum of $\eta_l(k)$ along the session's path, i.e.,

   $$\hat{r}_s(k) = \begin{cases} \min_{l \in \mathcal{L}_s} \eta_l(k) & s \in \mathcal{S}(k-1) \\ \hat{r}_s(k-1) & \text{otherwise} \end{cases}$$

4. Calculate the rate of each session as the maximum of the MCR and the virtual rate

   $$r_s(k) = \max(\mu_s, \hat{r}_s(k))$$

5. Calculate the new total flow through each link $l \in \mathcal{L}$.

$$f_l(r(k)) = \sum_{s \in \mathcal{S}_l} r_s(k)$$

6. Find the new set of unsaturated links.

$$\mathcal{L}(k) = \{l : f_l(r(k)) < C_l\}$$

7. Find the new set of unbottlenecked sessions; these are the sessions all of whose links are in $\mathcal{L}(k)$.

$$\mathcal{S}(k) = \{s : \mathcal{L}_s \subseteq \mathcal{L}(k)\}$$

8. Find the flow in each link $l \in \mathcal{L}$ due to the bottlenecked sessions.

$$F_l(k) = \sum_{s \in \mathcal{S}_l \setminus \mathcal{S}(k)} r_s(k)$$

□

The parameter $\eta_l(k)$ is called the *link control parameter*; if a session using link l is above its MCR then its rate must be less than or equal to $\eta_l(k)$; considering this requirement for all the links in the path of a session, if the session s is above its MCR, its rate can be no more than the *virtual rate* $\hat{r}_s(k)$. The algorithm essentially computes the virtual rate and then, in step 4, uses the MCR to obtain the rate allocation. After every iteration, at least one link is a bottleneck for some sessions. The following theorem shows that the rate allocation obtained is the Max-Min rate as per Definition 2.2.

Theorem 2.2 *If M denotes the number of iterations executed by Algorithm 2.1, then $r_s(M)$, $\forall s \in \mathcal{S}$ is the Max-Min allocation.*

Proof: See Appendix B

3 Distributed Algorithms

Algorithm 2.1 is a centralised scheme. At each iteration information about every link is required either to assert whether sessions have bottlenecks and/or for computing the link control parameters. Thus centralised algorithms are of little value in practice. In this section we discuss distributed algorithms that compute the link control parameter (c.f., Section 2.1) at the links (in a practical network this computation would take place at the switches associated with the links). We begin by briefly discussing an extension of Charny's Algorithm [9], which imitates the centralised Algorithm 2.1 in a distributed manner. We then discuss distributed algorithms that operate without information exchange between the links.

3.1 Charny's Algorithm

Note that a link can compute its own control parameter (c.f., Step 2 of Algorithm 2.1) once all sessions through it that have bottlenecks in other links can be determined and the rates they use are found (i.e., computing $F_l(k-1)$ in Algorithm 2.1). Hence a message passing methodology is needed to make such information available at each link; then a distributed algorithm can be implemented. In [9], Charny proposes a single bit based message passing scheme for this purpose. The work in [9] was done with the assumption of zero MCRs. We can use her message passing scheme in conjunction with the link control parameter computation in step 2 of Algorithm 2.1 to give a distributed algorithm for Max-Min computation for the case where MCR's are nonzero.

3.2 Autonomous Distributed Algorithms

In Charny's algorithm, the computation of the link control parameter at a link required bottleneck information from other links. In this section we discuss algorithms that do not require an explicit exchange of information between the links. To motivate the construction of such algorithms consider Algorithm 2.1. If M is the number of iterations executed before termination, then the set $\mathcal{L}(M)$ gives the set of links that are not bottlenecks for any of the sessions; i.e., even after the algorithm terminates, these links have spare capacity even though all sessions are bottlenecked. Consider the link control values computed by Algorithm 2.1; if we retain the link control values computed for the links that are bottlenecks for some sessions, and increase the link control values of the "nonbottleneck" links, the rate allocation still remains Max-Min fair. Note that at each execution of Step 2 of Algorithm 2.1, the link control parameter is computed to maximise link utilisation. The theorem stated below relates the two ideas of maximising link utilisation and obtaining the Max-Min allocation.

