Achieving fairness in multicasting with almost stateless rate control

Saswati Sarkar[1], Tianmin Ren[2] and Leandros Tassiulas[2]
[1]Department of Electrical and Systems Engineering, University of Pennsylvania
swati@ee.upenn.edu, 215 573 9071 (Ph) 215 573 2068 (fax)
[2]Department of Electrical and Computer Engineering and Institute for Systems Research
University of Maryland, College Park
{rtm,leandros}@isr.umd.edu, 301 405 6620(Ph)
*Address all correspondence to Saswati Sarkar

ABSTRACT

Several flow control algorithms have been proposed for attaining maxmin fair rates. All of these strategies use flow specific states at all nodes in the network, and are hence unsuitable for deployment in the current internet. We propose a new approach which attains maxmin fairness in unicast networks while maintaining per flow states at the edge nodes only. Next, we consider multirate, multicast network and argue that any flow control algorithm must use flow specific states at the forking points of the multicast tree. This happens because the packet forwarding rates must be different for different branches at the forking point, and thus the session specific forwarding rates must be stored at the forking point. Finally, we present a maxmin fair bandwidth allocation policy for multirate, multicast networks, which uses per flow states only at the forking nodes, and no per flow state at the intermediate nodes.

1. INTRODUCTION

One primary goal for rate based flow control is to achieve fairness of rate allocation among different sessions in a network. Different notions of fairness are possible and each of these leads to a different optimization objective. We adopt the notion of maxmin fairness. A rate allocation is maxmin fair, if no receiver can be allocated a higher rate without hurting another receiver having equal or lower rate. Maxmin fairness satisfies many intuitive fairness properties.

Maxmin fair allocation has been studied extensively in the unicast context[1, 4, 5, 7] and more recently in the multirate, multicast scenario.[6, 8] However, all the algorithms proposed so far need per flow state. In this paper, we propose a distributed asynchronous algorithm for computation of maxmin fair rates in a unicast network, which does not require per flow states at any intermediate node. This property makes our algorithm scalable and practical for large networks.

Next, we consider multicast networks. Multicasting allows a single source to simultaneously communicate with several receivers. We consider multirate multicast networks which allow different receivers of the same session to receive messages at different rates. Multirate mechanisms allow the network to cater to the specific requirements and capabilities of the individual receivers. However, it is not possible to eliminate per flow states at the forking nodes in any multirate congestion control scheme. This is because different branches need different transmission rates for the same session at the forking nodes, and such a node should know these bandwidth specifically in order to cater to the individual traffic demands of the different receivers. Thus session specific states must be maintained at the forking points of the multicast tree. We propose a flow control scheme for allocating the maxmin fair rates in multirate, multicast networks with per flow states only at these junction nodes and the source nodes. Thus this algorithm uses minimal per flow states in multirate, multicast network. Since a node needs to maintain per flow states for only those multicast sessions which fork at the node, storage and processing requirements at the nodes are well under control in this policy.

This paper is organized as follows. We first describe our network model and formally define our fairness objective for unicast and multirate multicast networks in Section 2. Next, we describe a stateless flow control

scheme for unicast networks in Section 3. Subsequently, we generalize our policy for multirate, multicast networks in Section 4. We evaluate the fairness properties of these algorithms in Section 5. The conclusion is presented in Section 6.

2. NETWORK MODEL

We present a mathematical model for multicast networks. Network heterogeneity poses a challenge in the implementation of multicast communication. This is because different receivers of the same session receive traffic through different paths, and these paths often have widely varying congestion levels. Also, different receivers have different service limitations. For example, one may be a modem, whereas another may be an ethernet LAN. Thus, a single rate of transmission for all receivers tends to overwhelm the slow ones and starve the fast ones. This problem can be addressed if the receivers of the same application are allowed to receive information at different rates, and the session rate in a link is the maximum of that of the receivers downstream of the link. This mechanism is known as multirate multicast.

Multirate multicast can be attained by layered encoding. For this purpose, a source hierarchically encodes its signal in several layers. The lowest layer consists of the most important information and all receivers of the session should receive it. Receivers can subscribe to higher layers for successive refinement of reception quality at the expense of additional bandwidth. If the path to a receiver is congested, then it receives the lowest layer only, whereas a receiver subscribes to a large number of layers if its data path has a lot of bandwidth. Different receivers of the same session can receive different number of layers. The number of layers transmitted in a link is the maximum of the subscriptions of the receivers downstream.

