Fair Distributed Congestion Control in Multirate Multicast Networks*

Saswati Sarkar*and Leandros Tassiulas†

Abstract

We study fairness of resource allocation in multirate, multicast networks. In multirate networks, different receivers of the same multicast session can receive service at different rates. We develop a mathematical framework to model the maxmin fair allocation of bandwidth with minimum and maximum rate constraints. We present a necessary and sufficient condition for a rate allocation to be maxmin fair in a multirate, multicast network. We propose a distributed algorithm for computing the maxmin fair rates allocated to various source-destination pairs. This algorithm has a low message exchange overhead, and is guaranteed to converge to the maxmin fair rates in finite time.

1 Introduction

Multicasting provides an efficient way of transmitting information from a sender to a set of receivers. A single source node or a collection of source nodes send identical messages simultaneously to multiple destination nodes. Single destination or unicast and broadcast to the entire network are special cases of multicast. Multicasting reduces bandwidth consumption as message replication takes place at only the forking nodes. This is particularly useful for real time multi-party communications like audio or video teleconferencing, video-on-demand services, distance learning etc., as these applications consume a lot of bandwidth. We would study resource allocation for real time multicast applications.

There are two possible transmission modes in multicast networks for loss tolerant real time traffic like audio, video, etc. In one, all receivers in the same session receive information at the same rate. However, this unirate transmission has severe shortcomings for multicast networks. This is because of network heterogeneity. A single session may have many destinations. The paths to different destinations may have different bandwidth capacities, e.g., one may consist

*S. Sarkar is with the department of Electrical and Systems Engineering in University of Pennsylvania, email: swati@ee.upenn.edu, Phone: 215 573 9071. Her work was supported in part by NSF grants ANI01-06984 and NCR02-38340.

†L. Tassiulas is with the Computer Engineering and Telecommunications department in the University of Thessaly, Volos, Greece, and Department of Electrical and Computer Engineering and Institute for Systems Research in the University of Maryland, College Park. Email: leandros@inf.uth.gr

*Parts of this paper have been presented at GLOBECOM' 1999 and INFOCOM' 2000.

of multi-megabit links, such as, $T3$ (45 Mbps) and another may have a 128 kbps ISDN line. A single rate of transmission per session is likely to either overwhelm the slow receivers or starve the fast ones, in absence of additional provisions. For real time traffic, multirate transmission can be used to counter network heterogeneity. The receivers of the same session are allowed to receive at different service rates. We would discuss several multirate encoding schemes later, but we mention one of the possibilities now. A source encodes its signal in several layers, and these layers can be combined at the decoder for signal reconstruction. These layers are transmitted as separate multicast groups, and receivers adapt to congestion by joining and leaving these groups [8]. Precision of the reconstruction improves with the number of layers received. This layered transmission scheme has been used for both video [21] and audio [4] transmissions over the internet and has potentials for use in ATM networks as well [11].

Layer bandwidth may be flexible or predetermined. In the former, layer bandwidth can be tuned to closely match any desired receiver rates with fine granularity [15]. The feasible set can be assumed to be continuous in this case. In the latter, layer bandwidths are predetermined and have coarse granularity. A receiver either receives a layer fully or does not receive the layer, at all. It can not partially subscribe to a layer. Effectively, the network can only allocate a discrete set of rates to the receivers, whereas a continuous set of rates can be allocated when receivers can subscribe to fractional layers. We study the continuous case here and have studied the discrete case in [20]. We discuss how to attain a continuous allocation of rates in Section 4.

We study fair allocation of rates in multirate, multicast networks. Max-min fairness is a well accepted definition of fairness [3]. A bandwidth allocation is *maxmin fair*, if no receiver can be allocated a higher service rate without lowering that of another receiver having equal or lower rate. The objective is to serve every receiver of every session at a fair rate. The service rate of a receiver should depend only on the congestion in the path leading to the receiver, and its processing capability. Attaining this objective is complicated in multicast networks as different receivers have different paths and processing capabilities.

We first review the previous work in this area. We have proposed a routing and scheduling policy which stabilizes the system, if the network can accommodate all the traffic demands [18]. However, resource limitations may not allow fulfilling all traffic demands, particularly for bandwidth expensive real time applications. Fairness of resource allocation becomes important in such a scenario. Tzeng *et al.* have investigated the problem of fair allocation of bandwidth to multicast sessions under the constraint that all receivers of the same session must receive service at the same rate [22]. However, under this unirate transmission, due to network heterogeneity, service rate may not match the path bandwidth and the processing capability of every receiver. Fairness properties of a multicast network improves if multi-rate transmission is used instead of single rate transmission [17]. Chiung *et al.* [6] advocate simulcast, but that is bandwidth inefficient. Fair allocation of layers (rates) in presence of layered transmission (transcoding) provisions has not been cultivated very systematically, until recently [17]. There are two well known network protocols for layered transmission, RLM (Receiver-driven Layered Multicast) [16] and LVMR (Layered Video Multicast with Retransmissions) [12]. These approaches improve fairness among members of the same session. However, neither handles fairness among members of different sessions very well, when there are multiple sessions competing for bandwidth [13]. Li. *et al.* [13] suggest a scheme for fair allocation of layers for multi-session layered video multicast to rectify this defect in RLM and LVMR. The authors present empirical evidence that the scheme can share bandwidth fairly with TCP and improves inter-session fairness for networks with multiple video sessions sharing only one link. Rubenstein *et al.* [17] propose a centralized algorithm for computing the maxmin fair rates in a multirate multicast network. Centralized algorithms can not be implemented in large networks.

In Section 2, we formulate the problem of fair allocation of bandwidth in multirate multicast networks. In Section 3.1, we present an algorithm for computing the maxmin fair rates in an arbitrary network with any number of multicast sessions. This algorithm requires only local information at any node in the network and is thus amenable to distributed computation. In Section 3.2, we present a framework for scalable distributed implementations of the above algorithm. Our analytical results guarantee that the distributed algorithm converges to the maxmin fair rates in finite time, and presents worst case convergence time bounds. Our algorithm can be used in internet and ATM networks. We have incorporated minimum rate constraints keeping in mind ATM networks. In Section 3.3, we investigate using simulations the performance of the distributed algorithm. In Section 4, we discuss several features of the fairness framework. We conclude in Section 5. We have summarized the notations used throughout the paper in Symbol Table 1. Unless otherwise stated, the proofs can be found in the appendix.

2 Network Model

Symbol	Meaning
N	Number of sessions.
M	Number of virtual sessions.
$\chi(j)$	Session of virtual session j.
$n(l)$	Set of sessions passing through link l.
$m(k,l)$	Set of virtual sessions of session k passing through link l.
C_l	Capacity of link l.
r_j	Bandwidth allocated to virtual session j.
λ_{il}	Bandwidth allocated to session i in link l, $\lambda_{il} = \max_{j \in m(i,l)} r_j$.
μ_i	Minimum bandwidth requirement of virtual session i.
μ_{il}	Minimum bandwidth required by session i in link l, $\mu_{il} = \max_{j \in m(i,l)} \mu_j$.

Table 1: Summary of symbols used throughout the paper

We consider an arbitrary topology network with N multicast sessions. A multicast session is identified by a number n, and is associated with a source and destination set pair (v, U), where v is the source node of the session and U is the set of destination nodes. There is a tree associated with each session that carries the traffic of the session. The tree can be established during connection establishment phase in connection-oriented networks, or can be established

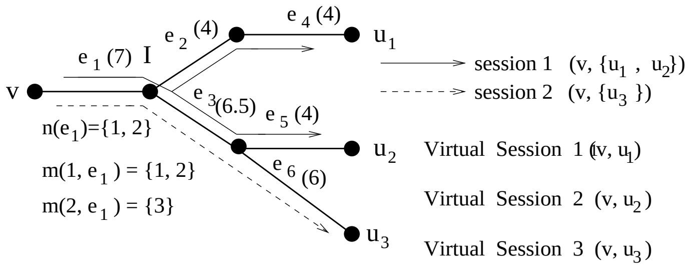

Figure 1: The numbers in brackets () denote the capacities of the respective links. The capacity constraint for link e_1 is $\max(r_1, r_2) + r_3 \leq 7$ and that for link e_3 is $r_2 + r_3 \leq 6.5$. Thus minimum and the maximum rate constraints are $4 \leq r_1 \leq 5$, $1 \leq r_2 \leq \infty$ and $0 \leq r_3 \leq 5$. The maxmin fair rate vector is $(4, 3.5, 3)$. Under the maxmin fair rate allocation, link e_1 is bottlenecked w.r.t. virtual sessions (v, u_1) and (v, u_3) and e_3 is bottlenecked w.r.t. virtual session (v, u_2). Link e_6 is not bottlenecked w.r.t. any virtual session because its capacity is not fully utilized. Consider another feasible rate vector $(4, 4.5, 2)$. Now no link is bottlenecked w.r.t. virtual session (v, u_3).

by some well known multicast routing protocol like DVMRP [7], MOSPF [14], CBT [2] and PIM [8] in connectionless networks.

We call every source destination pair of a session a virtual session. If a session n has source v and destination set U, where $U = \{u_1, \ldots, u_t\}$, then it corresponds to t virtual sessions, (v, u_1), $\ldots, (v, u_t)$. For example, in figure 1, session 1 has two receivers, u_1, u_2, $U = \{u_1, u_2\}$ and 2 virtual sessions, (v, u_1) and (v, u_2). Our objective is to achieve the maxmin fair rate allocation for the virtual sessions. Every virtual session (source-destination pair) has a minimum and a maximum rate. These are decided from application requirements. For example, the reception quality may be poor for a high fidelity video transmission if the bandwidth is below a certain threshold, which constitutes the minimum rate for the receiver.

Let there be M virtual sessions in the network. Rate allocation is an M-dimensional vector $(r_{11}, \ldots, r_{1m_1}, \ldots, r_{i1}, \ldots, r_{im_i}, \ldots, r_{N1}, \ldots, r_{Nm_N})$, with r_{ij} being the rate allocated to the jth virtual session of the ith session. For simplicity, henceforth, we will use a single index. A rate allocation $(r_1, \ldots, r_M)$ is feasible if the following conditions hold.

