Fast, Fair and Frugal Bandwidth Allocation in ATM Networks

Yair Bartal* Martin Farach-Colton† Shibu Yooseph‡ Lisa Zhang§

July 7, 1998

Abstract

ATM networks are used to carry a variety of types of traffic. For some types of traffic, in particular *Available Bit Rate (ABR)* traffic, the bandwidth of a network is typically insufficient to satisfy the requests of all the sessions, and so some fair allocation scheme must be devised. The ATM Forum, the standards setting body for ATM networks, has specified that the fairness criterion for ABR traffic should be *Max-Min Fairness*, which intuitively means that raising the bandwidth of any session comes at the expense of some other session of no greater bandwidth.

Protocols to allocate bandwidth to sessions in a max-min fair manner are an important part of a network design. For a protocol to be realistic, it must conform to the *Resource Management (RM) Cell* mechanism specified by the ATM Forum. Such RM Cells get sent as a constant fraction of all cells sent by the source, however they have only a few fields. RM Cells are the only means of communication allowed between switches so any reasonable protocol is totally distributed and asynchronous, since the RM Cell mechanism does not easily lend itself to synchronization.

Finally, RM Cells must be handled very quickly at the switches. We call a protocol *frugal* if the switch spends $O(1)$ computation on each RM Cell it receives, and it has $O(1)$ local space for each session through it.

Recently, several frugal RM Cell protocols have been proposed for ABR traffic, but none have been shown to converge to the max-min fair state. Protocols which are known to converge in a linear number of maximum roundtrip times require RM Cell processing which is linear in the number of session at a switch. In this paper, we give a frugal RM Cell protocol for ABR that matches the convergence time of the fastest known non-frugal protocol.

A second type of ABR traffic is the *Minimum Cell Rate (MCR)* type, where every session can specify a minimum amount of bandwidth. The max-min fair allocation should then respect these MCR requests. We extend our results to give the first frugal RM Cell protocol for MCR and achieve a quadratic convergence rate.

*Bell Laboratories. yair@research.bell-labs.com
†Bell Laboratories. farach@research.bell-labs.com
‡DIMACS. yooseph@dimacs.rutgers.edu
§Bell Laboratories. ylz@research.bell-labs.com

1 Introduction

Asynchronous Transfer Mode (ATM) technology is critical for modern communication networks, which are being designed to carry many different types of traffic. An ATM network consists of switches with capacities. A *session* is specified by a path in the network, along with a request for bandwidth. The capacity of a switch is parcelled out to the sessions going through that switch. Some sessions may have bandwidth allocated on a *Quality of Service (QoS)* basis. In this case, their entire bandwidth requests will be fulfilled at every switch they traverse. However, not all sessions require a guarantee of bandwidth. Instead, some client may be willing to take a variable amount of bandwidth in exchange for, say, lower costs. In this paper, we focus on such non-QoS sessions, assuming that all the QoS requests along with the necessary bandwidth are removed from the network.

For non-QoS sessions, a bandwidth allocation protocol, also known as flow control, is a crucial part of an ATM network. In this paper, we give such protocols for the two standard forms of non-QoS services. We give the first protocols that have provable convergence and that conform to both the communications standards set by the ATM Forum and the computational limitations of ATM switches.

The ATM Forum has specified two types of non-QoS sessions. In *Available Bit Rate (ABR)* sessions, each session i has a requested bandwidth ρ_i and is allocated a *fair* amount of the available bandwidth of at most ρ_i. In *Available Bit Rate with Minimum Cell Rate (MCR)* sessions, each session i has a minimum required bandwidth MCR_i in addition to the maximum requested bandwidth ρ_i. Each session gets a fair amount of available bandwidth, consistent with respecting all session minima and maxima. Without loss of generality, we assume that each session i comes with an infinite bandwidth request by adding an extra switch at the source with capacity equal to the actual ρ_i.

In this paper, we are concerned with the fair allocation of bandwidth to ABR and MCR sessions at all switches. (Some *admissions control* mechanism is assumed to maintain the feasibility of the MCR sessions.) There are many possible notions of a fair bandwidth allocation in an ATM network. The one adopted by the ATM Forum is *Max-Min fairness*. Informally, an allocation is said to be max-min fair if raising the bandwidth of any session would require that some other session of no greater allocation have its allocation reduced, or have its MCR violated. Formally, we define max-min fairness as follows. Let $\vec{a} = (a_1, \ldots, a_n)$ be an *allocation vector* if session i is allocated bandwidth a_i. An allocation vector $\vec{a}$ is *feasible* if $a_i \geq \mathrm{MCR}_i$ for every session i, and $\sum_{i \in S_j} a_i \leq C_j$ for every switch j where S_j denotes the set of sessions going through j, and C_j denotes its *capacity*. For an allocation vector $\vec{a}$, let $sort(\vec{a})$ be the sorted sequence of the a_i values. An allocation $\vec{a}$ is *max-min fair* if it is feasible, and if its $sort(\vec{a})$ vector is lexicographically greatest amongst all sorted vectors of feasible allocations.

Finding the max-min allocation in a centralized manner is easy. Bertsekas and Gallager [4] presented a simple iterative algorithm to compute the max-min fair vector for ABR sessions. In the first iteration, each switch computes how much bandwidth it can give to each session by dividing its bandwidth by the number of sessions going through it. The switch is found that has the lowest such per session bandwidth, and all sessions through that switch are allocated that bandwidth. This switch is called the *bottleneck switch* for all sessions going through it. All such sessions, along with their allocated bandwidth, are removed from the network, and the procedure is repeated on the residual network. This protocol is easily modified to handle MCR sessions. At each iteration,

all sessions are found such that there is some switch which has allocated it less than its MCR. That session is given its MCR for its allocated bandwidth, and is removed from the network, and the procedure is repeated. The admissions control policy allows us to assume that this procedure maintains feasibility. If at some step all sessions are allocated at least their MCR at every switch, then the switch with the lowest bottleneck level is found, as before, and all sessions through that switch are removed. The procedure then moves on to the next iteration.