A multicast session is identified by the pair (v, U), where v is the source node of the session and U is the group of intended destination nodes or receivers. We assume that the traffic from node v is transported across a predefined multicast tree to nodes in U and do not concern ourselves with the routing problem in this paper. Since different receivers of the same session can be allotted different bandwidth, we need to consider receiver rates instead of session rates. Let there be N sessions and M receivers. Rate allocation is an M-dimensional vector $(r_{11}, \ldots, r_{1m_1}, \ldots, r_{i1}, \ldots, r_{im_i}, \ldots, r_{N1}, \ldots, r_{Nm_N})$, where r_{ij} is the rate allocated to the jth receiver of the ith session. For simplicity, we will use a single index, henceforth. A rate allocation $(r_1, \ldots, r_M)$ is feasible, if the total bandwidth consumed by all sessions traversing a link does not exceed the capacity of the link. Bandwidth consumed by a session in a link is equal to the maximum of the bandwidth allocated to its receivers downstream of the link. Bandwidth consumed by a session in one link may be different from that in another link, as different receivers of the same session may have different bandwidth. Formally, a rate allocation $\vec{r} = (r_1, \ldots r_M)$ is a feasible rate allocation if $\sum_{i \in n(l)} \max_{j \in m(i,l)} r_j \leq C_l$ (capacity condition), where $n(l)$ denotes the set of sessions passing through link l, $m(k, l)$ denotes the set of receivers of session k downstream of link l and C_l denotes the capacity of link l. Figure 1 illustrates an example network with a few capacity and maximum rate constraints.

A feasible bandwidth allocation is maxmin fair if it is not possible to maintain feasibility and increase the rate of a receiver without decreasing that of any other receiver which has equal or lower rate. More formally, a feasible rate allocation $\vec{u}^1$ is maxmin fair if it satisfies the following property with respect to any other feasible rate allocation $\vec{u}^2$: if there exists i such that the ith component of $\vec{u}^2$ is strictly greater than that of $\vec{u}^1$ ($u_i^2 > u_i^1$), then there exists j such that the jth component of $\vec{u}^1$, u_j^1 is less than or equal to the ith component of $\vec{u}^1$, u_i^1 ($u_j^1 \leq u_i^1$) and the jth component of $\vec{u}^2$ (u_j^2) is strictly less than the jth component of $\vec{u}^1$ ($u_j^2 < u_j^1$). Refer to the network of figure 1 for an example of maxmin fair bandwidth allocation.

Now, we discuss the special case of a network with unicast sessions only. A unicast session has only one receiver. Every session can be characterized by a source destination pair and we need to consider session rates only. If there are N sessions then the rate allocation vector is an $N-$dimensional vector. The fairness objective is similar for unicast and multicast. The essential difference in the mathematical model is in the feasibility condition. For unicast networks the capacity condition is that for every link l, $\sum_{j \in n(l)} r_j \leq C_l$, where r_j is the rate of session j. Thus the capacity condition is linear as opposed to the nonlinear capacity condition for networks involving multicast sessions. This additional complication necessitates different resource allocation

approaches in unicast and multirate, multicast networks. We first focus on unicast networks, and later generalize our approach for multirate, multicast sessions.

We present the intuition behind our approach in the next section.

3. A STATELESS ALGORITHM FOR MAXMIN FAIRNESS IN UNICAST NETWORKS

In this section, we present a flow control scheme for attaining maxmin fairness in unicast networks. The algorithm does not need any per flow state in the intermediate nodes, and thus scales well with an increase in the number of sessions in the network. We first describe the intuition behind the policy.

3.1. Intuition behind our policy for unicast networks

We first describe the basic approach of the flow control policy of[4] for attaining the maxmin fair rate allocation. Incidentally, the basic philosophy is similar for other algorithms[5, 7] as well but the algorithm details differ. Every link offers equal bandwidth to all sessions traversing the link. The bandwidth utilized by a session equals the minimum offered in its path. For this purpose, a link computes a fair share, where fair share of a link equals the total bandwidth divided by the number of sessions. This fair share is offered to each session traversing the link. If a session can not utilize this fair share, it is saturated, otherwise it is unsaturated. Fair shares are updated to reflect the total bandwidth not utilized by saturated sessions divided by the number of unsaturated sessions. The saturation status and the saturation bandwidth of sessions change all the time. A link keeps track of both the quantities for each session. Hence, it maintains per flow states.

We follow the same approach without maintaining any per flow state at the routers. The basic observation is that a link needs the sum of the utilization of the saturated sessions and the number of unsaturated sessions. The link does not require the individual bandwidth utilization of any session. However, the scheduler uses the individual utilization and the fair shares to compute the saturation status of each session, and the saturation status of the sessions are used to compute their total utilization and the total number of unsaturated sessions. We compute these quantities using estimation and some additional information carried in the data packets, but without maintaining the saturation status or the individual bandwidth utilization of sessions in the links.

At each link, the total rate of arrival of packets of saturated sessions is considered an estimate for the total utilization of the saturated sessions. Note that this does not require per flow states at the individual routers. The storage burden is shifted to the source, which maintains the saturation status of the session in each link on its path. This is acceptable as the source node often handles only one session. The data packets need an additional overhead of 1 bit per link to indicate the saturation status of the respective link.