1. $\mu_i \leq r_i \leq p_i \; \forall i$, where μ_i and p_i are respectively the minimum and the maximum rates of virtual session i, $p_i \geq \mu_i \geq 0$,

2. Let $n(l)$ denote the set of sessions passing through link l, $m(k, l)$ denote the set of virtual sessions of session k passing through link l and C_l denote the capacity of link l. The rate allocated to session i in link l, λ_{il}, under rate allocation $\vec{r}$ is the maximum of the rates allocated to the virtual sessions in $m(i, l)$, i.e., $\lambda_{il} = \max_{j \in m(i,l)} r_j$. Then the total bandwidth consumed by all sessions traversing link l can not exceed l's capacity, i.e., $\sum_{i \in n(l)} \lambda_{il} \leq C_l$. In other words,

$$\sum_{i \in n(l)} \max_{j \in m(i,l)} r_j \leq C_l \qquad \text{(capacity condition).}$$

Figure 1 illustrates an example network with a few capacity and maximum and minimum rate constraints. We assume here that the minimum rate requirements of all sessions can be satisfied. If not, then the network can either not accept the session or lower the minimum rate guarantee.

A feasible rate allocation vector $\vec{r}^1$ is maxmin fair if it satisfies the following property with respect to any other feasible rate allocation vector $\vec{r}^2$: if there exists i such that the ith component of $\vec{r}^2$ is strictly greater than that of $\vec{r}^1$ ($r_i^2 > r_i^1$), then there exists j such that the jth component of $\vec{r}^1$, r_j^1 is less than or equal to the ith component of $\vec{r}^1$, r_i^1 ($r_j^1 \leq r_i^1$) and the jth component of $\vec{r}^2$ (r_j^2) is strictly less than the jth component of $\vec{r}^1$ ($r_j^2 < r_j^1$). The bandwidth allocations according to $\vec{r}^2$ are less even than those according to $\vec{r}^1$ in some sense. Refer to figure 1 for an example of a maxmin fair allocation.

Lemma 1 *Maxmin fair allocation exists uniquely.*

Proof of Lemma 1: We will present an algorithm which attains the maxmin fair allocation in M iterations (Theorem 1). This proves the existence result.

Suppose two different rate allocations $\vec{A}$ and $\vec{B}$ are both maxmin fair. There exists a component i_1 such that $a_{i_1} \neq b_{i_1}$. Without loss of generality $a_{i_1} > b_{i_1}$. Since $\vec{B}$ is maxmin fair, there exists i_2 such that $b_{i_2} < b_{i_1}$ and $a_{i_2} < b_{i_2}$. Thus $a_{i_2} < a_{i_1}$. Since $a_{i_2} < b_{i_2}$, from the maxmin fairness of $\vec{A}$, there exists i_3 such that $a_{i_3} < a_{i_2}$ and $b_{i_3} < a_{i_3}$. Thus $b_{i_3} < b_{i_2}$. Continuing similar arguments, we get an infinite sequence $i_1, i_2, \ldots i_p, \ldots$ such that $a_{i_k} < a_{i_{k-1}}$, $b_{i_k} < b_{i_{k-1}}$ and $a_{i_k} \neq b_{i_k}$. It follows that the elements in this sequence are all distinct. This contradicts the fact that the number of components is finite. □

Henceforth, we shall ignore the maximum rate constraints. This does not cause any loss of generality because maximum rate constraints can be incorporated by adding artificial links between receivers and the rest of the network. The capacity of an artificial link equals the maximum rate of the receiver. The size of the augmented network and hence the computational complexity for the fair rates is similar to that of the given network.

Next, we introduce the concept of bottleneck links. A link l is said to be *bottlenecked* with respect to a virtual session k traversing l if the following conditions hold.

- Capacity of link l is fully utilized, i.e., the sum of the rates allocated to the sessions traveling the link equals the capacity of the link: $\sum_{i \in n(l)} \lambda_{il} = C_l$.
- The virtual session k has the maximum rate amongst all virtual sessions of the same session traveling the link, i.e., $r_k = \lambda_{\chi(k)l}$, where $\chi(k)$ is the session of the virtual session k.
- If any other virtual session j traversing link l has a rate higher than that of virtual session k, then j's rate is less than or equal to the minimum rate of some virtual session in $m(\chi(j), l)$. Thus, if $r_j > r_k$ then $r_j \leq \mu_{\chi(j)l}$.

Refer to figure 1 for an example of bottleneck links.

We now compare the definition of a bottleneck link with that in the unicast context. First assume that the minimum rate constraints do not exist. In a unicast network, a link is bottlenecked w.r.t. a session if its capacity is fully utilized and no other session traversing the link has a greater rate. In a multirate multicast network, the definition for a bottleneck link can be obtained by replacing session with virtual session in the above definition. Now assume that the minimum rate constraints exist. In the unicast context, let l be the bottleneck link of session i. Then the capacity of link l is fully utilized like in the multicast case. However, a session j traversing l can have greater bandwidth than i, but then j's rate must equal j's minimum required rate. In the multicast case, let l be the bottleneck link of virtual session i. Now the difference with the unicast case is that a virtual session j traversing link l can has greater bandwidth than i, and j's rate can be greater than j's minimum required rate. The constraint in this case is that j's rate must be less than the minimum rate requirement of its session in link l, $(\mu_{\chi(j)l})$, and virtual sessions i and j must belong to different sessions. Consider figure 1 for an example. Let the minimum rate requirements be $(0, 4, 0)$. Consider an allocation $(4, 4, 2.5)$. Link e_1 is bottlenecked with respect to virtual session 3. Note that $r_1 > \max(\mu_1, r_3) = 2.5$. However, virtual sessions 1 and 3 are in different sessions, and $r_1 = \mu_{1e_1} = 4$.

Lemma 2 (Bottleneck Lemma) *A feasible rate vector is maxmin fair iff every virtual session has a bottleneck link.*

Remark: Bottleneck Lemma serves as a test for maxmin fairness of a feasible rate allocation vector. It indicates that if a rate vector is maxmin fair, then the rate of a virtual session s is at least $C_l/|n(l)|$ for some link l in its path if there are no minimum rate requirements. In presence of minimum rate requirements, this lower bound becomes $\frac{C_l - \sum_{i \in \tau(l)} \mu_{il}}{|n(l) \setminus \tau(l)|}$, where $\tau(l)$ is the set of sessions traversing link l whose session rates in link l are greater than s's rate.

3 A distributed scheme for computation of the maxmin fair rates

We now present a distributed scheme for computing the maxmin fair rates. We first describe the basic algorithm and then discuss its distributed implementation.

3.1 Basic Algorithm

We describe the basic algorithm here. We first present a definition. In Table 2, we have summarized other notations used in this algorithm.

A virtual session s is *saturated* under rate vector $\vec{r}(k)$ if there exists a link l in its path such that l's capacity is fully utilized and s has the maximum rate amongst all virtual sessions traversing l, i.e.,

$$\sum_{i \in n(l)} \lambda_{il}(k) = C_l \text{ and } r_s(k) = \lambda_{\chi(s)l}(k).$$

A session is *saturated* in a link l if all the virtual sessions of the session traveling link l are saturated.

Now, we present the algorithm.

Symbol	Meaning
L_s	Set of links traversed by virtual session s.
$\mathcal{L}$	Set of links in the network.
$r_s(k)$	Bandwidth allocated to virtual session s at the end of the kth iteration.
$\vec{r}(k)$	Rate vector at the end of the kth iteration, with components $r_s(k)$.
$\lambda_{il}(k)$	Bandwidth allocated to session i in link l $\lambda_{il}(k) = \max_{j \in m(i,l)} r_j(k)$.
$S(k)$	Set of unsaturated virtual sessions at the end of the kth iteration.
$\Theta_l(k)$	Set of unsaturated sessions passing through link l at the end of the kth iteration.
$F_l(k)$	Total bandwidth consumed by the saturated sessions passing through link l at the end of the kth iteration.
$\eta_l(k)$	Link control parameter of link l at the end of the kth iteration.
$\eta_{il}(k)$	Session link parameter of session i in link l at the end of the kth iteration, $\eta_{il}(k) = \max(\eta_l(k), \mu_{il})$.

Table 2: Summary of symbols used in the rate computation algorithm

1. $k = 0$ $\eta_l(0) = 0$, $F_l(0) = 0$, $S(0) = \{1, \ldots, M\}$, $\Theta_l(0) = n(l)$ $\forall$ link l, $r_s(0) = \mu_s$ $\forall$ virtual session s.

2. $k \to k + 1$

3. For every link l in the network, compute the link control parameter. If $\Theta_l(k-1) \neq \phi$, then $\eta_l(k)$ is the maximum possible θ, which satisfies the equation, $F_l(k-1) + \sum_{i \in \Theta_l(k-1)} \max(\theta, \mu_{il}) = C_l$, else $\eta_l(k) = \eta_l(k-1)$. Here, $\eta_{il}(k) = \max(\eta_l(k), \mu_{il})$.*

4. Compute $r_s(k)$ for all virtual sessions s, where $r_s(k) = \min_{l \in L_s} \eta_{\chi(s)l}(k)$, if $s \in S(k-1)$, else $r_s(k) = r_s(k-1)$.

5. For every link l in the network compute the session rate in link l, for every session in $n(l)$, $\lambda_{il}(k) = \max_{s \in m(i,l)} r_s(k)$.

6. Compute the set of virtual sessions unsaturated after the kth iteration, $S(k) = S(k-1) \setminus \{s : \exists\, l \in L_s,$ s.t. $\sum_{i \in n(l)} \lambda_{il}(k) = C_l$ and $r_s(k) = \lambda_{\chi(s),l}(k)\}$.

7. If $S(k) = \phi$, i.e., all virtual sessions are saturated, the algorithm terminates, else go to next step.

8. For every link l, compute the set of unsaturated sessions passing through link l at the end of the kth iteration: $\Theta_l(k) = \{n : n \in \{1, \ldots, N\}, m(n,l) \cap S(k) \neq \phi\}$.

9. For every link l, for which $\Theta_l(k) \neq \phi$, compute the bandwidth consumed by the saturated sessions passing through link l, $F_l(k) = \sum_{i \in n(l) \setminus \Theta_l(k)} \lambda_{il}(k)$.

*This computation need to be done for all unsaturated sessions traversing link l only.