This protocol is useful for gaining some understanding of max-min fairness, but is inadequate for implementation. As the name implies, ATM networks are asynchronous. Furthermore, with the exception of expensive procedures such as admissions control, they are distributed. The ATM Forum has specified that, for the purposes of bandwidth allocation, sessions and switches may only communicate through *Resource Management (RM)* cells. An RM cell consists of a small number (2 for ABR, 3 for MCR) of fields which can be updated by the source and switches. One in every 32 packets injected into the network must be an RM cell, so the source sends out new RM cells before receiving old ones. Thus, each session has many RM cells active in a network at once. The transmission of an RM cell by the source, and the updating of these small number of fields by switches is all the communication that can take place in the network.

Our Results. In this paper, we design fast and frugal protocols to achieve exact max-min fairness using only RM cell communication, both for ABR and MCR sessions. A protocol is *frugal* if each update of an RM cell requires $O(1)$ memory per session and a total of $O(1)$ local computation over all sessions. A *fast* protocol converges to the max-min fair allocation in a small number of round trip times of the RM cells, where the round trip time is taken to be the maximum over all session in the network. Let b be the number of distinct bottleneck levels at convergence. Let m be the number of switches in the network and n be the number of sessions. Note that $b \leq m, n$. We provide protocols with the following properties:

Our ABR protocol converges within $O(b)$ roundtrips. This convergence time matches the fastest known convergence for a non-frugal protocol. No frugal protocols were known to converge at all.

Our MCR protocol converges within a number of roundtrips which is $O(nm)$. We give a tighter bound below. While a non-frugal protocol in known with $O(b)$ roundtrip convergence, ours is, to the best of our knowledge, the first frugal protocol known for MCR sessions, and the first frugal protocol which can be shown to converge.

The difficulty in proving the convergence of these protocols is that many of them tend to oscillate, thus destabilizing already converged parts of the network. If some damping is put into the protocols to reduce the oscillations, then it is tricky to show that the protocols make progress. Thus, we achieve a balance between damping oscillations and non-negligible progress in order to prove our results.

History. Many max-min fair protocols have been designed. In [4], the simple centralized algorithm described above was converted into a distributed one, but synchronization of all switches for each iteration was assumed. Other earlier algorithms [10, 12, 8] all required synchronization.

Much effort has been made to design distributed and asynchronous protocols. Mosley [14] and Ramakrishnan, Jain and Chiu [15] were amongst the first to design asynchronous protocols. Unfortunately, Mosely's protocol did not have satisfactory convergence performance. The protocols offered by Charny, Clark and Jain [5, 6] were amongst the very few that have provable convergence in

the distributed and asynchronous setting. The number of round-trip times required for convergence was $O(b)$. However, Charny's protocol was not frugal. Each update required an amount of work linear in the number of sessions crossing the switch. Hou, Tzeng and Panwar [11] generalized Charny's protocol to MCR sessions with linear update complexity and $O(b)$ roundtrip convergence time.

Recently, many distributed and asynchronous protocols were invented, e.g. [7, 13, 16, 17, 18]. In particular, the protocol by Kalampoukas, Varma and Ramakrishnan [13] was frugal. Although their protocol did not have a provable convergence bound, it showed fast convergence in extensive simulations. Other recent work includes [1] which presented a synchronous and convergent protocol for MCR sessions.

Algorithms have been considered for finding an *approximate* max-min allocation for ABR sessions [3]. However, in [2], Afek, Mansour and Ostfeld prove that the max-min fair vector can be sensitive to small changes. For example, if the allocation to one session is changed by δ, then the allocation to some other session may be changed by $\Omega(\delta 2^{n/2})$, where n is the number of sessions in the network. This means an approximated max-min fair solution, e.g. bandwidth allocations restricted to integral components only, can be substantially different from the exact solution.

2 Preliminaries

2.1 Bottlenecks

The concept of *bottleneck* is central to max-min fairness. In order to discuss bottlenecks in more detail, let us first introduce some notation. Let $(a_1, a_2 \ldots, a_n)$ be the max-min fair vector, where a_i is the allocation to session i for $1 \leq i \leq n$. A switch is *saturated* if the total allocation is equal to the switch capacity; a switch is *unsaturated* otherwise.

The *bottleneck level* L_s of an unsaturated switch s is infinity. For a saturated switch s, the bottleneck level is the maximum allocation of the sessions at s that get more than their MCR's, i.e. $L_s = max_{i \in S_s} a_i$ where S_s is the set of sessions that go through s and whose allocations are more than their MCR's. If S_s is empty, then $L_s = 0$. (In the case of ABR sessions, the bottleneck level of a saturated switch s is simply the maximum allocation at s.) A session i is *bottlenecked* at switch s if s has the lowest bottleneck level among all the switches along the path of session i. Switch s is called the *bottleneck switch* of session i.

We now order the switches $s_1, s_2, \ldots, s_m$ by their bottleneck levels, such that $L_{s_k} \leq L_{s_{k+1}}$ for $1 \leq k < m$. For notational simplicity, we denote L_{s_k} by L_k and refer to the set of sessions that are bottlenecked at switch s_k as $\mathcal{S}_k$. We also assume $L_k < L_{k+1}$ for $1 \leq k < m$, since our proofs upper bound the time for all switches with the same bottleneck level to converge, thus we do not really distinguish between them.