The link maintains the number of saturated sessions. The estimate is updated as the saturation status of sessions change. The source sends special control packets (rate packets) periodically, to obtain information about the saturation status of the session in the links. The link uses the information in these rate packets to update the number of saturated sessions. A rate packet contains the estimated rate of the session. The saturation status of a session can be estimated in a link by comparing the rate value in the rate packet and the fair share of the link. If the rate value is less than the fair share, then the session is not utilizing the offered bandwidth, and is thus congested elsewhere. It is considered to be saturated in the link. If the rate value exceeds the fair share, then the session is not congested elsewhere, and is considered unsaturated in the link. The rate value in the rate packet is subsequently decreased to the fair share so that the estimated rate is the minimum of the fair shares on the path of the session. However, the number of saturated sessions can not be updated based on the current determination of the saturation status only, as the current estimate carries no information about the previous saturation status, and this information is essential in updating the number of saturated sessions. The router can learn the earlier saturation status if the rate packet contains a saturation vector indicating the previous saturation status in the links. So the source adds a saturation vector in the rate packet. The routers use the saturation vector, rate value and fair share for updating the number of saturated sessions. Subsequently, the corresponding bit in the saturation vector is also updated to reflect the new saturation status of the session in the link. When the rate packet returns, the source stores the new saturation vector, and inserts it in the next rate packet.

Finally, the fair share computation uses the maximum utilization of the sessions in a link when all sessions are saturated in the link in the algorithm of.[4] The maximum rate of saturated sessions can be estimated periodically from the rate values in the rate packets for the saturated sessions. We discuss this in details later.

3.2. Algorithm description for unicast networks

We first describe the policy. Subsequently we will provide a pseudo-code and discuss the salient features of the policy.

Control Packet Exchange: Every source periodically sends forward rate packets downstream along the session path. When a receiver receives a forward rate packet, it returns it as a backward rate packet to the source. The rate packets infer the congestion condition along the path, and the source adjusts its transmission rate accordingly. Each rate packet contains: 1) a congestion bit to identify whether this session should increase its data transmission rate, 2) a rate field to specify the data transmission rate and 3) a saturation vector which contains the saturation status of the session in the links on its path. The source always sets the congestion bit to 0 to indicate no congestion, and the links update the congestion bit appropriately. The source sets the rate field of the first forward rate packet to a large rate value (ideally infinity). The saturation vector in the first rate packet indicates unsaturation for all the links.

Every link stores the fair bandwidth share also known as the link control parameter, the number of unsaturated sessions and an estimate of the maximum rate of the saturated sessions traversing the link. When a link receives a forward rate packet, it compares the rate value in the rate packet with its link control parameter. If the rate value is greater than or equal to its link control parameter, the session is considered unsaturated in this link. If the rate is less than its link control parameter, the session is considered saturated in the link. The link can detect a change in the saturation status from the saturation vector. Subsequently, the link updates the number of unsaturated sessions and the corresponding saturation status indicator in the saturation vector of the rate packet. Next, if the rate value is greater than or equal to the link control parameter, the link sets the congestion bit in the rate packet to 1 indicating that there is congestion in the path of the session, and the session rate should not be increased. Also, the link will reduce the rate value to the link control parameter. Then, the link updates the maximum rate estimate and finally forwards the forward rate packet downstream.

When a receiver gets a forward rate packet, it sends the packet upstream as a backward rate packet. Every node simply forwards a backward rate packet upstream. A source node stores its last received backward rate packet. It uses the contents of this backward rate packet to determine the contents of the next forward rate packet. If the congestion bit in the backward rate packet is 1, then there is congestion in the path, and the rate value in the forward rate packet is set equal to that in the backward rate packet. Otherwise, the rate value is a large number, typically infinity. The saturation vector of the forward rate packet is the same as that in the backward rate packet.

Data transmission: Source node regulates the rate of data packet transmission in accordance to the rate value in its last received backward rate packet. The header of every data packet contains the saturation status vector. We may use the unused bytes in the IP header for this purpose. Also, the header contains a pointer. Initially, the pointer points to the first bit in the saturation vector, and the links gradually advance this pointer. Every link maintains two separate queues of data packets, one for packets of saturated sessions, another for packets of unsaturated sessions. When a link receives a data packet, it infers the saturation status of the session from the corresponding bit in the saturation vector. The position in the saturation vector is indicated by the pointer. If the bit indicates saturation, then the data packet is placed in the saturation queue, otherwise it is placed in the unsaturation queue. The link periodically measures the total arrival rate of the saturated sessions. This is an estimate of the total bandwidth utilized by the saturated sessions.