10. Go to step (2).

We now describe the algorithm's operation. The maxmin fair bandwidth is computed via an iterative procedure. The algorithm classifies every virtual session as either saturated or unsaturated. Initially, all virtual sessions are unsaturated, and their status change from unsaturation to saturation. Ignore the minimum rate constraints for the initial intuition. At the beginning of an iteration k, every link l computes a "fair share" for every virtual session traversing l (" link control parameter" $\eta_l(k)$) as per step (3). This fair share is assigned to every virtual session traversing l if there are no bandwidth constraints on other links. The bandwidth restrictions of other links are considered as follows. A virtual session s is allocated a rate equal to the minimum of the link control parameters on its path. In presence of minimum rate requirements, a link additionally computes the session link parameter, $\eta_{il}(k)$ for every session i traversing the link, as $\eta_{il}(k) = \max(\eta_l(k), \mu_{il})$. Thereafter, a virtual session is allocated a rate equal to the minimum of its session link parameters in its path. Now, $\vec{r}(k)$ is the bandwidth allocation in iteration k. The algorithm subsequently checks the saturation condition for each unsaturated virtual session. It turns out that when a virtual session is saturated its bandwidth is maxmin fair, and its bandwidth allocation does not change subsequently. The algorithm terminates if all the virtual sessions are saturated, otherwise there is at least one more iteration. In the latter case, the algorithm makes computations which are used in the next iteration. We describe them now. The bandwidth consumed by the saturated sessions in a link, if any, are computed. This bandwidth is subtracted from the link capacity, and the link control parameters are recomputed in the next iteration using this residual capacity. We illustrate the operation of the algorithm with an example.

Example 3.1.1: Consider the network of figure 1. The maximum rate constraints do not exist. Link control parameters are as follows: $\eta_{e_1}(1) = 3$, $\eta_{e_2}(1) = 4$, $\eta_{e_3}(1) = 3.25$, $\eta_{e_4}(1) = 4$, $\eta_{e_5}(1) = 4$, $\eta_{e_6}(1) = 6$. Now, the session link control parameters are as follows. $\eta_{1e_1}(1) = 4$, $\eta_{2e_1}(1) = 3$, $\eta_{1e_2}(1) = 4$, $\eta_{1e_3}(1) = 3.25$, $\eta_{2e_3}(1) = 3.25$, $\eta_{1e_4}(1) = 4$, $\eta_{1e_5}(1) = 4$, $\eta_{2e_6}(1) = 6$. Computing the $r_s(1)$s as per step 4, we have $r_1(1) = 4$, $r_2(1) = 3.25$, $r_3(1) = 3$. Observe that virtual sessions 1 and 3 are saturated, while virtual session 2 is not. $S(1) = \{2\}$. Thus session 2 is saturated on all links. Session 1 is unsaturated on only those links which are on the path of virtual session 2. $\Theta_{e_1}(1) = \Theta_{e_3}(1) = \Theta_{e_5}(1) = \{1\}$, $\Theta_l(1) = \phi$, if $l \notin \{e_1, e_3, e_5\}$. $F_{e_1}(1) = F_{e_3}(1) = 3$, and $F_{e_5}(1) = 0$. Computations for the next iteration are as follows: $\eta_{e_1}(2) = 4$, $\eta_{e_3}(2) = 3.5$, $\eta_{e_5}(2) = 4$, and $\eta_l(2) = \eta_l(1)$ for the rest of the links. Now, $\eta_{1e_1}(2) = 4$, $\eta_{1e_3}(2) = 3.5$, $\eta_{1e_5}(2) = 4$. Thus $r_2(2) = 3.5$. $r_s(2) = r_s(1)$, $s \in \{1, 3\}$. Now virtual session 2 is also saturated. So $S(2) = \phi$ and the algorithm terminates. The rates obtained upon termination are $(4, 3.5, 3)$.

Theorem 1 *The algorithm yields a maxmin fair rate allocation in at most M iterations, where M is the number of virtual sessions.*

The intuition behind the result is as follows: maxmin fair sharing implies that if there are k sessions sharing a link, each session should get a "fair share" of the link bandwidth. If a session is constrained to have a rate less than its fair share because it is assigned a lower bandwidth on another link, then the residual bandwidth is split fairly among other sessions. This is exactly what the algorithm does.

Let $\mathcal{L}$ be the set of links. Every step of this algorithm has a complexity of $O(|\mathcal{L}|M)$. Since the algorithm must terminate in M iterations, the overall complexity of this algorithm is $O(|\mathcal{L}|M^2)$.

The algorithm terminates in at most $|\mathcal{L}|$ iterations in a special case, i.e., if all the virtual sessions of the same session sharing a link have the same minimum rate requirement ($\mu_i = \mu_j$ if $\chi(i) = \chi(j)$ and $L_i \cap L_j \neq \phi$) (Lemma 3). This condition on minimum rates always holds in unicast networks because every session has only one virtual session, and in multicast networks without any minimum rate requirements.

Lemma 3 *The algorithm terminates in at most* $\min(|\mathcal{L}|, M)$ *iterations, if* $\mu_i = \mu_j$ *for all* i, j *such that* $\chi(i) = \chi(j)$ *and* $L_i \cap L_j \neq \phi$.

3.2 Distributed Implementation of the Basic Algorithm

We outline the distributed implementation of the basic algorithm presented in the previous subsection. The details can be found in technical report [19]. We will exploit the facts that (a) the computation of the session link parameters of sessions in any link needs information only about the saturation status of the sessions traversing the link and the previous iteration rates of the unsaturated sessions traversing the link, and (b) a virtual session (receiver) can determine its rate and saturation status if it knows the session link rates and the bandwidth utilizations in the links on its path.

Every node maintains an information record for each of its outgoing links l. The record maintains the following entries for every session i traversing link l : (a) minimum session link rate: μ_{il}, (b) session link rate, (c) rate bit and (d) saturation entry. Every link also stores its link control parameter. Note that the storage in a record does not maintain separate information about the virtual sessions of the session. Nodes interchange control messages during the distributed computation. The control messages are (a) backward rate packets (b) forward rate packets (c) probe packets (d) saturation/unsaturation messages. The first two are used to update the session link rates at the intermediate nodes and the receiver rates at the receivers. The last two are used to update the saturation status of the sessions and the receivers at the intermediate nodes and the receivers respectively.

Initially, all the sessions and virtual sessions are unsaturated. Receivers send backward rate packets towards the respective source with large rate values. Links compute their link control parameters. The intermediate nodes modify the rate values in the backward rate packets as they travel towards the source. The rate value in a backward rate packet is decreased to the minimum of the link control parameter and the current value in the rate packet. The nodes record these modified rate values as the session link rates. After this modification, nodes merge the backward rate packets of a session. The merged backward rate packet is transmitted towards the source with a rate value that equals the maximum of those in the individual backward rate packets.

A session source generates a forward rate packet after it receives a backward rate packet, with the same rate value as in the incoming backward rate packet, and transmits it downstream. Once a forward rate packet reaches a node, the node updates the session link rates in its outgoing links i.e., sets them equal to the rate value in the incoming forward rate packet. This modified value is the current iteration session link rate of the basic algorithm. The node then multicasts the forward rate packet in each of its outgoing links on the path of the session, with a rate value equal to that in the modified session link rate field of the session. When a receiver receives a forward rate packet, it records its rate and sends a probe packet towards the source to query its saturation status.

An intermediate node determines the saturation status of the session in the link when it receives a probe packet. A node receives probe packets of a session in each of its outgoing links on the path of the session. It merges all of these into a single probe packet, and sends it upstream. The content of the probe packet reflects the saturation status of the session in the links originating from the node [19].

When a source receives a probe packet, it generates a "saturation" or an "unsaturation" message. The nature of the message depends on the contents of the probe packet [19]. The intermediate nodes and the receivers update the saturation status of the sessions based on the nature of this message. A receiver sends a backward rate packet upstream on receiving an unsaturation message. The backward rate packets start a new iteration.

Complexity: The distributed implementation terminates in $2DM$ units of time where D is the maximum round trip delay from a source to a receiver in the network. The complexity can be analyzed as follows. Let a receiver send a backward rate packet at time $t = 0$. The forward rate packet returns to the receiver in D units of time. It sends a probe after receiving the forward rate packet. The saturation message reaches the receiver in D units of time. Thus one iteration is completed in $2D$ units of time. As Theorem 1 indicates, any node needs to perform at most M iterations. Note that the termination time does not explicitly depend on the number of links, whereas the complexity of the basic distributed algorithm is $O(\mathcal{L}M^2)$. However, the round trip delay D depends on the number of links in the path of a session. The convergence time for the distributed implementation does not depend on the total number of links in the network as many of the link computations can be performed in parallel. The round trip delay combines the propagation and the processing delays. The worst case bound of $O(DM)$ can become large for a large network. However, as our experimental results indicate, faster convergence is attained in practice.

3.3 Experimental Results

Using simulation, we evaluate the performance of the scheme presented in Section 3.2 in a dynamic network where multicast group memberships change. Simulation results indicate that the convergence time is much less than the analytical worst case bound of $O(DM)$, and also corroborate that the control message overhead is low.

We consider a 15 session 400 node random network for experimental evaluation. Nodes are points on a $20X20$ grid. There exists an edge between any two nodes with a probability (p) that depends on the euclidean distance between the nodes (d) ($p = \exp(\alpha(1-d))$), where α is the decay constant. We assume $\alpha = 2$. We adopt this edge probability model because distant nodes are less likely to have an edge between them. The source of every session has

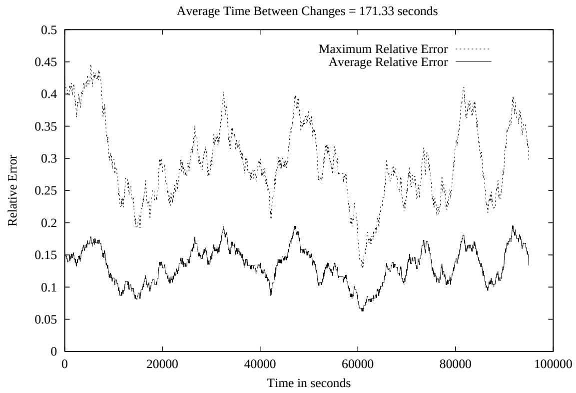

Figure 2: We study the relative computation error in a dynamic network with frequent session membership changes. Mean time between the session membership changes is 171.33 seconds.

been selected uniformly. A node is a receiver of a session with a given probability which was 0.02 in this case. The session route consists of the shortest paths between the source and the destinations. Propagation delay in a link is equal to the euclidean distance between the end nodes of the link. Maximum round trip propagation delay in this network is 45.54 seconds. We ignore processing delays. Every link is bidirectional. Receivers join and leave the sessions randomly as per Poisson random processes, and do not have any minimum rate requirement. Link occupancy (i.e., the number of sessions traversing a link) changes as receivers join and leave. Maximum number of receivers at any time is 96 and when all these receivers are present the average link occupancy is 1.118557 and the maximum number of sessions traversing a link is 5.