2.2 RM Cells

In this paper, an RM cell has the form $\langle A_i, t \rangle$ and for an ABR session i and has the form $\langle A_i, \mathrm{MCR}_i, t \rangle$ for an MCR session i. The first field A_i represents the current requested bandwidth from session i, and the last field represents the current bottleneck switch of session i. During each update, a switch updates A_i and t before passing the RM cell to the next switch.

2.3 Local Labels

Our protocols locally label sessions at each switch. We first focus on ABR sessions. At each switch, a session can be labelled either SAT or UNSAT. These labels are given when the session is updated at the switch, and thus different sessions are given their labels at different times. Informally, a session is labelled SAT if given the current state at the switch the protocol can allocate to the session at least its current requested bandwidth, and is labelled UNSAT otherwise. Labelling is a local event, i.e. a session can be labelled SAT by one switch but UNSAT by another. At convergence each session is labelled UNSAT at its bottleneck switch(es) and SAT elsewhere.

Each switch uses a number of quantities to determine the label of the sessions. The current allocation to session i at s, is A_i^s. The *residual level* is the amount of allocation to the UNSAT sessions if all the SAT sessions fulfill their requests. Formally, the residual level R at switch s is,

$$R = \frac{C - \sum_{i \in S} A_i^s}{|U|},$$

where C is the switch capacity of s, S is the set the SAT sessions at switch s, U is the set of UNSAT sessions at s. If the allocation to some SAT session is higher than R, then this SAT session is getting more than its fair share. It is important to point out that the residual level at a switch can be computed in $O(1)$ time, as long as the switch keeps track of the sum of the SAT allocation and the number of UNSAT sessions.

In the case where sessions have MCR requirements, a session can be labelled SAT, UNSAT or MCR. In this case the residual level at switch s is defined as follows:

$$R = \frac{C - \sum_{i \in S} A_i^s - \sum_{i \in M} \mathrm{MCR}_i}{|U|},$$

where M is the set of MCR sessions at switch s, and the other quantities are defined as before.

Other useful quantities include the maximum allocation to a SAT session at a switch s, i.e. MAXSAT $= \max_{i \in S} A_i^s$, and the minimum allocation to an MCR session at s, i.e. MINMCR $= \min_{i \in M} MCR_i$. However, neither quantity can be updated in $O(1)$ time for every update. Instead, we keep an upperbound for MAXSAT and a lower bound for MINMCR. This is further explained in Section 2.4.

Local labelling is used in some previous work, e.g. [5, 6, 13]. The computation of the residual level also appears in [13].

2.4 Phases and Round-trip Time

In a distributed and asynchronous setting each switch performs local computations whenever it receives an RM cell. Our protocols make use of the notion of *phases* for fast local updates. Each switch *independently* partitions time into phases. A phase at a switch s is the time period during which s receives at least one RM cell from every session that goes through s. It is easy to keep track of the beginning and end of each phase. Our protocols keep a bit vector that associates a bit with each session that goes through the switch. At the beginning of a phase, each bit is set to the same value, say 0. Whenever an RM cell arrives, the corresponding bit is set to its complement, in this case 1. When all bits are set to the same value, which we can detect by keeping a counter which we update when we flip a bit, the current phase ends and a new phase begins.

We now use phases to keep track of MAXSAT and MINMCR, or more precisely an upperbound of MAXSAT and a lowerbound of MINMCR. For example the computation of MAXSAT is performed as follows. During one phase, each switch records the maximum bandwidth that is ever allocated to a SAT session. At the end of the phase we set MAXSAT to this value. During the next phase, MAXSAT is updated whenever a SAT session gets an allocation higher than the current MAXSAT. As we shall see in Sections 3 and 4 our analysis takes this effect into account.

We emphasize again that phases are defined *independently* for different switches. Hence, each switch can operate asynchronously. It is also easy to see that only $O(1)$ computation per update is required to keep track of the phases and the max/min values.

We measuer the convergence time in *round-trip time*, i.e. the maximum time taken by an RM-cell to go from source to destination and back. Note that a phase is no longer than one round-trip time, and typically much shorter.

3 A Convergent ABR Protocol

3.1 Protocol Description

Every switch maintains local SAT and UNSAT labels of sessions as described. We break the protocol into two parts: the *local computation* and the *forwarding action*. The local computation determines how the session will be labelled locally, and other actions such as how much bandwidth it will be allocated locally. The forwarding action determines the content of the RM cell which will be passed along.

When an RM cell of session i passes through the switch we compute the residual level, R, in constant time, according to the current local labels. We use R to determine the new label for session i. The session is labelled SAT if its current request (passed in the RM cell) is smaller than R, and otherwise it is labelled UNSAT.

For the forwarding action sets the new request for the session. To determine that we compute the "bottleneck level". Intuitively, at convergence we want this level to be equal to the residual level. However, because of the distributed way in which the session labels are updated, the residual level may be too low and using that level as a bottleneck may destabilize sessions that are already stable. The residual level can be artificially low when there are sessions labelled SAT at a level higher than their final max-min fair level. Thus, to keep the bottleneck high enough, we set it to $B = \max\{\text{MAXSAT}, R\}$. If B is lower than the current request then the new request is updated to be B and the switch becomes the bottleneck switch for the session.

The bottleneck level B can be higher than the current request for two reasons. The session may have some other bottleneck, and so the session request should not be changed. Otherwise, the current switch is the bottleneck, and B has increased since the session was last updated. In this case, we should increase the request to B. Thus, only the bottleneck switch is allowed to increase the level.

The psuedocode for the local computation is given in Figure 1 and that for the the forwarding action is given in Figure 2.