Link Control Parameter Computation: Every link maintains an estimate of the total bandwidth utilized by the saturated sessions, the number of unsaturated sessions and an estimate of the maximum rate of the saturated sessions. The link control parameter update procedure can be described as follows. The residual capacity is the bandwidth not utilized by the saturated sessions, i.e., it is the difference between the link capacity and the bandwidth utilized by the saturated sessions. If there is at least one unsaturated session, then the link control parameter is the residual capacity divided by the number of unsaturated sessions. If all sessions are saturated, then link control parameter is the sum of the maximum rate of sessions and the residual link capacity. Every link estimates the maximum rate of the saturated sessions periodically and computes its link control parameter at regular intervals. We discuss the choice of the computation intervals later.

Estimation of the maximum rate of the saturated sessions: Every link maintains two estimates, new estimate and old estimate. At the beginning of each estimation interval, the new estimate is initialized as 0. Every time a forward rate packet of a saturated session arrives, the new estimate is updated as the maximum of the current value of the new estimate and the rate value in the forward rate packet. At the end of each interval, the old estimate is assigned equal to the new estimate. The maximum rate used to update the link control parameter is the maximum of the new and the old estimates.

3.3. Pseudocode

First we introduce some terminologies.
ρ_i is the rate of transmission of session i.

r	Rate value of rate packet	Rate Packet
u	Congestion bit of rate packet	
$\vec{q}$	Saturation vector of rate packet	

Any link will store the following.

ψ_l	Link control parameter	Link l record
κ_l	Number of sessions traversing link l	
γ_l	Number of unsaturated sessions traversing link l	
ν_l	Number of packets transmitted for saturated sessions in the current interval	
R_{l1}	New estimate of the maximum rate of the saturated sessions	
R_{l2}	Old estimate of the maximum rate of the saturated sessions	

|(&) indicates logical or (and) operation.

The pseudocode follows:
Source process for session i:
Let the previous backward rate packet have rate value r_b, congestion bit u_b, saturation vector $\vec{q}^b$.

1. Send traffic at the rate r_b.
2. Send forward rate packets periodically with parameters $r_f, u_f, \vec{q}^f$.
   If $u_b = 0$, $r_f = \infty$, else $r_f = r_b$.
   $u_f = 0$, $\vec{q}^f = \vec{q}^b$.

Process at Link l :

1. When session i forward rate packet$(r, u, \vec{q})$ is received:

(a) If $(r < \psi_l)$ & $(\vec{q}_l = 0)$ or
$(r \geq \psi_l)$ & $(\vec{q}_l = 1)$, (If saturation status changes.)
call **estimate-link-control-parameter**$(\tau_l, \gamma_l, \nu_l, C_l)$

(b) If $r < \psi_l$,
- if $\vec{q}_l = 0$, $\gamma_l = \gamma_l - 1$ (number of unsaturated sessions decreasing) and $\vec{q}_l = 1$
- $R_{1l} = \max(R_{1l}, r)$

(c) if $r \geq \psi_l$,
- $r = \psi_l$
- $u = 1$
- if $\vec{q}_l = 1$, $\gamma_l = \gamma_l + 1$ (number of unsaturated sessions increasing)and
- $\vec{q}_l = 0$

(d) **generate-forward-rate-packet**$(r, u, \vec{q})$ and transmit it on link l.

2. When session i backward rate packet$(r, u, \vec{q})$ is received, link l sends the backward packet upstream.
3. Periodically calls **estimate-link-control-parameter**$(\tau_l, \gamma_l, \nu_l, C_l)$
4. Periodically calls **update-maximum-saturated-rate**(R_{1l}, R_{2l}) to update the estimate of the maximum rate of the saturated sessions in the link

Process at Receiver s of session i :
When a forward rate packet$(r, u, \vec{q})$ is received, **generate-backward-rate-packet**$(r, u, \vec{q})$ and send it towards the source of session i.

Subroutine **generate-forward-rate-packet**$(r, u, \vec{q})$ (**generate-feedback-rate-packet**$(r, u, \vec{q})$) simply generates a forward(backward) rate packet with rate value equal to r and the congestion bit equal to u and saturation vector equal to $\vec{q}$.

estimate-link-control-parameter$(\tau_l, \gamma_l, \nu_l, C_l)$ is described below.

1. if $\gamma_l = \kappa_l$, $\psi_l = \frac{C_l}{\gamma_l}$
2. else if $\gamma_l > 0$, $\psi_l = \frac{C_l - \frac{\nu_l}{\text{interval size}}}{\gamma_l}$
3. else $\psi_l \rightarrow \max(R_{1l}, R_{2l}) + C_l - \frac{\nu_l}{\text{interval size}}$
4. $\nu_l = 0$

Routine **estimate-link-control-parameter**$(\tau_l, \gamma_l, \nu_l, C_l)$ is called by a link, if the saturation status changes, or if the link control parameter has not been updated for a pre-specified period of time.