First, we examine the convergence time of the algorithm in this dynamic network. The computation restarts whenever there is a session membership change. The computed rate for a user may not converge to its maxmin fair rate before the next session membership change. This occurs when there are frequent changes. So we studied the discrepancy from the maxmin fair rate at any time. If the maxmin fair rate of a receiver s at time t is $r_s^m(t)$, and the computed rate at time t is $r_s^c(t)$, then relative computation error for receiver s is $(1 - \frac{r_s^c(t)}{r_s^m(t)})$ at time t. Figure 2 plots a moving average of the maximum of the relative errors of the receivers and the average of the relative errors of the receivers. The maximum and the average are taken over all currently active receivers, and the moving average is computed with a window of 5000 seconds. The algorithm restarts and the relative computation error increases to 1 every time there is a session membership change. Subsequently, the relative computation error decreases steadily till the next change event. Since the time window is large, the moving average of the errors does not drop to zero. The moving average exhibits sharp increase over the periods with more frequent session membership changes. However, average errors are small in general (always less than 0.2). Thus, even in presence of frequent session membership changes, the computed rate of an average receiver is close to the maxmin fair rate. In this figure, a receiver's average rate is always above 80% of its maxmin fair rate. The maximum relative error is somewhat higher, indicating that the convergence is slower for a few receivers.

Figure 3(a) plots the percentage of receivers which attain the maxmin fair rate before

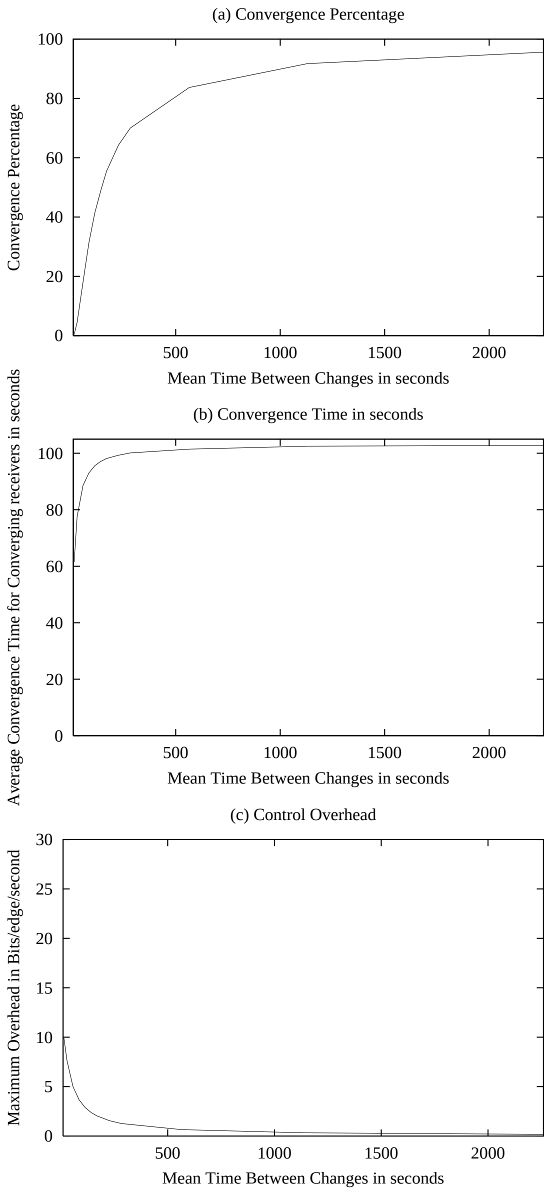

Figure 3: We study the convergence speed and the control information exchange complexity for a dynamic network with session membership changes. The distributed computation does not always converge in between the changes, on account of frequent session membership changes. Figure (a) plots the average percentage of receivers attaining their respective maxmin fair rates in between the changes. Figure (b) plots the average convergence time of these receivers. Figure (c) studies the control information traversing each link per second. It plots the maximum of this control information per second, the maximum taken over all links. This control information comprises of rate packets, probe packets and saturation/unsaturation messages.

the next change versus the mean time between changes. As expected, this percentage is low for a high frequency of membership change, and increases with decrease in the frequency of membership change. However, on an average, around 80% receivers converge to their maxmin fair rates, when the average time between changes equal 500 seconds.

Figure 3(b) presents the curve of average convergence time for the receivers which converge to their maxmin fair rates, in between two membership change events versus the mean time between changes. This convergence time is low for a high frequency of membership change, because when session membership changes rapidly, only the receivers who need little time to converge to the maxmin fair rates, actually converge to the maxmin fair rates, and others do not converge. As the frequency of membership change decreases, most of the receivers' rates converge to the maxmin fair values between the change events, and the convergence time represents the convergence time of an average receiver in the network. When the mean time between changes equal 2259.34 seconds, around 95.6% receivers' rates converge to the maxmin fair values between changes on an average. The mean convergence time of the converging receivers in this case is 102.81 seconds. This represents the convergence time of an average receiver, because the percentage of converging ones is quite high. On an average, around 47 receivers are members of different sessions at any time. Thus worst case convergence time of $2DM$ is 4280.76. Similarly, convergence time is 155.89 seconds in the same random network with 96 receivers. The worst case bound is 8743.68 in this case. Thus convergence time is normally significantly better than the analytical bound of $2DM$ seconds.

Now we examine the message exchange overhead. We measure the number of control message bits traversing a link per second. The control message comprises of rate packets, probe packets and saturation/unsaturation messages. We plot the maximum of this overhead per second against the mean time between membership changes in figure 3(c). The maximum is taken over all links. This maximum is always less than 10 bits per second, and decreases further with a decrease in the frequency of session membership change. This is because as the session membership changes less frequently, a larger number of receivers saturate in between the changes, and a receiver stops exchanging messages after it saturates. So, for a low frequency of membership change, receivers do not send control messages for long periods of times between the changes. Figure 3(c) indicates that the message exchange complexity is indeed low for this algorithm. This plot does not show the bytes consumed in the TCP and IP headers. If IP headers are considered (20 octet [5]), then the maximum overhead is around 58 bits per second instead of 10, which is still low.

4 Discussion

In this section, we discuss certain generic features of the fair rate allocation procedure.

Allocation of Rates: First we discuss how to attain continuous rates. We would also discuss how to allocate the fair rates once they are computed. Several approaches are possible. (a) Each source transmit at a rate equal to the maximum of the fair rates allocated to its receivers. A source knows this maximum value from the distributed computation. Rate adaptive video gateways are used at the forking points to transcode the signal into a lower bit

rate such that the rate in every link is equal to the maxmin fair session link rate [1, 21]. These gateways control the rate by dropping frames as necessary. Active network architecture [23] provides a framework for deployment of rate adaptive video gateways within the network.
(b) A second solution is to use layered encoding [4, 9, 24]. An information stream is partitioned in to a base layer, comprising the information needed to represent the lowest fidelity media and a number of enhancement layers. Some number of these enhancement layers are combined by the decoder with the base layer to recover a signal of incremental fidelity. Oftentimes, layer bandwidth can be tuned to provide the desired rates, e.g., by using an *embedded code*. In an embedded code, any prefix of a valid codeword is a valid codeword and if the code is chosen appropriately, the prefix codeword corresponds to a lower quality version of the longer codeword. Hence, one can trivially form a layered code from an embedded code by breaking up the longer code at arbitrary boundaries. Moreover, one can generate as many layers as desired and tune the rates of the different layers since the embedded code can be broken at arbitrary points. Mccanne [15] presents a low complexity video codec which uses an instance of an embedded code, PVH (progressive video with hybrid transform), and is thus amenable to dynamic rate control. Once the fair rates are computed, the source partitions its signal to form as many layers as there are distinct receiver rates and tunes the layer bandwidth to match the receiver rates. Each layer is sent as a separate multicast group, and a receiver's fair rate determines the number of layers delivered to a receiver. The total number of layers transmitted across a link equals the maximum number of layers allotted to the session receivers downstream. Layers are dropped at the forking points selectively. The advantage of this approach is that it is an end to end one, while the previous requires network participation.
We would like to mention that neither of these approaches is crucial for attaining the fair rates. We described the above approaches as possibilities for attaining any desired bandwidth granularity. The basic assumption in our case is that the bandwidth can be allotted in reasonably fine granularity. If a perfect match between the layer bandwidth and the maxmin fair receiver rates is not feasible, then the allocated rates can still approximate the computed values. The receivers can subscribe to as many layers as permitted by the computed fair rate. For instance, if the source of session 1 in Example 3.1.1 (Figure 1) can transmit unit bandwidth layers only, then receivers u_1 and u_2 will be allotted $4, 3$ layers respectively. If the layer granularity is fine, then the approximation will be satisfactory.

If the layer granularity is coarse, then the assumption of a continuous feasible set is not valid, and discrete feasible sets must be considered. Fairness in a discrete feasible set is vastly different from fairness in a continuous feasible set, and neither is a special case of the other. For example, the usual notions of fairness either do not exist (e.g., maxmin fairness) or are computationally NP-hard in the discrete case, while these are polynomial complexity computable in the continuous case. Weaker notions of fairness like maximal fairness need to be used in the discrete case. If the granularity is fine, then the computation of the maxmin fair allocation with a continuous set assumption and subsequent approximation as discussed before gives a better approximation of the maxmin fair rates than the weaker notion. Also, the computation strategy in the discrete case is different from that in the continuous case. We address the discrete scenario in [20].

Scalability: The rate allocation needs to consider virtual session rates. Nevertheless, the distributed algorithm is designed so that the system does not need separate entries for separate virtual sessions traversing a link. Thus the routers are oblivious of the number and identity of the receivers downstream of a link. Also, the amount of control messages of a session in a link in an iteration is independent of the number of receivers downstream of the link.