Theorem 3.1 *Given M as the number of iterations that are executed by Algorithm 2.1 to solve a given Max-Min problem $(\mathcal{L}, \mathcal{C}, \mathcal{S}, \mathcal{M})$. Let $\mathcal{L}(M)$ be as given in Algorithm 2.1. Consider any vector $(\eta_l,\ l \in \mathcal{L})$ such that*

$$\begin{array}{lrcl} (i) & \min_{j \in \mathcal{L}_s} \eta_j & = & \min_{j \in \mathcal{L}_s \setminus \mathcal{L}(M)} \eta_j \quad \text{for all } s \in \mathcal{S} \\ (ii) & \sum_{s \in \mathcal{S}_l} \max(\mu_s, \min_{j \in \mathcal{L}_s} \eta_j) & = & C_l \quad \text{for all } l \in \mathcal{L} \setminus \mathcal{L}(M) \end{array}$$

Then the allocation obtained as

$$r_s = \max(\mu_s, \min_{j \in \mathcal{L}_s} \eta_j)$$

is the Max-Min allocation.

Proof: See Appendix C.
□

Consider the case when there are no nonbottleneck links, i.e., $\mathcal{L}(M)$ is empty and every link is a bottleneck for at least one session. Let $\underline{\eta} = (\eta_l,\ l \in \mathcal{L})$ be the vector of link control parameters and $\underline{C} = (C_l,\ l \in \mathcal{L})$ be the vector of link control parameters. Define a vector function $\underline{f}(\underline{\eta}) = (f_l(\underline{\eta}),\ l \in \mathcal{L})$ with

$$f_l(\underline{\eta}) = \sum_{s \in \mathcal{S}_l} \max(\mu_s, \min_{j \in \mathcal{L}_s} \eta_j) \quad \forall l \in \mathcal{L}$$

then by Theorem 3.1, the Max-Min allocation can be obtained by solving

$$\underline{f}(\underline{\eta}) = \underline{C}$$

For each value of $\underline{\eta}$, $f_l(\underline{\eta})$ is just the total flow in link l. A distributed algorithm can be viewed as one that solves the above vector equation. One approach to deriving a distributed algorithm can be the following. Given a bottleneck link whose capacity is not fully utilised, we increase the link control parameter until the capacity is fully utilised. Similarly, given a link with total flow through it exceeding the link capacity, we decrease the link control parameter to maintain feasibility. One such additive increase/decrease algorithm was presented by Hayden in [11]. Some simple modifications to it were made by Mosely in [15] in order to prevent the link control parameters from increasing infinitely in non-bottleneck links. Hayden's Algorithm has been adapted to our framework in the following algorithm.

Algorithm 3.1

Initialisation:
$k = 0, \forall l \in \mathcal{L}$
Do forever

1. $k \leftarrow k + 1$
2. For all $l \in \mathcal{L}$ update the link control numbers.
$$\eta_l(k) = \max_{s \in \mathcal{S}_l} \hat{r}_s(k-1) + \frac{C_l - \sum_{s \in \mathcal{S}_l} r_s(k-1)}{n_l}$$
3. For each session $s \in \mathcal{S}$, update the virtual session rates.
$$\hat{r}_s(k) = \min_{l \in \mathcal{L}_s} \eta_l(k)$$
4. Each session $s \in \mathcal{S}$ calculates is actual allowed rate.
$$r_s(k) = \max(\hat{r}_s(k), \mu_s)$$

□

Theorem 3.2 *The rates $r_s(k)$, $s \in \mathcal{S}$, Algorithm 3.1 converge to the Max-Min value.*

The proof of this algorithm with no MCR requirements was provided by Mosely in [15]. We have proved Theorem 3.2 along the lines of Mosely's proof. The details [4] are not provided due to lack of space. Another algorithm which uses a multiplicative increase/decrease scheme at the links has been discussed in [10]. Since these algorithms attempt to solve $\underline{f}(\underline{\eta}) = \underline{C}$, unlike the one-bit feedback based increase/decrease algorithms [17] [12], they can be shown to yield Max-Min fair rates. We have proved convergence for a class of such synchronous algorithms [4]. Note that for such algorithms the MCR's need not be known at the switches; thus time varying MCR's can be supported; this may be useful with sessions that do not behave as if they are infinitely backlogged.