Routine **update-maximum-saturated-rate**(R_{1l}, R_{2l}) is used to update the estimate of the maximum rate of saturated sessions. The operation is described below.

1. $R_{2l} = R_{1l}$, (old maximum rate estimate is updated)
2. $R_{1l} = 0$, (new maximum rate estimate is reduced to 0 at the start of a fresh estimation cycle)

3.4. Discussion

We discuss some salient features of this policy in this subsection.

We first examine the convergence properties. Intuitively, the output of this algorithm should converge to the maxmin fair allocation. This is because in our policy every link offers equal bandwidth to all sessions initially. If a session can not utilize the offered bandwidth, then the residual bandwidth is distributed among other sessions. Observe that the link control parameter is the offered bandwidth in any link, and the estimated rate of a session is the minimum of the link control parameters on the path of the session. A session is considered saturated in

a link if this estimated bandwidth is less than the link control parameter. Initially, all sessions are unsaturated and the link control parameter is the link capacity divided by the number of sessions. Link control parameter is updated as the residual capacity divided by the number of unsaturated sessions (residual capacity is the bandwidth not utilized by the saturated sessions). This update redistributes the unutilized bandwidth among the unsaturated sessions. This is the basic principle for well established flow control algorithms which attain maxmin fairness.[1, 4] Thus, intuitively our policy should converge to the maxmin fair rates. Experimental results support this intuition. Our contribution is to implement this philosophy using a message exchange sequence and an estimation based approach which eliminate the flow specific states at the nodes. This does not alter the convergence properties in general. However, our policy exhibits oscillatory behavior for certain choices of the parameters. We discuss this later.

We estimate the bandwidth not utilized by the saturated sessions through estimation, and estimation errors may be caused by transients. Also, initially the sessions transmit at high bandwidth and the transmission rates are not feasible. This leads to transients which cause estimation error. But, estimation error is present for any stateless rate computation scheme(e.g.,[2]), because any such scheme typically computes the utilized bandwidth in a link from the arrival rates, and the arrival rates in a link depend on the scheduling and the transients in all previous links. However, typically links have additional bandwidth to absorb the transients. For example, if a link l actually has C_l bandwidth then it aims at an utilization of αC_l, where $\alpha < 1$. In our case, link control parameter computations assume that the capacity of link l is αC_l for all links l, $\alpha < 1$. The additional bandwidth $(1-\alpha)C_l$ is used to absorb the initial transients. Thus, the convergence is not sensitive to the transients and the exact scheduling policy. We found that our policy converges to the maxmin fair rates for an utilization factor as high as 90% with simple FIFO scheduling. We also found that the convergence is not sensitive to the size of the estimation interval. However, as expected very short intervals lead to transients, and very large intervals slow response to congestion, and the policy exhibits oscillatory behavior for both the extremes. Convergence properties are good for all reasonable interval sizes, e.g., order of a few round trip times.

We refresh the maximum rate estimate periodically. As a result the maximum rate estimate at a particular time may not be the right one. Besides, very short estimation intervals lead to incorrect estimates as rate packets of all sessions may not be encountered in one interval. Very long intervals lead to slow response to change in the maximum rates. We found good convergence properties for moderate estimation intervals, e.g., intervals of the order of a few round trip times.

We experimentally observed convergence for a wide range of network topologies and link capacities. However, as discussed, some pathological choice of the estimation intervals lead to oscillation. But, the only known flow control policies which guarantee convergence for all parameter choices, e.g.,[1, 4, 8] need per flow states at the intermediate routers. From an implementation point of view, it is better to deploy a stateless algorithm which converges to the maxmin fair output in general, rather than an algorithm which guarantees convergence, but requires per flow states.

Now, we examine the storage and the processing requirements at the links. A link originating from an intermediate node, just needs to store the number of unsaturated sessions, link control parameter, and the maximum rate of the saturated sessions. Thus intermediate nodes does not need any per flow state. A source node needs to store the previous backward rate packet of the session, and thus needs flow specific states for the session. Source nodes are at the edge of a network and only a few sessions originate from a source node, and hence it is acceptable to have a source node store specific states of the sessions originating from the node. Processing wise, a link needs to periodically compute the fair share, and estimate the maximum rate of the saturated sessions. Both of these are computationally simple. A link also needs to modify the contents of the forward rate packets suitably. The necessary computation just involves computing the minimum of two real numbers and again comparing two real numbers. Neither operation is computationally intensive. However, modifying the contents may consume some more time, but the link needs to modify the contents of only the control packets, and not the data packets (a link may need to advance a pointer in a data packet header, but the IP module anyway alters the packet header before transmitting the packet in a new link). Control packets are sent periodically, and this period can be adjusted suitably. The tradeoff is that if the control packets are sent after long time intervals then the convergence is slow. Thus there is a tradeoff between convergence speed and link processing complexity.