The system needs per-session states at the intermediate nodes. Arguing in the lines of Grossglausser and Bolot [10], implementing a multicast/multilayer service requires per-session state in the routers anyway. So, the incremental cost of maintaining some more information for each session and using this additional information in the fair rate allocation policy is much smaller than that in the unicast case. However, if these additional session states become an issue, then this policy can be used in the VPNs and intranets, and state aggregation may be resorted to in the backbones. The algorithm will provide the same fairness guarantees if individual sessions have bandwidth guarantees from the backbone. In this case, the backbone will be treated as a single link with only one session and capacity equaling the bandwidth guarantee for the session.

Intermediate Feasibility: The intermediate rates are always feasible (Lemma 7). So, during the computation, sources can still transmit at intermediate rates, without causing congestion.

Dynamic Version: The computations must restart if the network changes, i.e., if a new receiver joins a session, or an existing receiver leaves a session, or if the available link capacity changes. So this algorithm is not appropriate when the sessions have short lives, but operates very well if sessions have long lives, and session memberships change less frequently. So, the target applications are video-conference, distance-learning, etc., and the application domains can be intranets and VPNs.

An interesting direction for future research will be to adapt the algorithm for applications with a large number of receivers that join and leave frequently, e.g., a CNN live telecast of an Olympic event. We present possible approaches here. Receivers do not have minimum rate constraints in these applications. Maxmin fair rates are computed periodically. If a receiver leaves an application in between, then the maxmin fair rates of the remaining ones are not altered. This may cause under-utilization of link bandwidth. The amount of under-utilization will depend on the frequency of the computation and the frequency of the leave events. Note that the link bandwidth will be under-utilized only in those links where the receiver which left had the highest bandwidth among all receivers in its session. Consider the case when a receiver joins an application. Assume that the receiver has a dedicated bandwidth in its last link, and other existing receivers of the same application share other links in its path. This is typically the case, e.g., quite a few users subscribe to a CNN event in each domain. Maxmin fair bandwidth of the new receiver is the minimum of its dedicated bandwidth in its last link and the bandwidth allocated to the application in the link before the last. The maxmin fair bandwidth of the remaining receivers are the same as before. If the last link is a shared medium, e.g., an ethernet connection, then it is likely that a few other users using the shared medium have already subscribed to the application. In this case, due to the multicast nature of the transmission, the new receiver can be assigned the same bandwidth as the existing receivers without introducing any additional load in the links. The challenge will be to accommodate the cases where a receiver is the only one of its application in a link which is not the last hop. In these cases which are likely to be infrequent, the residual bandwidth in the applicable links can be used to serve the new receiver in the intervals between the computations. Normally, the links have unused bandwidth to accommodate transients. The new receivers can also conduct RLM-like layer join experiments to decide the amount of subscription in the interim period. The existing receivers can utilize the residual bandwidth generated by the leave events by the same mechanism. The difference between the bandwidth allocated to the receivers and the maxmin fair allocation will depend on the frequency of computation and the network topology.

We now present techniques for attaining the maxmin fair rates with partial recomputation in the event of topology changes. Let the algorithm execute its kth iteration[†] when there is a change. We first determine a value γ.

- When a receiver s leaves, $\gamma = r_s(k)$.
- When a receiver s joins, $\gamma = \min_{l \in L_s} C_l/|n(l)|$ ($n(l)$ is the updated set of sessions traversing a link l).
- If the capacity of a link l changes, $\gamma = C_l/|n(l)|$.

In each of these cases, any virtual session with rate greater than or equal to γ is considered unsaturated. The saturation state and the rate of any other virtual session is not changed. In the last case, if C_l increases, and l is the only bottleneck link of some virtual session j, the saturation state of j is set unsaturated as well.

The algorithm continues (starts if it had terminated) with the updated saturation states and rate values. The algorithm can use the previously computed rates and the saturation states of all virtual sessions with rates less than γ. The rates of the other virtual sessions are not used as they are rendered unsaturated. The output still converges to the maxmin fair rates. The intuition behind the result is that any change in the maxmin fair rate of a virtual session s, r_s, does not affect the rates of receivers that have maxmin fair rates lower than r_s, but may affect those of receivers with higher maxmin fair rates. This is because the maxmin fair rate of a receiver is determined by that offered by its bottleneck link, and a receiver s does not share its bottleneck link with any other receiver which has maxmin fair rate higher than that of s [19]. Thus, if the frequency of departure/arrival is the same for all receivers, then 50% of the computation can be re-used on an average.

The algorithm can also terminate prematurely after a few iterations so as to reduce the computation complexity. We present some bounds on the rate of virtual sessions in the intermediate steps for this purpose. The proof for theorem 1 shows that at least one virtual session saturates in every iteration. A virtual session attains its maxmin fair rate by the time it saturates. Thus at least k virtual sessions attain their maxmin fair rates at the end of k iterations. If there are no minimum rate constraints, then it follows from the proof for Lemma 3 that at the end of the kth iteration the rate $r_s(k)$ of any virtual session s is upper

[†]Algorithm may also have terminated. In that case $k = M$.

bounded by its maxmin fair rate and lower bounded by the minimum of its maxmin fair rate and the kth smallest component in the maxmin fair allocation [19]. Thus partial convergence is attained if the algorithm terminates prematurely.

5 Conclusion

Summarizing, we have presented a quantitative framework that can model the fair allocation of bandwidth in the multicast scenario, while considering both fairness among the members of the same and different sessions. We have presented an algorithm for computing the maxmin fair rates in arbitrary multicast networks with minimum rate requirements. We have also presented a framework for a distributed implementation of the algorithm.

References

[1] E. Amir, S. McCanne and H. Zhang An Application Level Video Gateway *Proceedings of ACM Multimedia'* 95, pp 255 – 265, San Francisco, CA, November 1995. ACM.

[2] T. Ballardie, P. Francis, and J. Crowcroft. Core based trees: an architecture for scalable inter-domain multicast routing, *Proceedings of ACM SIGCOMM,* Ithaca, NY, Sept. 1993, pp. 85-95

[3] D. Bertsekas and R. Gallager, *Data Networks*, Englewood Cliffs, NJ:Prentice-Hall, 1987.

[4] T. Bially, B. Gold, and S. Seneff. A technique for Adaptive Voice Flow Control in Integrated Packet Networks, *IEEE Transactions on Communications,* Vol. 28, No. 3, March 1980, pp. 325 – 333

[5] D. Comer, "Internetworking with TCP/IP Vol. I: Principles, Protocols, and Architecture, 4/e," (2000) ISBN 0-13-018380-6

[6] S. Chiung, M. Ammar and X. Li. On the use of Destination Set Grouping to Improve Fairness in Multicast Video Distribution *IEEE INFOCOM'96,* pp. 553 – 560, (San Francisco, CA), March 1996

[7] S. Deering and D. Cheriton. Multicast routing in datagram internetworks and extended LANs, ACM Transactions on Computer Systems, vol 8, no. 2, pp. 54-60, Aug. 1994.

[8] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei. An architecture for wide area multicast routing, *Proceedings of ACM SIGCOMM,* London, UK, Aug. 1994, pp. 126-135.

[9] M. Ghanbari. Two-Layer Coding of Video Signals for VBR Networks. *IEEE Journal on Selected Areas in Communications*, VOL. 7. No. 5. June 1989.

[10] M. Grossglausser and J. Bolot. On service models for Multicast Transmission in Heterogeneous Environments. *Proceedings of IEEE' INFOCOM'* 2000, Tel-Aviv, Israel, 2000, pp. 71 – 80, April, 2000.

[11] F. Kishino, K. Manabe, Y. Hayashi and H. Yasuda. Variable Bit-Rate Coding of Video Signals for ATM Networks, *IEEE Journal on Selected Areas In Communications*, Vol. 7, No. 5, pp. 801 – 806, June 1989.

[12] X. Li, S. Paul and M. H. Ammar. Layered Video Multicast with Retransmission (LVMR): Evaluation of Hierarchical rate control, *Proceedings of IEEE Infocom' 98*, pp. 1062 – 1072, San Francisco, CA, March 1998,

[13] X. Li, S. Paul and M. H. Ammar. Multi-session Rate Control for Layered Video Multicast *Technical Report* GT-CC-98 – 21, College of Computing, Georgia Institute of Technology, 1998

[14] J. Moy, Multicast routing extensions for OSPF, *Comm, ACM* vol 37, no. 8, pp 61-66, Aug 94.

[15] S. McCanne, Scalable Compression and Transmission of Internet Multicast Video. *PhD thesis*, Univ. of California, Berkeley, December 1996.

[16] S. McCanne, V. Jacobson and M. Vetterli. Receiver-Driven Layered Multicast, *Proceedings of ACM SIGCOMM '96*, pp. 117 – –130, Stanford, CA, September 1996

[17] D. Rubenstein, J. Kurose and D. Towsley. The Impact of Multicast Layering on Network Fairness *IEEE Transactions on Networking,* Vol. 10, No. 2, pp. 169 – 182, April 2002

[18] S. Sarkar and L. Tassiulas: A Framework for Routing and Congestion Control for Multicast Information Flows, *IEEE Transactions on Information Theory,* October 2002, Vol. 48, No. 10, pp. 2690-2708

[19] S. Sarkar and L. Tassiulas: Distributed Algorithms for Computation of Fair Rates in Multirate Multicast Trees *Technical Report* TR 99 – 42, Institute for Systems Research and University of Maryland, 1999, A more recent version available at http://www.seas.upenn.edu/~swati/publication.htm

[20] S. Sarkar and L. Tassiulas: Fair Allocation of Discrete Bandwidth Layers in Multicast Networks *Proceedings of INFOCOM'* 2000, pp. 1491 – 1500, Tel Aviv, Israel, March 2000

[21] T. Turletti and J. C. Bolot Issues with multicast video distribution in heterogeneous packet networks, *Packet Video Workshop*, '94, Portland, Oregon, 26-27 Sept. 94, pp. F3.1-3.4

[22] H. Y. Tzeng and K. Y. Siu. On Max-Min Fair Congestion Control for Multicast ABR Service in ATM, *IEEE Journal on Selected Areas In Communications*, Vol 15, No. 3, pp. 545-556 1997.

[23] D. L. Tennenhouse and D. J. Wetherall. Towards an active network architecture. *Computer Communication Review,*, 26(2) : 5 − 18, April 1996.