Local computation: update the label and allocation for session i

1 Compute residual level R, treating i as UNSAT
2 if $R > A_i$
    label i SAT
    set $A_i^s = A_i$
3 if $R \leq A_i$
    label i UNSAT
    set $A_i^s = R$

Figure 1: Local computation at switch s upon receiving $\langle A_i, t \rangle$, a session-i RM cell.

Forward RM cell for session i

4 Compute bottleneck level, $B = \max\{\text{MAXSAT}, R\}$
5 if $s = t$
    pass $\langle B, t \rangle$ { t remains bottleneck }
6 if $s \neq t$
7 if $A_i \leq B$
    pass $\langle A_i, t \rangle$ { t remains bottleneck }
8 if $A_i > B$
    pass $\langle B, s \rangle$ { s becomes new bottleneck }

Figure 2: RM Cell Forwarding Action at switch s upon receiving $\langle A_i, t \rangle$, a session-i RM cell.

3.2 Convergence of the Protocol

We say that session i is *stable* if i gets its max-min fair allocation a_i on every switch on its path and if i is labelled UNSAT at its final bottleneck switch and labelled SAT elsewhere. We also say that a switch s is *stable* if every session that goes through s is stable.

We show that the sessions in $\mathcal{S}_1$ become stable first, and then sessions in $\mathcal{S}_2$, $\mathcal{S}_3$, etc., become stable. In the end all sessions are stable, which means our protocol converges to max-min fairness. Let $\mathcal{S}_0$ be an empty set. We first observe,

Lemma 1 *For $0 \leq k < m$, if every session in $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_k$ remains stable, then the bottleneck level B computed during each update is at least L_{k+1} at switches $s \in \{s_{k+1}, s_{k+2}, \ldots, s_m\}$.*

Proof: By definition, the bottleneck level B is the maximum of the residual level R and the maximum satisfied allocation MAXSAT. If MAXSAT $\geq L_{k+1}$, then we are done. Otherwise, MAXSAT $< L_{k+1}$. Let C_r be the remaining capacity at switch s after satisfying the requests of the sessions in $\mathcal{S}_1, \ldots, \mathcal{S}_k$. For all other sessions that go through s, let x be the number of such sessions. Among these x sessions let y be the number of the SAT ones and Y be the allocation to these y sessions. We have,

$$\frac{C_r}{x} \geq L_{k+1}$$

$$\Rightarrow \quad C_r - Y \geq xL_{k+1} - yL_{k+1}$$
$$\Rightarrow \quad R = \frac{C_r - Y}{x - y} \geq L_{k+1}$$

The first inequality holds since every session in $\mathcal{S}_1, \ldots, \mathcal{S}_k$ remains stable by assumption and the final bottleneck level of s is at least L_{k+1}. The second inequality holds since MAXSAT $< L_{k+1}$. □

Lemma 2 *For $0 \leq k < m$, if every session in $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_k$ remains stable, then after one round-trip time every RM-cell of every session i in $\mathcal{S}_{k+1}, \ldots, \mathcal{S}_m$ has the form $\langle A_i, t \rangle$ for $A_i \geq L_{k+1}$.*

Proof: Consider the first time session i is updated at a switch t, where t is either the current bottleneck of i or t becomes the bottleneck of i due to this update. In either situation, t passes $\langle B, t \rangle$ to the next switch, where B is the bottleneck level. (See line 4 of Figure 2.) By Lemma 1, B is at least L_{k+1}. Now consider every subsequent update of i on any switch s. (Note that $s \in \{s_{k+1}, \ldots, s_m\}$.) For this update, s receives $\langle A_i', t' \rangle$ where $A_i' > L_{k+1}$ and passes on $\langle A_i'', t'' \rangle$ where $A_i'' \geq L_{k+1}$. The latter holds since both A_i and the bottleneck level B are at least L_{k+1}. (See lines 4, 6 and 7 of Figure 2.)

Note that in one round-trip time, session i is certainly updated by such a switch t as described in the above paragraph. Hence, the lemma follows. □

Lemma 3 *For $0 \leq k < m$, if every session in $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_k$ remains stable, then after three round-trip time every session in $\mathcal{S}_{k+1}$ becomes stable and remains stable.*

Proof: We prove by induction on k. Since $\mathcal{S}_0$ is empty, the base case that every session in $\mathcal{S}_0$ remains stable holds trivially. Now we assume that every session in $\mathcal{S}_1, \ldots, \mathcal{S}_k$ remains stable, where $0 \leq k < m$.

We first show that, after three round-trip time, switch s_{k+1} remains the bottleneck of every session in $i \in \mathcal{S}_{k+1}$, i.e. it labels session i UNSAT and allocates L_i to it. By Lemma 2, every RM cell of session $i \in \mathcal{S}_{k+1}$ requests at least L_{k+1} after one round-trip time. Hence, during the second round-trip time s_{k+1} either labels $i \in \mathcal{S}_{k+1}$ UNSAT, or labels i SAT and allocates at least L_{k+1} to i. At the end of the second round-trip time, the residual level at $s_{k+1}$is at most L_{k+1} since sessions in $\mathcal{S}_1, \ldots, \mathcal{S}_k$ remain stable. During the third round-trip time, s_{k+1} labels every $i \in \mathcal{S}_{k+1}$ UNSAT. At the end of the third round-trip time, the residual level at s_{k+1} is L_{k+1}. From now on, s_{k+1} labels $i \in \mathcal{S}_{k+1}$ UNSAT and allocates L_{k+1} to i. As a result, the RM cells of sessions $i \in \mathcal{S}_{k+1}$ remain in the form $\langle L_{k+1}, s_{k+1} \rangle$ for any subsequent update at any switch. This is guaranteed by Lemma 1 and line 6 of Figure 2.