4 Conclusion: Towards Stochastic Algorithms

In this paper we developed a theory for Max-Min fair bandwidth allocation for sessions with nonzero MCRs. The theory developed is a natural generalisation of the theory for Max-Min allocation of bandwidth for sessions without MCR requirements; the Max-Min fair rate vector obtained for the nonzero MCR case is also the lexicographic maximum of all the feasible rate vectors. We presented a centralised algorithm for the computation of the Max-Min fair rate vector. Finally we motivated a class of distributed algorithms by showing that the Max-Min vector is obtained by solving a vector equation.

The theory of Max-Min fair allocation developed so far assumes that the capacity available for ABR sessions is constant. As indicated in the beginning of this paper, the ABR sessions are expected to utilise the bandwidth left over after servicing the guaranteed QoS traffic such as CBR and VBR. The inherently bursty nature of VBR traffic implies that the left over bandwidth would be varying. In Section 3.2 we showed that the Max-Min allocation could be obtained by solving a vector equation. If we view the available capacity as a mean capacity corrupted by noise, then the Max-Min allocation problem can be viewed as the search for the root of a function with noisy observations. Such problems are effectively handled using stochastic approximation algorithms [7], and we are currently exploring the usefulness of these techniques in the ABR flow control problem [4].

Appendix A

Proof of Theorem 2.1:

$(i) \Rightarrow (ii)$

Let r be max-min fair. Let $\exists s \in \mathcal{S}$ such that s does not have a bottle-neck link. Then, for each $l \in \mathcal{L}_s$ do one of the following

(a) if $C_l > f_l(r)$, then let $\delta_l = C_l - f_l(r)$

(b) if $C_l = f_l(r)$ and $\exists k \in \mathcal{S}_l$ such that $r_k \geq \mu_s$ and $r_k > r_s$, then let $\delta_l = r_k - r_s$

(c) if $C_l = f_l(r)$ and $\exists k \in \mathcal{S}_l$ such that $\mu_k > r_s$ and $r_k > \mu_k$, then let $\delta_l = r_k - \mu_k$

Finally, let

$$\delta = \min_{l \in \mathcal{L}_s} \delta_l$$

Adding δ to r_s, and subtracting it from an appropriate r_p with $r_p > r_s$ if necessary to maintain feasibility, we can increase r_s without affecting any r_p with $r_p \leq r_s$. We have a contradiction.

$(ii) \Rightarrow (i)$

Every session s has a bottle-neck link, say l_s. If we want to increase r_s, then we must decrease the rate of some other session in l_s. Every other session p in l_s, however, either has $r_p \leq r_s$, or $r_p = \mu_p$; reducing the rate of a session with $r_p = \mu_p$ will violate feasibility.

$(i) \Rightarrow (iii)$:

If r is max-min fair, then increasing r_s while maintaining feasibility requires that some r_p with $r_p \leq r_s$ be decreased. The newly obtained sequence is lexicographically smaller. Hence the max-min fair vector is lexicographically the largest in the feasible set.

$(iii) \Rightarrow (ii)$:

Suppose there is a session s that has no bottleneck link. Then we can increase r_s by an amount $\epsilon > 0$ without decreasing any of the other flows $j \in \mathcal{S}$ with $r_j < r_s$, while maintaining feasibility. The new flow is lexicographically larger than r, thus contradicting (iii).

□

Appendix B

Proof of Theorem 2.2:

Lemma: Let M be the number of iterations executed by Algorithm 2.1. Let $k \in \{1, \ldots, M\}$. Let $l \in \mathcal{L}(k-1) \backslash \mathcal{L}(k)$, i.e., the link l is saturated at iteration k. The sequence of link control values $\eta_l(i), i = 0, \ldots, k$ is a nondecreasing sequence.