[24] D. Taubman and A. Zakhor. Multirate 3-D Subband Coding of Video, *IEEE Transactions on Image Processing*, Vol 3, No. 5, pp. 572-588, September 1994.

A Proof of Bottleneck Lemma:

Proof of Lemma 2 (Bottleneck Lemma): Let a virtual session s not have a bottleneck link under a feasible rate vector $\vec{r}$. If s traverses a link l, then at least one of the following holds.

1. Link l has some unused capacity.
2. The rate of some other virtual session of the same session as s traversing l is greater than that of s, i.e., bandwidth consumed by the session of s in link l is greater than s's rate.
3. A session i traversing link l consumes greater bandwidth than s in link l. This bandwidth is greater than the minimum rate of session i in link l (μ_{il}). Virtual session s does not belong to session i.

If all links on s's path satisfy the first two conditions, then s's rate can be increased without decreasing that of any other virtual session. If some links on s's path satisfy only the last property, then to increase s's rate, we must decrease the rate of some virtual sessions having greater rate than that of s. However we can still increase s's rate s, without decreasing that of any other virtual session having rate less than that of s and still maintain feasibility. Thus $\vec{r}$ is not a maxmin fair rate vector.

Let $\vec{r}^1$ be a feasible rate vector such that each virtual session has a bottleneck link. Consider any other feasible rate vector $\vec{r}^2$. Let there exist a virtual session s such that $r_s^2 > r_s^1$. Let l be a bottleneck link for virtual session s. $\lambda^2_{\chi(s),l} \geq r_s^2 > r_s^1 = \lambda^1_{\chi(s),l}$. The last equality follows from the property of a bottleneck link. Since l is a bottleneck link w.r.t. virtual session s, its capacity is fully utilized, i.e., $\sum_{i \in n(l)} \lambda^1_{il} = C_l$. From the feasibility of $\vec{r}^2$, $\sum_{i \in n(l)} \lambda^2_{il} \leq C_l$. Since $\lambda^2_{\chi(s),l} > \lambda^1_{\chi(s),l}$, and $\chi(s) \in n(l)$ (since virtual session s traverses through link l), it follows that there exists a session j such that $\lambda^2_{jl} < \lambda^1_{jl}$. From feasibility of $\vec{r}^2$, $\lambda^2_{jl} \geq \mu_{jl}$. Thus, $\lambda^1_{jl} > \mu_{jl}$. There exists virtual session t such that $t \in m(j,l)$, $r_t^1 = \lambda^1_{jl}$. Thus, $r_t^1 > \mu_{\chi(t)l}$. From the last condition for a link to be bottleneck w.r.t. a virtual session, $r_t^1 \leq r_s^1$. Now $r_t^2 \leq \lambda^2_{jl} < \lambda^1_{jl} = r_t^1$. The first inequality follows since $\chi(t) = j$. The second inequality and the last equality have been argued before. Thus, if $r_s^2 > r_s^1$, then there exists a virtual session t such that $r_t^2 < r_t^1 \leq r_s^1$. Hence $\vec{r}^1$ is a maxmin fair rate vector. □

B Proof of correctness and termination guarantee for the algorithm for computation of Maxmin fair rates (Theorem 1)

We outline the proof as follows. We assume that the set of feasible rate vectors is nonempty. The first part of the proof shows that the output of this algorithm is maxmin fair. For this, we first show that the link control parameters increase with every iteration (Lemma 4). Thus, the virtual session rates and the session rates do not decrease in subsequent iterations (Lemma 5). Using this, we show that the rate allocation at the end of every iteration is feasible (Lemma 7). Next we show that, if a virtual session saturates in the kth iteration, it has a bottleneck link in subsequent iterations (Lemma 8). Since the algorithm terminates only when all virtual sessions saturate, each virtual sessions has a bottleneck link when the algorithm terminates. The rate allocation upon termination is also feasible by Lemma 7. Thus maxmin fairness of the rate allocation upon termination follows from the Bottleneck Lemma. For the second part of the theorem we show in Lemma 9 that the algorithm terminates in at most M iterations.

Lemma 4 *If $k \geq 1$ and the algorithm has not terminated in $k-1$ iterations, $\eta_l(k) \geq \eta_l(k-1)$ $\forall k \geq 1$.*

Proof of Lemma 4: We prove by induction. Let $k = 1$. The algorithm can not terminate in 0 iteration. $S(0) \neq \phi$, $F_l(0) = 0$. If $\Xi_l(0) = \phi$, then $\eta_l(1) = \eta_l(0) = 0$, and the lemma holds for link l and iteration 1. Let $\Xi_l(0) \neq \phi$. Since the feasible set of rate vectors is nonempty, $\sum_{i \in \Xi_l(0)} \mu_{il} \leq C_l$. $\mu_{il} \geq 0$, for every session i and link l. Thus $\theta = 0$, satisfies the inequality $\sum_{i \in \Xi_l(0)} \max(\theta, \mu_{il}) \leq C_l$. Clearly $\eta_l(1)$ is the maximum possible θ which satisfies the above inequality. Hence $\eta_l(1) \geq 0 = \eta_l(0)$. The equality follows from the initialization of $\eta_l(0)$. Thus the lemma holds for $k = 1$.

Let the lemma hold for iterations $1, \ldots, k$ and $S(k) \neq \phi$. We show that the lemma holds for the $k+1$th iteration. If $\Xi_l(k) = \phi$, then $\eta_l(k+1) = \eta_l(k)$. Let $\Xi_l(k) \neq \phi$. Consider any virtual session s traversing through link l, i.e., $\chi(s) \in n(l)$. If $s \in S(k-1)$, $r_s(k) \leq \eta_{\chi(s)l}(k) = \max(\eta_l(k), \mu_{\chi(s)l})$. Let $s \notin S(k-1)$. Thus $s \in S(t) \setminus S(t+1)$, $0 \leq t < k-1$. $r_s(t+1) \leq \max(\eta_l(t+1), \mu_{\chi(s)l}) \leq \max(\eta_l(k), \mu_{\chi(s)l})$, since $t+1 < k$, and $\eta_l(0) \leq \eta_l(1) \leq \ldots \leq \eta_l(k)$ by induction hypothesis. Since $S(k-1) \subseteq \ldots \subseteq S(t+1)$ ($t+1 < k$ and $S(p+1) \subseteq S(p)$, $\forall p$), $s \notin S(t+1), \ldots, S(k-1)$. Thus $r_s(k) = \ldots = r_s(t+1) \leq \max(\eta_l(k), \mu_{\chi(s)l})$.

$$
\begin{aligned}
r_s(k) &\leq \max(\eta_l(k), \mu_{\chi(s)l}) \text{ for all } s && (1)\\
\text{Thus } \lambda_{il}(k) &\leq \max(\eta_l(k), \mu_{il}) && (2)
\end{aligned}
$$

$$
\begin{aligned}
& F_l(k) - F_l(k-1) \\
& \sum_{i \in \Xi_l(k-1) \setminus \Xi_l(k)} \lambda_{il}(k) \\
\leq \quad & \sum_{i \in \Xi_l(k-1) \setminus \Xi_l(k)} \max(\eta_l(k), \mu_{il}) \text{ (from (2))}
\end{aligned}
$$

$$
\begin{aligned}
& F_l(k) + \sum_{i \in \Xi_l(k)} \max(\eta_l(k), \mu_{il}) \\
\leq\ & F_l(k-1) + \sum_{i \in \Xi_l(k-1) \setminus \Xi_l(k)} \max(\eta_l(k), \mu_{il}) + \\
& \sum_{i \in \Xi_l(k)} \max(\eta_l(k), \mu_{il}) \\
=\ & F_l(k-1) + \sum_{i \in \Xi_l(k-1)} \max(\eta_l(k), \mu_{il}) \\
& (\text{since } \Xi_l(k) \subseteq \Xi_l(k-1) \text{ as } S(k) \subseteq S(k-1)) \\
=\ & C_l
\end{aligned}
$$

Thus $\theta = \eta_l(k)$ satisfies the inequality $F_l(k) + \sum_{i \in \Xi_l(k)} \max(\theta, \mu_{il}) \leq C_l$. Clearly $\eta_l(k+1)$ is the maximum possible value of θ which satisfies the inequality and hence there exists a $\eta_l(k+1)$ and $\eta_l(k+1) \geq \eta_l(k)$. Hence the result follows from induction. □

Lemma 5 *$r_s(k+1) \geq r_s(k)$ if $k \geq 0$ for all virtual sessions s. $\lambda_{il}(k+1) \geq \lambda_{il}(k)$ if $k \geq 1$ for all sessions i and links l.*

Proof of Lemma 5: Let $k = 0$. Now, $s \in S(0)$, $r_s(1) = \min_{l \in L_s} \eta_{\chi(s)l}(1)$, $\eta_{\chi(s)l}(1) = \max(\eta_l(1), \mu_{\chi(s)l}) \geq \mu_{\chi(s)l} \geq \mu_s$, for all virtual sessions s. The last inequality follows since $s \in m(\chi(s), l)$. Thus $r_s(1) \geq \mu_s = r_s(0)$. Thus the result holds for $k = 0$. Let $k \geq 1$. If $s \notin S(k)$, $r_s(k+1) = r_s(k)$. If $s \in S(k)$, $r_s(k+1) = \min_{l \in L_s} \eta_{\chi(s)l}(k+1)$. $\eta_{\chi(s)l}(k+1) = \max(\eta_l(k+1), \mu_{\chi(s)l})$. $\eta_l(k+1) \geq \eta_l(k)$. Thus $\eta_{\chi(s)l}(k+1) \geq (\eta_l(k), \mu_{\chi(s)l}) = \eta_{\chi(s)l}(k)$. Thus $r_s(k+1) \geq \min_{l \in L_s} \eta_{\chi(s)l}(k)$.Since $s \in S(k)$, and $S(k-1) \supseteq S(k)$, $s \in S(k-1)$. Thus $r_s(k) = \min_{l \in L_s} \eta_{\chi(s)l}(k)$. Hence $r_s(k+1) \geq r_s(k)$. The second part of the lemma follows from the first and the fact that $\lambda_{il}(k) = \max_{s \in m(i,l)} r_s(k)$ □

Lemma 6

$$
\begin{aligned}
r_s(k) &\leq \eta_{\chi(s)l}(k) \text{ for all virtual sessions } s, \text{ links } l \in L_s \text{ and } k \geq 0 && (3) \\
\textit{Thus } \lambda_{il}(k) &\leq \eta_{il}(k) \text{ for all sessions } i, \text{ links } l \text{ and } k > 0 && (4)
\end{aligned}
$$

Remark: We will use this lemma in proofs of feasibility of rate allocations (Lemma 7), the fact that every saturated virtual session has a bottleneck link (Lemma 8) and the fact that the algorithm terminates in finite number of iterations (Theorem 9).