It remains to show that switches $s \in \{s_{k+2}, \ldots, s_m\}$ label $i \in \mathcal{S}_{k+1}$ SAT and allocate L_{k+1} to i. We first show that the residual levels at switches $s_{k+2}, \ldots, s_m$ become higher than L_{k+1} at some point. By Lemma 2, every RM cell of session i in $\mathcal{S}_{k+1}, \ldots, \mathcal{S}_m$ requests at least L_{k+1} after one round-trip time. During the second round-trip time if s labels some session i in $\mathcal{S}_{k+1}, \ldots, \mathcal{S}_m$ SAT then the residual level R must be higher than L_{k+1}. (See line 2 of Figure 1.) Otherwise, s labels every session i in $\mathcal{S}_{k+1}, \ldots, \mathcal{S}_m$ UNSAT, in which case R is higher than L_{k+1} by the end of the second round-trip time.

Finally, we show that the residual levels at switches $s_{k+2}, \ldots, s_m$ remain higher than L_{k+1} for every subsequent update. Let $\langle A_i, t \rangle$ be an RM-cell that arrives at $s \in \{s_{k+2}, \ldots, s_m\}$.

Case 1: i in $\mathcal{S}_1, \ldots, \mathcal{S}_k$. By assumption, session i is stable. Since the residual level is higher than L_{k+1} before this update and $A_i < L_{k+1}$, the residual level is unchanged by this update.

Case 2: i in $\mathcal{S}_{k+1}, \ldots, \mathcal{S}_m$. By Lemma 2, $A_i \geq L_{k+1}$ after one round-trip time. If i is labelled SAT after this update, then the residual level R after this update is higher than A_i and thus higher than L_{k+1}. (See line 2 of Figure 1). If i is labelled UNSAT both before and after this update, then R is not changed by this update. If i is labelled SAT before and UNSAT after this update, we define the following variables for the configuration at switch s right *before* this update. Let A' be the allocation to session i, R' be the residual level, x be the number of sessions labelled UNSAT and Y be the total allocation to the SAT sessions. If C is the switch capacity of s, then,

$$\begin{aligned} & R' = \frac{C - Y}{x} > L_{k+1} \\ \Rightarrow \quad & C - (Y - A') > xL_{k+1} + A' \geq (x+1)L_{k+1} \\ \Rightarrow \quad & R = \frac{C - (Y - A')}{(x+1)} > L_{k+1}. \end{aligned}$$

The second line follows from the fact that $A' \geq L_{k+1}$ due to Lemma 2.

To summarize, we have shown the following. First, after three round-trip time, every session in $\mathcal{S}_{k+1}$ is labelled UNSAT and gets L_{k+1} at switch s_{k+1}, and requests exactly L_{k+1} in its RM cells. Second, after two round-trip time the residual level at switches s_{k+2} and higher remain higher than L_{k+1}. Hence, after three round-trip time every session in $\mathcal{S}_{k+1}$ is labelled SAT and gets L_{k+1} at switches s_{k+2} and higher. Thus, all sessions in $\mathcal{S}_{k+1}$ become and remain stable in three round-trip time. □

We have shown,

Theorem 4 *Our protocol for the ABR sessions requires $O(1)$ local computation per update and converges to the max-min fair vector in $3m$ round-trip times, where m is the number of bottleneck levels.*

4 A Convergent MCR Protocol

4.1 Protocol Description

Each switch performs the protocol independently. As in the ABR case the protocol is composed of two parts. Once part carries the local computation of the SAT, UNSAT and MCR labels at the switch, and the other part determines the new request for the session that is currently being updated.

Both parts of the algorithm use a *threshold* level, τ. Intuitively, the threshold is chosen so that first sessions that should be labelled MCR will gradually receive their correct labels and then it will be equal to the residual level.

The threshold is based on the following refined definition of the residual level, R.

$$R = \frac{C - \sum_{i \in S} A_i^s - \sum_{i \in M} MCR_i}{|U|} \tag{1}$$

Here, C is the switch capacity, S is the set of SAT sessions, M is the set of MCR sessions and U is the set of UNSAT sessions. We also have the following special cases:

$$R = \infty \quad \text{if} \quad U = \emptyset \text{ and } C - \sum_{i \in S} A_i^s - \sum_{i \in M} MCR_i > 0,$$

$$R = 0 \quad \text{if} \quad U = \emptyset \text{ and } C - \sum_{i \in S} A_i^s - \sum_{i \in M} MCR_i \leq 0.$$

The other quantity required to define the threshold is MINMCR. Recall that MINMCR is the minimum allocation of an MCR session which is (computed as described in section 2.4). Also, let MINMCR $= \infty$ if $M = \emptyset$.

Finally, the threshold τ is defined as $\tau = \min\{R, \text{MINMCR}\}$. The threshold is computed at the beginning of each phase and remains unchanged throughout the whole phase.

Local computation: When a session-i RM cell $\langle A_i, MCR_i, t \rangle$ arrives, the new label and allocation of session i are determined by comparing A_i to the threshold τ.

The session is labelled SAT if its current request (passed in the RM cell) is smaller than τ, it is labelled MCR if its MCR value is at least τ, and otherwise it is labelled UNSAT.

The only exception is during a "special phase" which is executed once we end a phase in which $\tau =$ MINMCR and $R > \tau$ (note that R is the new residual level). During the special phase we relabel MCR sessions with MCR value equal to MINMCR as UNSAT and do not make any changes to other labels.

A pseudocode of the protocol is given in Figure 3.

Forwarding action: The value passed along in the RM cell is dependent on whether is switch is currently *steady* or not.