Proof of the Lemma: Note that $\eta_l(1) > \eta_l(0)$ since $C_l > 0$ for all $l \in \mathcal{L}$, hence the statement is true for $k = 1$. Now consider $k \geq 2$ and $1 < i \leq k$. $\eta_l(i-1)$ satisfies the equation

$$C_l - F_l(i-2) - \sum_{s \in \mathcal{S}_l \cap \mathcal{S}(i-2)} \max(\mu_s, \eta_l(i-1)) = 0 \tag{1}$$

and $\eta_l(i)$ satisfies the equation

$$C_l - F_l(i-1) - \sum_{s \in \mathcal{S}_l \cap \mathcal{S}(i-1)} \max(\mu_s, \eta_l(i)) = 0 \tag{2}$$

Recall that $\mathcal{S}(i), i \geq 0$ is a strictly decreasing sequence of sets of sessions. Observe that $F_l(i-1)$ exceeds $F_l(i-2)$ by the total flow of sessions in link l that leave $\mathcal{S}(i-2)$ at the end of the $(i-1)th$ iteration, i.e.,

$$F_l(i-1) = F_l(i-2) + \sum_{s \in \mathcal{S}_l \cap (\mathcal{S}(i-2) \backslash \mathcal{S}(i-1))} \max(\mu_s, \hat{r}_s(i-1)) \tag{3}$$

Note that the sessions $s \in \mathcal{S}_l \cap (\mathcal{S}(i-2) \backslash \mathcal{S}(i-1))$ are the sessions through link l that have one of their other links saturated in iteration $i-1$. Using Equation (3), we can rewrite Equation (2) as follows

$$C_l - F_l(i-2) - \sum_{s \in \mathcal{S}_l \cap (\mathcal{S}(i-2) \backslash \mathcal{S}(i-1))} \max(\mu_s, \hat{r}_s(i-1)) - \sum_{s \in \mathcal{S}_l \cap \mathcal{S}(i-1)} \max(\mu_s, \eta_l(i)) = 0 \tag{4}$$

From step 3 of Algorithm 2.1 we have

$$\hat{r}_s(i-1) \leq \eta_l(i-1) \quad \forall s \in \mathcal{S}_l \cap (\mathcal{S}(i-2) \backslash \mathcal{S}(i-1)) \tag{5}$$

Hence Equation 4 holds only if

$$\eta_l(i) \geq \eta_l(i-1) \tag{6}$$

This proves the lemma.

Returning to the proof of the theorem, it is sufficient to show that each $s \in \mathcal{S}$ has a bottleneck link. Consider $s \in \mathcal{S}$; there is a k such that $s \in \mathcal{S}(k-1)\backslash\mathcal{S}(k)$, i.e., s has at least one of its links saturated at iteration k, let l_s be one such link, then note that $l_s \in \mathcal{L}_s \cap (\mathcal{L}(k-1)\backslash\mathcal{L}(k))$. Now

$$F_{l_s}(k-1) + \sum_{q \in (\mathcal{S}_{l_s}\backslash\{s\}) \cap \mathcal{S}(k-1)} r_q(k) + r_s(k) \;=\; C_{l_s} \tag{7}$$

Now $\forall q \in \mathcal{S}_{l_s}$ note that $q \in \mathcal{S}(m_q - 1)\backslash\mathcal{S}(m_q)$ for some $m_q \in \{1, \ldots, k\}$. It follows that

$$r_q(k) = r_q(m_q) = \max(\mu_q, \min_{l \in \mathcal{L}_q} \eta_l(m_q)) \leq \max(\mu_q, \eta_{l_s}(m_q)) \tag{8}$$

By the Lemma $\eta_l(m), m = 1, \ldots, k$ is a nondecreasing sequence, hence we get

$$r_q(k) \leq \max(\mu_q, \eta_{l_s}(m_q)) \leq \max(\mu_q, \eta_{l_s}(k)) \tag{9}$$

Recall that $\eta_{l_s}(k)$ satisfies

$$F_{l_s}(k-1) + \sum_{q \in \mathcal{S}_{l_s} \cap \mathcal{S}(k-1)} \max(\mu_q, \eta_{l_s}(k)) = C_{l_s} \tag{10}$$