Proof of Lemma 6: We prove (3) by induction. $r_s(0) = \mu_s \leq \max(\eta_l(0), \mu_{\chi(s)l})\ \forall\ l \in L_s$. The last inequality follows since $\mu_{\chi(s)l} = \max_{j \in m(\chi(s),l)} \mu_j$ and $s \in m(\chi(s), l)$, since $l \in L_s$. Thus (3) holds for $k = 0$.

Let (3) hold for $k \geq 0$. If $s \in S(k)$, $r_s(k+1) \leq \eta_{\chi(s)l}(k+1)$. If $s \notin S(k)$,

$$
\begin{aligned}
r_s(k+1) &= r_s(k) \\
&\leq \eta_{\chi(s)l}(k) \text{ (from induction hypothesis)} \\
&= \max(\eta_l(k), \mu_{\chi(s)l}) \\
&\leq \max(\eta_l(k+1), \mu_{\chi(s)l}) \text{ (} \eta_l(k) \leq \eta_l(k+1) \text{ by Lemma 4)} \\
&\leq \eta_{\chi(s)l}(k+1)
\end{aligned}
$$

Thus (3) holds for $k+1$. Thus (3) holds for all $k \geq 0$ by induction. Now, (4) follows from (3) and the definition of $\lambda_{il}(k)$. $\square$

Lemma 7 *The rate allocation at the end of the kth iteration is feasible, $k \geq 0$.*

Proof of Lemma 7: We prove by induction. We first prove the lemma for $k = 0$. Since the set of feasible rate vectors is nonempty, a rate allocation where each virtual session s's rate equals its minimum rate, satisfies the capacity constraints. Thus $\vec{r}(0)$ satisfies the capacity constraints. Since $r_s(0) = \mu_s$, $\vec{r}(0)$ satisfies the minimum rate requirements. Thus $\vec{r}(0)$ is feasible and the lemma holds for $k = 0$.

Let the rate allocation at the end of the kth iteration, $\vec{r}(k)$, be feasible. Consider the $k+1$th iteration. $r_s(k+1) \geq r_s(k) \geq \mu_s$. The first inequality follows from Lemma 5 and the last from the feasibility of $\vec{r}(k)$. Thus $r_s(k+1) \geq \mu_s$, for all virtual sessions s.

$$\begin{aligned} r_s(k+1) &= r_s(k) \text{ (if } \chi(s) \in n(l) \setminus \Xi_l(k) \text{ since then } s \notin S(k)) \\ \text{Thus } \lambda_{il}(k+1) &= \max_{s \in m(i,l)} r_s(k) \text{ (if } i \in n(l) \setminus \Xi_l(k)) \\ &= \lambda_{il}(k) \text{ (if } i \in n(l) \setminus \Xi_l(k)) \end{aligned} \tag{5}$$

If $\Xi_l(k) = \phi$, from (5), $\lambda_{il}(k+1) = \lambda_{il}(k)\ \forall\ i \in n(l)$.

$$\begin{aligned} \sum_{i \in n(l)} \lambda_{il}(k+1) &= \sum_{i \in n(l)} \lambda_{il}(k) \\ &\leq C_l \text{ (from the feasibility of } \vec{r}(k)) \end{aligned}$$

If $\Xi_l(k) \neq \phi$,

$$\begin{aligned} \sum_{i \in n(l)} \lambda_{il}(k+1) &= \sum_{i \in n(l) \setminus \Xi_l(k)} \lambda_{il}(k+1) + \sum_{i \in \Xi_l(k)} \lambda_{il}(k+1) \\ &\leq \sum_{i \in n(l) \setminus \Xi_l(k)} \lambda_{il}(k) + \sum_{i \in \Xi_l(k)} \max(\eta_l(k+1), \mu_{il}) \\ &\qquad \text{(from (4) of Lemma 6 and (5))} \\ &= F_l(k) + \sum_{i \in \Xi_l(k)} \max(\eta_l(k+1), \mu_{il}) \\ &= C_l \end{aligned}$$

Since link l was chosen arbitrarily, the rate vector at the end of iteration $k+1$, $\vec{r}(k+1)$ satisfies the capacity condition for every link l. Hence $\vec{r}(k+1)$ is feasible. Thus the lemma follows by induction. $\square$

Lemma 8 *Let $s \in S(k-1) \setminus S(k)$. Let $t \geq k$ and $S(t-1) \neq \phi$, i.e., the algorithm does not terminate in $t-1$ iterations. Then virtual session s has a bottleneck link, under rate vector $\vec{r}(t)$.*

Proof of Lemma 8: Let $s \in S(k-1) \setminus S(k)$, i.e., virtual session s becomes saturated at the end of the kth iteration. Thus there exists link $l \in L_s$ such that

$$\begin{aligned} \sum_{i \in n(l)} \lambda_{il}(k) &= C_l & (6) \\ \text{and } r_s(k) &= \lambda_{\chi(s)l}(k) & (7) \end{aligned}$$

By Lemma 6, $r_s(k) \leq \eta_{\chi(s)l}(k)$. Let $r_s(k) < \eta_{\chi(s)l}(k)$.

$$\begin{aligned} \sum_{i \in n(l)} \lambda_{il}(k) &= \sum_{i \in n(l) \setminus \Xi_l(k-1)} \lambda_{il}(k) + \sum_{i \in \Xi_l(k-1) \setminus \chi(s)} \lambda_{il}(k) + \lambda_{\chi(s)l}(k) & (8) \\ \lambda_{il}(k) &= \lambda_{il}(k-1) \text{ if } i \in n(l) \setminus \Xi_l(k-1) \text{ (argued in (5))} & (9) \\ \lambda_{il}(k) &\leq \max(\eta_l(k), \mu_{il}) \forall i \in n(l) \text{ (from (4) of Lemma 6)} & (10) \\ \lambda_{\chi(s)l}(k) &= r_s(k) \\ &< \eta_{\chi(s)l}(k) \\ &= \max(\eta_l(k), \mu_{\chi(s)l}) & (11) \\ \sum_{i \in n(l)} \lambda_{il}(k) &< \sum_{i \in n(l) \setminus \Xi_l(k-1)} \lambda_{il}(k-1) + \sum_{i \in \Xi_l(k-1)} \max(\eta_l(k), \mu_{il}) & (12) \\ &= F_l(k-1) + \sum_{i \in \Xi_l(k-1)} \max(\eta_l(k), \mu_{il}) \\ &= C_l \text{ (since } s \in S(k-1), l \in L_s \text{ implying } \Xi_l(k-1) \neq \phi) & (13) \end{aligned}$$

Now, (8) holds since $\Xi_l(k-1) \subseteq n(l)$ and $\chi(s) \in \Xi_l(k-1)$ as $l \in L_s$, $s \in S(k-1)$, and thus $m(\chi(s), l) \cap S(k-1) \supseteq \{s\}$. Now, (12) follows from (8), (9) and (11). (13) contradicts (6).

$$\text{Thus, } r_s(k) = \eta_{\chi(s)l}(k) \tag{14}$$

Consider the t th iteration, $t \geq k$.

$$\begin{aligned} \sum_{i \in n(l)} \lambda_{il}(t) &\geq \sum_{i \in n(l)} \lambda_{il}(k) \text{ (since } \lambda_{il}(t) \geq \lambda_{il}(k), t \geq k \text{ from Lemma 5)} \\ &= C_l \\ \sum_{i \in n(l)} \lambda_{il}(t) &\leq C_l \text{ (from feasibility of } \vec{r}(t)) \\ \text{Thus } \sum_{i \in n(l)} \lambda_{il}(t) &= C_l & (15) \end{aligned}$$

Since $\lambda_{il_1}(t) \geq \lambda_{il_1}(k)$, for all sessions i, links l_1, if $\lambda_{il}(t) > \lambda_{il}(k)$, for some $i \in n(l)$, $\sum_{i \in n(l)} \lambda_{il}(t) > \sum_{i \in n(l)} \lambda_{il}(k)$. Thus from (6) $\sum_{i \in n(l)} \lambda_{il}(t) > C_l$. This contradicts the feasibility of $\vec{r}(t)$ ($\vec{r}(t)$ is feasible by Lemma 7). Thus

$$\lambda_{il}(t) = \lambda_{il}(k) \ \forall \ i \in n(l) \tag{16}$$

Since $s \notin S(k)$, $r_s(t) = r_s(k)$, $\forall t \geq k$. $r_s(k) = \lambda_{\chi(s)l}(k)$ by (7). $l \in L_s$ and thus $\chi(s) \in n(l)$. Thus from (16)

$$r_s(t) = \lambda_{\chi(s)l}(t) \tag{17}$$

Let j be a virtual session traversing through link l and $r_j(t) > r_s(t)$.

$$
\begin{aligned}
\max(\eta_l(k), \mu_{\chi(j)l}) &\geq \lambda_{\chi(j)l}(k) \text{ (from (4) of Lemma 6)} \\
&= \lambda_{\chi(j)l}(t) \text{ (from (16))} \\
&\geq r_j(t) \qquad\qquad (18)\\
&> r_s(t) \text{ (by assumption)} \\
&\geq r_s(k) \text{ (by Lemma 5 since } t \geq k) \\
&= \max(\eta_l(k), \mu_{\chi(s)l}) \text{ (by (14))} \\
\text{Thus } \max(\eta_l(k), \mu_{\chi(j)l}) &= \mu_{\chi(j)l} \qquad\qquad (19)\\
\text{Thus } r_j(t) &\leq \mu_{\chi(j)l} \text{ if } r_j(t) > r_s(t) \text{ and } l \in L_j \qquad\qquad (20)\\
&\qquad \text{(from (18) and (19))}
\end{aligned}
$$

From (15), (17) and (20), link l is bottlenecked w.r.t. virtual session s under rate vector $\vec{r}(t)$. □