A switch s is said to be *steady* if the threshold τ, and the label and allocation of every session at s, remain unchanged in the previous two phases. The switch keeps track of whether it is steady at the end of a phase.

If the switch is steady, then the bottleneck level B is defined to be $\max\{\tau, \text{MCR}_i\}$; otherwise, $B = \infty$. The forwarding action is then similar to the ABR case.

In our pseudo code of the protocol we have omitted the code that checks to see if the switch is steady or not. We note that this is easily done by keeping track of τ and also the label and allocation for each session. At the end of an update, if any of these quantities change, then the switch is declared unsteady.

A pseudocode of the protocol is given in Figure 4.

4.2 Convergence of the Protocol

A switch s is *stable* if every session that goes through s gets its max-min fair allocation and is labelled correctly at s. We proceed to show that our protocol converges to the max-min fair allocation. The outline of our argument is as follows:

- Assuming $s_1, s_2, \ldots, s_k$ remain stable, for any $j > k$, if switch s_j becomes steady, then its threshold is higher than L_{k+1}.
- In particular, s_{k+1} will become steady at level exactly L_{k+1} and thus will become stable. After this happens, any switch $s_{j'}$ (where $j' > k+1$) can become steady only at a level higher than L_{k+1}. This implies that s_{k+1} will remain stable.

Local computation: update the label and allocation for session i

1 if $A_i < \tau$ { τ is the current threshold }
  label i as SAT
  set $A_i^s = A_i$
2 if $\text{MCR}_i < \tau \leq A_i$
  label i as UNSAT
  set $A_i^s = \tau$
3 if $\tau \leq \text{MCR}_i$
  label i as MCR
  set $A_i^s = \text{MCR}_i$

At the end of a phase

4 Recompute the residual level R.
5 if $\tau = \text{MINMCR}$ and $\tau < R$
  execute a new phase
    label session i UNSAT if $\text{MCR}_i = \text{MINMCR}$
    all other sessions unchanged
6 set $\tau = \min\{R, \text{MINMCR}\}$
7 start a new phase with threshold τ

Figure 3: The local computation at switch s when an RM cell $\langle A_i, \text{MCR}_i, t \rangle$ is received.

In the max-min fair allocation, let Δ denote the amount of bandwidth allocated to UNSAT sessions at a switch, i.e. the numerator of Equation (1), and let n denote the number of UNSAT sessions, i.e. the denominator of Equation (1). Note that Δ/n is equal to the bottleneck level of the switch at convergence.

Lemma 5 *Suppose switches $s_1, s_2, \ldots, s_k$ remain stable. For any $j > k$, if switch s_j becomes steady, then the threshold τ at s_j is at least L_{k+1}.*

Proof: When a switch is steady, it must be the case that $\tau = R \leq \text{MINMCR}$. Otherwise, the session with MCR request equal to MINMCR would have been labelled SAT or UNSAT. For the purpose of contradiction, we assume $\tau < L_{k+1}$ and show that the residual level at the end of the phase is greater than τ. Hence, switch s_j could not have been steady.

We observe that sessions that are MCR in the max-min fair state are correctly labelled, since $\tau < L_{k+1} \leq L_j$. Furthermore, only the following mislabellings are possible at the end of the current phase.

- A session that is UNSAT in the max-min fair state of s_j but is currently labelled SAT. Let n_1 denote the number of such sessions. Each such session has a current allocation at most F_1, where $F_1 < \tau < L_j$.
- A session that is UNSAT in the max-min fair state of s_j but is currently labelled MCR. Let n_2 denote the number of such sessions. Each such session has a current allocation at most F_2, where $F_2 < L_j$.

```
Forward RM cell for session i
8   if switch s is steady
        bottleneck level B = max{τ, MCR_i}
9   if switch s is unsteady
        bottleneck level B = ∞
10  if s = t
        pass ⟨B, MCR_i, t⟩                          { t remains bottleneck }
11  if s ≠ t
        if A_i ≤ B
            pass ⟨A_i, MCR_i, t⟩                    { t remains bottleneck }
        if A_i > B
            pass ⟨B, MCR_i, s⟩                      { s becomes new bottleneck }
```

Figure 4: The forwarding action at switch s when an RM cell $\langle A_i, \text{MCR}_i, t\rangle$ is received.

- A session that is SAT in the max-min fair state of s_j but is currently labelled UNSAT. Let n_3 denote the number of such sessions. Each such session has a max-min allocation at least F_3, where $F_3 \geq \tau$.
- A session that is SAT in the max-min fair state of s_j but is currently labelled MCR.

Recall $\Delta/n = L_j$. From the above observations, we have,

$$\begin{aligned} & \frac{\Delta - n_1 F_1}{n - n_1} > L_j > \tau \\ \Rightarrow \quad & \frac{\Delta - n_1 F_1 - n_2 F_2}{n - n_1 - n_2} > L_j > \tau \\ \Rightarrow \quad & \frac{\Delta - n_1 F_1 - n_2 F_2 + n_3 F_3}{n - n_1 - n_2 + n_3} > \tau \end{aligned}$$

In the left-hand-side of the last expression, if we account for the final SAT sessions that are currently labelled MCR, then its numerator increases. Thus, the residual level R at the end of the current phase is greater than τ. We have thus reached the contradiction. □

Lemma 6 *Suppose switches $s_1, s_2, \ldots, s_k$ remain stable. After one round-trip time, every RM cell $\langle A_i, MCR_i, t\rangle$ of every session $i \in \mathcal{S}_{k+1}$ remain in the form $A_i \geq L_{k+1}$.*

Proof: The proof is almost identical to that of Lemma 2. The only difference is in the definition of the bottleneck level B. Lemma 5 ensures that B is at least L_{k+1}. □

Since a session that is labelled MCR at a switch s_j (where $j < k+1$) can be labelled SAT in the max-min fair state of s_{k+1}, the allocation for a SAT session at s_{k+1} need not correspond to any L_j. Thus, lets use $B_1, B_2, \ldots, B_p$ to denote the distinct allocation levels in the max-min fair state at switch s_{k+1}. We have $L_{k+1} = B_r$, for some $r \leq p$. Also, let q be the number of sessions at switch s_{k+1}.