From Equations (8) and (9) it follows that for Equation (10) to hold, it must be the case that

$$r_s(k) = \max(\mu_s, \eta_{l_s}(k)) \tag{11}$$

Thus $\eta_{l_s}(k) \leq r_s(k)$ and $\forall q \in \mathcal{S}_{l_s}$ we have

$$r_q(k) \leq \max(\mu_q, r_s(k)) \tag{12}$$

Hence by Definition 2.3, l_s is a bottleneck link for $s \in \mathcal{S}$.
Thus every session $s \in \mathcal{S}$ has a bottleneck link.
Note: In fact, Equation (9) must be an equality for all $q \in \mathcal{S}_{l_s} \cap \mathcal{S}(k-1)$; hence l_s is a bottleneck link for all $q \in \mathcal{S}_{l_s} \cap \mathcal{S}(k-1)$.
□

Appendix C

Proof of Theorem 3.1: It is sufficient to show that every session $s \in \mathcal{S}$ has a bottleneck link. Consider any $s \in \mathcal{S}$. Let $l_s \in \mathcal{L}_s \backslash \mathcal{L}(M)$ be such that

$$\eta_{l_s} = \min_{j \in \mathcal{L}_s} \eta_j$$

Hence,

$$r_s = \max(\mu_s, \eta_{l_s})$$

It follows that $\forall q \in \mathcal{S}_{l_s}$

$$r_q \leq \max(\mu_q, \eta_{l_s}) \leq \max(\mu_q, r_s)$$

Hence by Definition 2.3, l_s is a bottleneck link for $s \in \mathcal{S}$.
□

References

[1] The ATM Forum Technical Committee Traffic Management Specification Version 4.0, April 1996

[2] ITU-T, "Traffic Control and Congestion Control in B-ISDN", Recommendation I.371, July 1995.

[3] S.P. Abraham and Anurag Kumar, "Algorithms for Max-Min Fair Rate Control of ABR Connections with Nonzero MCR in ATM Networks", manuscript, Dept. of Electrical Commn. Engg, Indian Institute of Science, Bangalore, 560012, August 1996.

[4] S.P. Abraham and Anurag Kumar, "Stochastic Approximation Algorithms for Fair Rate Control of ABR Connections", manuscript, Dept. of Electrical Commn. Engg, Indian Institute of Science, Bangalore, 560012, February 1997.

[5] L. Benmohamed and S.M. Meerkov, "Feedback Control of Congestion in Store-and-Forward Networks: The Case of Multiple Congested Nodes", Control Group Report No. CGR-94-0-6, The University of Michigan, March 1994.

[6] D. Bertsekas and R. Gallager, *Data Networks,* Prentice-Hall Inc., 1992.

[7] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computing,* Prentice-Hall Inc., 1989.

[8] F. Bonomi, K.W. Fendick, "The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service", *IEEE Network* March/April 1995, pp 25-39.

[9] Anna Charny, "An algorithm for rate allocation in a packet-switching, network with feedback," Master's thesis, MIT, Cambridge, May 1994.

[10] C. Fulton, San-qi Li and C.S. Lim, "UT: ABR Feedback Control with tracking", *Preprint.*

[11] H.P. Hayden, "Voice flow control in integrated packet networks", Master's thesis, MIT, Cambridge, October 1981.

[12] Nanying Yin and M.G. Hluchyj "On Closed-Loop Rate Control for ATM Cell Relay Networks", Proc. *INFOCOM'94*, 1994, PP 99-108.

[13] A. Kolarov and G. Ramamurthy, "A Control Theoretic Approach to the Design of Closed Loop Rate Based Flow Control For High Speed ATM Networks", *Submitted to INFOCOM'97*

[14] R. Jain, S. Kalyanraman, R. Viswanathan and R. Goyal, " A Sample Switch Algorithm", *ATM Forum/95-0178,* February 1995.

[15] J. Mosely, "Asynchronous Distributed Flow Control Algorithms," PhD thesis, MIT, Cambridge, October 1984.

[16] Y. Chang N. Golmie and David Su, "A rate based flow control switch design for abr service, in an atm network", Preprint.

[17] K.Y. Siu and H.Y. Tzeng, "Intelligent Congestion Control for ABR Service in ATM Networks", *ACM SIGCOMM Comp. Comm. Rev.* Vol.25, No.5,1985 pp 81-106.