Now, we examine the message exchange overhead. Additional control packets contribute to the message exchange overhead. However, control packet exchange interval can be adjusted suitably. Thus, there is a tradeoff between convergence speed and the message exchange overhead as well. There is a small amount of additional overhead in the data packet headers. The data packet header must contain the saturation vector, but the saturation vector has only one bit per link.

Summarizing, the algorithm is computationally simple, distributed, asynchronous and stateless. The algorithm need not restart every time there is a topology change, e.g., a session joins or leaves. Thus it is suitable for deployment in a dynamic scenario like the current internet.

We generalize our policy to address multirate, multicast sessions in the next section.

4. A REDUCED STATE FAIR ALLOCATION POLICY FOR MULTIRATE MULTICAST NETWORK

We first describe the distinctive challenges of multirate, multicast networks. We briefly argue that multirate multicast needs per flow states at the junction nodes. Multirate, multicast is realized by a layered transmission scheme. Source encodes its signal into several layers and different layers are transmitted as different multicast groups. Higher the number of layers received, better is the reception quality. Thus receivers with higher bandwidth subscribe to a higher number of layers for better reception quality. A junction node needs to maintain a list of downstream links for each layer in its routing table. There are several layers for each session and the forwarding is different for each. Hence flow specific states are required at each junction node. However, intermediate nodes can aggregate the flow states.

We highlight another important difference between resource allocation of multicast and unicast networks. Multicast trees have several receivers and hence several paths. The receivers receive service at different bandwidth. The bandwidth allocated to a session in a link should depend on the congestion in all the paths involving the link, and in particular on the bandwidth allocation of the least congested path. Maxmin fairness can be attained in a multirate, multicast network by initially allocating equal bandwidth to every session traversing a link.[8] If a session does not utilize the offered bandwidth because of constraint elsewhere, then the residual bandwidth is distributed among the unconstrained sessions. This is similar to the unicast case, and can be attained by offering every session a bandwidth equal to the link control parameter where the link control parameter is updated as the residual capacity per unsaturated session. The principal difference is in determining the saturation status and the bandwidth of a saturated session. A session will not utilize the full bandwidth offered in a link if all the session paths involving the link are congested, and it is only then that a session is considered saturated in a link. The congestion status of the least congested path is required. We acquire this information as follows.

The source sends the control packets downstream to collect congestion information. The junction nodes multicast the control packets to all downstream links of the multicast tree. The receivers return the control packets towards the source. The junction nodes aggregate all the backward rate packets and send only one backward rate packet upstream. The aggregated backward rate packets reflect the congestion information of the least congested path traversing the junction node only. The source and the junction nodes transmit data packets in accordance with the rate information in the returning control packets, and also infer the congestion status of the session in the downstream links from these control packets. The details are described next.

4.1. Algorithm description

We first describe the policy, and subsequently we will provide a pseudo-code and discuss the salient features of the policy. We focus on the differences with the unicast case.

We now define a junction node formally. A junction node is an intermediate node of a multicast tree (a non-terminal node) where the tree forks into branches. A junction link of a session is a link which originates from a junction node and is on the path of the session.

Control Packet Exchange: The content of the forward rate packet is similar to that for the unicast case, with one important difference. The saturation vector reflects the congestion status of the session in links till the next junction node, and contains one bit for each of these links.

Now we consider the processing of forward rate packets at the links. The processing of rate packets and storage at the non-junction links are the same as that for the unicast case. The storage at these non-juction links is also the same. There is additional storage and processing at the junction nodes and junction links though. Every junction link stores the previous backward rate packet for the session. When a forward rate packet arrives at a junction node, the junction node stores it till the corresponding backward rate packets arrive. In addition, it sends a new forward rate packet in each of its outgoing junction links. The saturation vector in each forward rate packet is set equal to that in the backward rate packet stored in the corresponding link. The rate value in the new forward rate packet is determined as the minimum of the link control parameter, rate value in the incoming forward rate packet and the rate value in the previous backward rate packet.

When a receiver gets a forward rate packet, it sends it upstream as a backward rate packet. A nonjunction node simply forwards a backward rate packet upstream. However, a junction node needs further processing with the backward rate packets. A junction node aggregates all the backward rate packets that arrive in the different outgoing links, and sends one backward rate packet upstream. The contents of the aggregated backward rate packet are determined as follows. The rate value in this backward rate packet is the maximum of the rates in the incoming backward rate packets. The congestion bit is set to 1 if the incoming forward rate packet has congestion bit 1 or if all the incoming backward rate packets have congestion bit 1. Thus the contents of an aggregated backward rate packet reflect congestion only if all downstream paths are congested. The contents also reflect the offered bandwidth in the least congested path.