Lemma 7 *Suppose switches $s_1, s_2, \ldots, s_k$ remain stable. If every RM cell $\langle A_i, \mathrm{MCR}_i, t\rangle$ for every session $i \in \mathcal{S}_{k+1}$ is such that $A_i \geq L_{k+1}$, then in $O(q)$ round-trip time, switch s_{k+1} becomes stable at level L_{k+1}.*

Proof: The proof is by case analysis, and appears in the appendix. □

We now show that s_{k+1} remains stable. From Lemma 5, we see that any switch s_j (where $j > k$) that becomes steady, has threshold at least L_{k+1}. We can modify Lemma 5 easily to show that the threshold at any steady switch s_j, where $j > k+1$, is *greater* than L_{k+1}. Thus, once switch s_{k+1} becomes stable, every session $i \in \mathcal{S}_{k+1}$, will be labelled SAT at s_j, when s_j is steady. Hence s_j will never become the bottleneck switch for these sessions. Thus, once switch s_{k+1} becomes stable, it remains stable.

Theorem 8 *Let $N = \sum_{s_j} |S_j|$, where S_j is the set of sessions that go through switch s_j. Given a network with sessions that have MCR requirements, our protocol converges to the max-min fair allocation in $O(N)$ round-trip time.*

Acknowledgment

The authors wish to acknowledge Lampros Kalampoukas for many helpful discussions.

References

[1] S. Abraham and A. Kumar. A stochastic approximation approach for max-min fair adaptive rate control of ABR sessions with MCRs. In *Proceedings of INFOCOM '98*, pages 1358 – 1365, San Francisco, CA, March 1998.

[2] Y. Afek, Y. Mansour, and Z. Ostfeld. Convergence complexity of optimistic rate based flow control algorithm. In *Proceedings of STOC'96*, pages 89 – 98, Philadelphia, PA, May 1996.

[3] B. Awerbuch and Y. Shavitt. Converging to approximated max-min flow fairness in logarithmic time. In *Proceedings of INFOCOM '98*, pages 1350 – 1357, San Francisco, CA, March 1998.

[4] D. Bertsekas and R. Gallager. *Data Networks.* Prentice Hall, Englewood Cliffs, NJ, 1992.

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

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

[7] R. Jain et al. ERICA switching algorithm: a complete description. ATM Forum Contribution, 96-1172, August 1996.

[8] E. Gafni. *The integration of routing and flow control for voice and data in a computer communication network.* PhD thesis, MIT, 1982.

[9] E. Gafni and D. P. Bertsekas. Dynamic control of session input rates in communication networks. *IEEE Transactions on Automatic Control*, pages 804 – 823, November 1984.

[10] Hayden H. Voice flow control in integrated packet networks. Technical Report LIDS-TH-1152, MIT Laboratory for Information and Decisions Systems, Cambridge, MA, 1981.

[11] Y. T. Hou, H. Y. Tzeng, and S. Panwar. A generalized max-min rate allocation policy and its distributed implementation using the ABR flow control mechanism. In *Proceedings IEEE INFOCOM*, pages 1366 – 1375, San Francisco, CA, March 1998.

[12] J. M. Jaffe. Bottleneck flow control. *IEEE Transactions on Communication*, pages 954 – 962, July 1981.

[13] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan. Dynamic explicit rate allocation algorithm for ABR service in ATM networks. In *Proceedings of the 6th IFIP International Conference in High Performance Networking*, pages 143 – 154, September 1995.

[14] J. Mosley. *Asynchronous distributed flow control algorithms.* PhD thesis, MIT, Cambridge, MA, 1984.

[15] K. K. Ramakrishnan, R. Jain, and D. Chiu. Congestion avoidance in computer networks with a connectionless network layer. Technical Report DEC-TR-510, Digital Equipment Corporation, 1987.

[16] L. Roberts. Enhanced PRCA proportional rate control algorithm. ATM Forum Contribution, 94-0735R1, August 1994.

[17] K. Y. Siu and H. Y. Tzeng. Intelligent congestion control for ABR service in ATM networks. *ATM SIGCOMM Computer Communication Review*, 24(5):81 – 106, October 1994.

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

A Proof of Lemma 7

Proof: The proof is by case analysis. Case 1 is when the current threshold is above L_{k+1}; in this case, we show that, in at most $(p-r)$ round-trip time, the threshold will become at most L_{k+1}. Case 2 is when the current threshold is at most L_{k+1}; in this case, we show that, in at most $O(q)$ round-trip time, the threshold will become L_{k+1} and then s_{k+1} will become stable.

Case 1: $\tau > L_{k+1}$.

In this case note that, at the end of the current phase, every session, whose max-min share is below L_{k+1} or above τ, is correctly labelled and gets its max-min share.

Let MAXMCR denote the largest MCR value that is less than τ at the end of current phase. There are two subcases to handle depending on whether MAXMCR is above or below L_{k+1}.

(i) MAXMCR $> L_{k+1}$: in this case we claim that the threshold τ for the next phase is less than MAXMCR. The proof of this claim follows.

The only sessions that can be mislabelled at the end of this phase are those sessions that are UNSAT in the max-min fair state but are currently labelled SAT and also those sessions that are MCR in the max-min state but are currently labelled SAT or UNSAT.