Lemma 9 *The algorithm terminates in at most M iterations, where M is the number of virtual sessions in the system. The algorithm terminates in at most* $\min(|\mathcal{L}|, M)$ *iterations, if $\mu_i = \mu_j$ for all i, j such that $\chi(i) = \chi(j)$ and $L_i \cap L_j \neq \phi$.*

Proof of Lemma 9: For the first part, it is sufficient to prove that $S(k)$ is a proper subset of $S(k-1)$, $\forall\ k$ s.t. $S(k-1) \neq \phi$ and $k \geq 1$. Since $|S(0)| = M$, the result follows. Let $l_{\min} = \arg\min_{l \in \cup_{s \in S(k-1)} L_s} \eta_l(k)$. $l_{\min}$ is well defined as $S(k-1) \neq \phi$ (If more than one links attain the minimum, choose any one of them as $l_{\min}$). Since $l_{\min} \in \cup_{s \in S(k-1)} L_s$, at least one virtual session in $S(k-1)$ (unsaturated virtual session) traverses through link $l_{\min}$. Thus $\Xi_{l_{\min}}(k-1) \neq \phi$. Thus $\eta_{l_{\min}}(k)$ is the maximum θ which satisfies $F_{l_{\min}}(k-1) + \sum_{i \in \Xi_{l_{\min}}(k-1)} \max(\theta, \mu_{il_{\min}}) \leq C_{l_{\min}}$ and observe that

$$
F_{l_{\min}}(k-1) + \sum_{j \in \Xi_{l_{\min}}(k-1)} \max\left((\min_{i \in \Xi_{l_{\min}}(k-1)} \mu_{il_{\min}}), \mu_{jl_{\min}} \right) \leq F_{l_{\min}}(k-1) + \sum_{i \in \Xi_{l_{\min}}(k-1)} \max(\theta, \mu_{jl_{\min}})\ \ \forall\ \theta
$$

since $\min_{i \in \Xi_{l_{\min}}(k-1)} \mu_{il_{\min}} \leq \mu_{jl_{\min}}\ \forall\ j \in \Xi_{l_{\min}}(k-1)$.

$$
\text{Thus, } \eta_{l_{\min}}(k) \geq \min_{i \in \Xi_{l_{\min}}(k-1)} \mu_{il_{\min}} \qquad (21)
$$

Let $p \in \Xi_{l_{\min}}(k-1)$ such that

$$
\eta_{l_{\min}}(k) \geq \mu_{pl_{\min}} \qquad (22)
$$

Let $i_{\min} = \arg\min_{i \in \Xi_{l_{\min}}(k-1)} \mu_{il_{\min}}$. $i_{\min}$ is well defined as $\Xi_{l_{\min}} \neq \phi$. Then $i_{\min}$ is one such p from (21).

$$
\begin{aligned}
\eta_{\chi(s)l_{\min}}(k) &= \max(\eta_{l_{\min}}(k), \mu_{pl_{\min}})\ \forall\ s \in m(p, l_{\min}) \cap S(k-1) \\
&= \eta_{l_{\min}}(k) \text{ (from (22))} \qquad\qquad (23)
\end{aligned}
$$

Consider a $s \in m(p, l_{\min}) \cap S(k-1)$ ($m(p, l_{\min}) \cap S(k-1) \neq \phi$ as $p \in \Xi_{l_{\min}}(k-1)$).

$$\begin{aligned}
\eta_{\chi(s)l_{\min}}(k) &= \eta_{l_{\min}}(k) \text{ (from (23))} \\
&\leq \eta_l(k)\ \forall\ l \in L_s \text{ (from the definition of } l_{\min} \text{ and since } s \in S(k-1)) \\
&\leq \eta_{\chi(s)l}(k)\ \forall\ l \in L_s \\
\text{Thus } r_s(k) &= \eta_{\chi(s)l_{\min}}(k) \text{ (since } s \in S(k-1), r_s(k) = \min_{l \in L_s} \eta_{\chi(s)l}(k)) \\
&= \max(\eta_{l_{\min}}(k), \mu_{\chi(s)l_{\min}}) \qquad (24) \\
&\geq \lambda_{\chi(s)l_{\min}}(k) \text{ (from (4) of Lemma 6)} \\
r_s(k) &\leq \lambda_{\chi(s)l_{\min}}(k) \\
\text{Thus } r_s(k) &= \lambda_{\chi(s)l_{\min}}(k) \qquad (25)
\end{aligned}$$

Consider any session $i \in \Xi_{l_{\min}}(k-1)$. Let $\eta_{il_{\min}}(k) = \mu_{il_{\min}}$.

$$\begin{aligned}
\lambda_{il_{\min}}(k) &= \max_{s \in m(i,l_{\min})} r_s(k) \\
&\geq \max_{s \in m(i,l_{\min})} \mu_s \text{ (from feasibility of } \vec{r}(k)) \\
&= \mu_{il_{\min}} \\
&= \eta_{il_{\min}}(k) \text{ (from hypothesis)} \\
\lambda_{il_{\min}}(k) &\leq \eta_{il_{\min}}(k) \text{ (from (4) of Lemma 6)} \\
\text{Thus } \lambda_{il_{\min}}(k) &= \eta_{il_{\min}}(k) \qquad (26)
\end{aligned}$$

Now let $\eta_{il_{\min}}(k) = \eta_{l_{\min}}(k)$. $r_j(k) = \eta_{il_{\min}}(k), \forall j \in m(i, l_{\min}) \cap S(k-1)$ ($m(i, l_{\min}) \cap S(k-1) \neq \phi$ as $i \in \Xi_{l_{\min}}(k-1)$). The reasoning is exactly the same as that behind (24). $\lambda_{il_{\min}}(k) = \max_{j \in m(i,l_{\min})} r_j(k) \geq \eta_{il_{\min}}(k)$, since $m(i, l_{\min}) \cap S(k-1) \neq \phi$. From (4) of Lemma 6,

$$\lambda_{il_{\min}}(k) = \eta_{il_{\min}}(k) \qquad (27)$$

From (26) and (27)

$$\lambda_{il_{\min}}(k) = \eta_{il_{\min}}(k)\ \forall\ i \in \Xi_l(k-1) \text{ (from (26) and (27))} \qquad (28)$$

$$\begin{aligned}
\lambda_{il_{\min}}(k) &= \lambda_{il_{\min}}(k-1) \text{ if } i \in n(l_{\min}) \setminus \Xi_{l_{\min}}(k-1) \text{ (from (9))} \\
\sum_{i \in n(l_{\min}) \setminus \Xi_{l_{\min}}(k-1)} \lambda_{il_{\min}}(k) &= \sum_{i \in n(l_{\min}) \setminus \Xi_{l_{\min}}(k-1)} \lambda_{il_{\min}}(k-1) \\
&= F_{l_{\min}}(k-1) \qquad (29) \\
\sum_{i \in n(l_{\min})} \lambda_{il_{\min}}(k) &= \sum_{i \in n(l_{\min}) \setminus \Xi_{l_{\min}}(k-1)} \lambda_{il_{\min}}(k) + \sum_{i \in \Xi_{l_{\min}}(k-1)} \lambda_{il_{\min}}(k) \\
&\qquad \text{(since } \Xi_{l_{\min}}(k-1) \subseteq n(l_{\min})) \\
&= F_{l_{\min}}(k-1) + \sum_{i \in \Xi_{l_{\min}}(k-1)} \max(\eta_{l_{\min}}(k), \mu_{il_{\min}}) \\
&\qquad \text{(from (28) and (29))} \\
&= C_l \qquad \text{(since } \Xi_{l_{\min}}(k-1) \neq \phi) \qquad (30)
\end{aligned}$$

Thus from (25) and (30) $s \in S(k-1) \setminus S(k), \forall s \in m(p, l_{\min}) \cap S(k-1)$. Thus $S(k-1) \setminus S(k) \supseteq m(p, l_{\min}) \cap S(k-1)$, $S(k) \subseteq S(k-1)$ by construction, and $m(p, l_{\min}) \cap S(k-1) \neq \phi$. Thus $S(k)$ is a proper subset of $S(k-1)$. This proves the first part.

For the second part, it is sufficient to show that every session in $l_{\min}$ is saturated at the end of iteration k. We have just shown that every session $p \in \Xi_{l_{\min}}(k-1)$ is saturated at the end of iteration k if $\eta_{l_{\min}}(k) \geq \mu_{pl_{\min}}$. It we will be sufficient to show that every session $p \in \Xi_{l_{\min}}(k-1)$ for which $\eta_{l_{\min}}(k) < \mu_{pl_{\min}}$ is saturated at the end of iteration k. For any such session p, $\eta_{pl_{\min}}(k) = \mu_{pl_{\min}}$. From (4) of Lemma 6,

$$\lambda_{pl_{\min}}(k) \leq \mu_{pl_{\min}} \tag{31}$$

Consider a virtual session $s \in m(p, l_{\min}) \cap S(k-1)$. There exists such virtual sessions as session p is not saturated at the end of iteration $k-1$. Note that $r_s(k) \leq \lambda_{pl_{\min}}(k) \leq \mu_{pl_{\min}}$. The last inequality follows from (31). From the additional condition in the second part in the lemma, $\mu_{pl_{\min}} = \mu_s$ Thus, $r_s(k) \leq \mu_s$. Now from feasibility of $\vec{r}(k)$, $r_s(k) = \mu_s$. Thus $r_s(k) = \mu_{pl_{\min}} \geq \lambda_{pl_{\min}}(k)$. The last inequality follows from (31). Thus $r_s(k) = \lambda_{\chi(s)l_{\min}}(k)$ for any virtual session $s \in m(p, l_{\min}) \cap S(k-1)$. Note that the link capacity is fully utilized from (30). Thus any virtual session $s \in m(p, l_{\min}) \cap S(k-1)$ saturates in iteration k. The result follows. □

Proof of Theorem 1: If the algorithm terminates in k iterations, $S(k) = \phi$. Since $S(0)$ is the set of all virtual sessions, for every virtual session s, there exists a $t \leq k$, such that $s \in S(t-1) \setminus S(t)$, $t \leq k$. Thus by Lemma 8 virtual session s has a bottleneck link $l \in L_s$, under rate vector $\vec{r}(k)$. By Lemma 7, $\vec{r}(k)$ is a feasible rate vector. Thus $\vec{r}(k)$ is max-min fair by Lemma 2. Lemma 9 shows that the algorithm terminates in at most M iterations. □

Proof of Lemma 3: The result follows from the second part in Lemma 9.