A source node stores its last received backward rate packet. It uses the contents of this backward rate packet in determining the contents of the next forward rate packet as in a unicast network.

Data Transmission: The source node incorporates the saturation vector of the stored backward rate packet in the headers of the data packets. The links use this saturation vector to estimate the total bandwidth of the saturated sessions. We first describe the transmission procedure in a junction link. Such a junction link stores a backward rate packet for the session. The junction node also stores the incoming forward rate packet. The rate of the session in the link is the minimum of the rate values in the two rate packets, and need not be estimated from the data packets. Also, the link knows the saturation status of the session in the link from the content of the saturation vector in the backward rate packet. Thus, the link does not perform any estimation for the session. It may estimate the total bandwidth consumed by other saturated sessions for which it is not a junction link. Also a junction node selectively forwards data packets downstream at a rate which equals the minimum of the rate values in the stored forward and backward rate packets. Finally, the saturation vector in a data packet header is set equal to that in the stored backward rate packet.

4.2. Pseudocode for multirate multicast networks

For each non-junction node and link, the algorithm is the same as that for the unicast networks.

The storage for each junction link and node follows:

If node n is the junction node for session i, it stores the last **forward rate packet** $(r_f^i,\ u_f^i,\ \vec{q}_f^i)$ for each session i, where r_f^i is the rate value, u_f^i is the congestion bit and $\vec{q}_f^i$ is the saturation vector:
Every junction link l of session i stores the previous **backward rate packet**,$(r_b^i,\ u_b^i,\ \vec{q}_b^i)$ for each session i, where r_b^i is the rate value, u_b^i is the congestion bit and $\vec{q}_b^i$ is the saturation vector. In addition, such a link stores the link control parameter.

Process at junction link l :

1. Junction link uses some local variables z_{il1} and z_{il2}. When the junction link receives a session i forward rate packet$(r_f^i, u_f^i, \vec{q}_f^i)$:
   (a) If $u_b^i = 0$, $z_{il1} = r_f^i$, else $z_{il1} = \min(r_b^i, r_f^i)$,

(b) $z_{il2} = 0$,

(c) If $(z_{il1} < \psi_l)$ & $(\vec{q}^i_{lb} = 0)$ or
$(z_{il1} \geq \psi_l)$ & $(\vec{q}^i_{lb} = 1)$, (If saturation status changes.)
call **estimate-link-control-parameter**$(\tau_l, \gamma_l, \nu_l, C_l)$

(d) If $z_{il1} < \psi_l$,
- if $\vec{q}^i_{lb} = 0$, $\gamma_l = \gamma_l - 1$ (number of unsaturated sessions decreasing) and $\vec{q}^i_{lb} = 1$
- $R_{1l} = \max(z_{il1}, R_{1l})$

(e) if $z_{il1} \geq \psi_l$,
- $z_{il1} = \psi_l$
- $z_{il2} = 1$
- if $\vec{q}^i_{lb} = 1$, $\gamma_l = \gamma_l + 1$ (number of unsaturated sessions increasing)and
- $\vec{q}^i_{lb} = 0$

(f) **generate-forward-rate-packet**$(z_{il1}, z_{il2}, \vec{q}^i_b)$ and transmit it on link l.

2. When session i backward rate packet$(r^i_b, u^i_b, \vec{q}^i_b)$ reaches link l, it stores it
3. Periodically calls **estimate-link-control-parameter**$(\tau_l, \gamma_l, \nu_l, C_l)$
4. Periodically calls **update-maximum-saturated-rate**(R_{1l}, R_{2l}) to update the estimate of the maximum rate of the saturated sessions in the link

Process at junction node n :

1. When the junction node receives a session i forward rate packet$(r^i_f, u^i_f, \vec{q}^i_f)$, it stores it
2. When all the junction links originating from a junction node receives a session i backward rate packet, the junction node generates a **backward-rate-packet**$(r^i_b, u^i_b, \vec{q}^i_b)$, and sends it towards the source, where
   - r^i_b is the maximum of the rates of all these backward rate packets
   - $u^i_b = 1$, if $u^i_f = 1$ or the congestion bits of all the backward rate packets are 1. Otherwise, $u^i_b = 0$.
   - $\vec{q}^i_b = \vec{q}^i_f$.

Let n be the origin node of link l and let link l be added to the path of a session i. Session initiation can cause addition of links on its path. A link may also be added to the path of an existing session on account of some new receiver joins. If n is the origin node,

1. $\kappa_l = \kappa_l + 1, \qquad \gamma_l = \gamma_l + 1,$
2. If session i was not traversing node n, prior to addition of link l, $\tau^n_i = \{l\}$, else $\tau^n_i = \tau^n_i \cup \{l\}$.