Let n_1 denote the number of final UNSAT sessions that are currently labelled SAT; each of these sessions has current allocation at least $F_1 \geq L_{k+1}$. Let n_2 denote the number of final MCR sessions that are currently labelled UNSAT; each of these sessions has a max-min share $F_2 \leq$ MAXMCR.

Recall that in the max-min state $L_{k+1} = \frac{\Delta}{n}$. Since $F_1 \geq L_{k+1}$,

$$\frac{\Delta - n_1 F_1}{n - n_1} \leq L_{k+1} < \text{MAXMCR}$$

Now, $F_2 \leq$ MAXMCR implies that

$$\frac{\Delta - n_1 F_1 + n_2 F_2}{n - n_1 + n_2} < \text{MAXMCR}$$

If we take into account those final MCR sessions that are currently labelled SAT then the numerator of the above expression decreases. Thus the residual level R at the end of the current phase is less than MAXMCR. Thus the new τ is less than MAXMCR.

Since there are $p - r$ MCR levels above L_{k+1}, we note that the above subcase holds true for at most $p - r$ round-trip time after which the following happens.

(ii) MAXMCR $\leq L_{k+1}$: in this case we claim that the threshold for the next phase is less than L_{k+1}. The proof of this claim follows.

Since every session with MCR level below τ is labelled either SAT or UNSAT, and since MAXMCR $\leq L_{k+1}$, it follows that every final MCR session is correctly labelled. Thus the only sessions that can be mislabelled are those sessions that are labelled UNSAT in the max-min fair state but are currently labelled SAT; these sessions currently get allocation at least L_{k+1}.

In the max-min state $L_{k+1} = \frac{\Delta}{n}$. Let n_1 be the number of sessions that are currently incorrectly labelled SAT. Each of these sessions get allocation at least F_1 (where $F_1 \geq L_{k+1}$). Then the residual level at the end of the current phase is

$$R \leq \frac{\Delta - n_1 F_1}{n - n_1} \leq L_{k+1}$$

Thus the new threshold τ is at most L_{k+1}.

Hence, if we are in Case 1, then, in at most $p - r$ round-trip time, the threshold τ will become at most L_{k+1}. We note that in Case 1, if # UNSAT $= 0$, then $C - \sum_{i \in S} A_i^s - \sum_{i \in M} MCR_i \leq 0$, and thus $R = 0 < L_{k+1}$; thus, in this case, the threshold is set to 0 for the next phase.

Case 2: $\tau \leq L_{k+1}$.

Suppose $\tau < L_{k+1}$. Let B_l (for some l) denote the highest max-min fair allocation at s_{k+} that is less than τ. We will show that, at the end of the *next* phase, the threshold will be greater than min$\{B_{l+1}$, MINMCR at end of next phase$\}$.

At the end of the current phase, all final MCR sessions are correctly labelled. In addition, all final SAT sessions with max-min shares below τ are correctly labelled with allocations equal to their max-min share. The only possible mislabellings involve final UNSAT sessions that are currently labelled MCR and final SAT sessions that are currently UNSAT or MCR. Let n_1 denote the number of final UNSAT sessions that are currently mislabelled as MCR; each such session has allocation at least F_1 (where current MINMCR $\leq F_1 < L_{k+1}$). Let n_2 denote the number of final SAT sessions that are currently mislabelled as UNSAT; each such session has a max-min allocation at least $F_2 \geq B_{l+1}$.

In the max-min fair state $\frac{\Delta}{n} = L_{k+1}$.

Suppose MINMCR $< L_{k+1}$. Then we have

$$\frac{\Delta - n_1 F_1}{n - n_1} \geq L_{k+1} > \text{MINMCR}$$

Now, if MINMCR $\leq B_{l+1}$, then

$$\frac{\Delta - n_1 F_1 + n_2 F_2}{n - n_1 + n_2} > \text{MINMCR}$$

However, if MINMCR $> B_{l+1}$, then

$$\frac{\Delta - n_1 F_1 + n_2 F_2}{n - n_1 + n_2} > B_{l+1}$$

In all of the above computations, if we take into account final SAT sessions that are currently labelled MCR, then the numerator will increase.

Suppose MINMCR $\geq L_{k+1}$; then $n_1 = 0$. Assuming $B_{l+1} < L_{k+1}$, we have

$$\frac{\Delta + n_2 F_2}{n + n_2} > B_{l+1} \text{ and } \frac{\Delta + n_2 F_2}{n + n_2} \leq L_{k+1}$$

Thus, the residual level R is greater than $min\{B_{l+1}, \text{MINMCR}\}$.

The above arguments also show that if the current τ = MINMCR, then at the end of the phase, the protocol will execute another phase in which all sessions with the MCR value equal to τ are labelled UNSAT. The residual level R after this phase can also be seen to be greater than $min\{B_{l+1}, \text{MINMCR at the end of next phase}\}$ by applying the above arguments.

From the above arguments, we also note that R remains at most L_{k+1}. The terminating scenario is when the threshold becomes equal to the $max\{\text{MCR}_{j'}, B_{r-1}\}$, where $\text{MCR}_{j'}$ is the largest MCR level below L_{k+1}. Then, using arguments similar to above, we can show that the threshold τ becomes equal to L_{k+1} in at most two additional round-trip time. Once this happens, all sessions will be correctly labelled and will get their max-min share. Thus, s_{k+1} becomes steady at L_{k+1} and hence stable.

Thus, if we are in Case 2, then, in at most $O(q)$ round-trip time, the threshold will become L_{k+1}. In the phase that follows, every session that passes through this switch is correctly labelled and also gets its max-min share; thus s_{k+1} becomes stable. □