Note that we do not need to initialize other parameters because of the operation of the algorithm.

Let n be the origin node of link l and let link l be removed from the path of a session i. Session exit can cause removal of all links on its path. A link may also be removed from the path of a continuing session, if all receivers downstream leave. When a session is removed from a link, the link knows about the removal from a control packet from the nearest upstream junction node or sender, and the junction node or sender knows the saturation status of the session in the links of the branch to be removed and puts the saturation status vector for this branch in the header of this control packet.

1. $\tau^n_i = \tau^n_i \setminus \{l\}$,
2. $\kappa_l = \kappa_l - 1$,
3. If $\vec{q}_l = 0$, $\gamma_l = \gamma_l - 1$.

A junction node forwards a data packet in a link l only at the rate b_{il} for session i in link l. Also, a junction node replaces the existing $\vec{q}$ in the source header by stored $\vec{q}^i$ for the link.

4.3. Discussion

We discuss some salient features of our policy in this subsection.

An algorithm for computing maxmin fair rates in multirate, multicast networks has been proposed in.[8] We follow the same philosophy as that in.[8] Namely the network first tries to equalize the bandwidth of all sessions in each link. If a session can not utilize the offered bandwidth in a link, then the residual bandwidth is distributed among the unsaturated sessions. The key point is that a session can not utilize the offered bandwidth in a link if all the session paths involving the link are congested. The basic approach is provably fair. However, the basic approach uses flow specific states at all links to attain its objective. We use estimation and additional information in the data packets to obtain the necessary information without maintaining per flow states. Thus, intuitively the policy should converge to the maxmin fair rates. Experimental results support this intuition.

The discussion regarding the choice of the estimation intervals for the unicast network applies in the multicast case as well. We describe the storage at each node now. The storage at a non-junction link is exactly the same as that for the unicast case. Hence, these links do not maintain per flow states. However, a junction link maintains flow specific states for the session. As we discussed before, any multirate flow control scheme needs per flow states at the junction nodes, and thus our policy uses the minimum possible per flow states.

The processing requirements are similar to the unicast case. The source and the junction nodes incorporate the saturation vector in the headers of the data packets.

Finally, the control packet aggregation prevents the control packet implosion problem, as the overhead in a link does not depend on the number of receivers downstream.

5. PERFORMANCE EVALUATION

We have studied the performance of our algorithm in a large number of different topologies. The output converged to the maxmin fair rates in all cases. We present some of the results below. We assume FIFO scheduling and an utilization factor of 90% all through. Thus, the link control parameters are computed assuming that the link capacity is 90% of its actual value. We deliberately used a high utilization factor, as estimation error is likely to be high at high loads, and the experiments indicate that the system converges at high loads as well. We selected an estimation interval of size 1500 for both link control parameter and maximum rate estimation. This is of the oder of $5 - 6$ round trip travel times of the control packets. The parameters were selected on an ad-hoc basis in this case. We found that the convergence is not sensitive to the size of these intervals.

We first present our simulation results for a 10-node network. We first study the network with unicast sessions only(Figure 2(a)). We assume negligible propagation delay for each link. The delay in convergence is mainly because of the time needed by the control packets to reach the receivers. The control packets need some time to reach the receivers as the links are loaded on account of the high utilization factor. Most of the sessions get their maxmin fair rates by the time one or two control packets complete their round trip journey. After four round trip travel of the control packets, the rates of all the sessions converge to the maxmin fair values. Figure 3(a) shows the convergence for two sessions whose rates converge relatively slowly. We studied the discrepancy from the maxmin fair rate at any time. If maxmin fair rate of a session 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 session s is $|1 - \frac{r_s^c(t)}{r_s^m(t)}|$ at time t. Figure 3(b) plots the maximum of the relative errors of the sessions. This figure indicates that the attained rates of all the sessions converge to the respective maxmin fair rates.

Next, we consider the same network with multicast sessions (Figure 2(b)). All sessions achieve their fair rates in a few round trip travels of the control packets. We show the convergences for two relatively slow receivers in Figure 4(a). Again, we study the discrepancy between the attained and the maxmin fair rates of the receivers at any time. Figure 4(b) plots the maximum of the relative errors of the receivers as a function of time. The experimental result indicates that the attained rates of all the receivers converge to the maxmin fair rates.

Next, we show our results for a large random network. The network has 100 nodes, 603 links, 10 multicast sessions and 84 receivers. Nodes are points on a $10X10$ grid. There exists an edge between any two nodes with

a probability (p) that depends on the euclidean distance between the nodes (d) ($p = \exp(\beta(1-d))$), where β is the decay constant. We assumed $\beta = 2$. We adopted this edge probability model because the distant nodes are less likely to have an edge between them. The propagation delay between the two nodes is equal to the euclidean distance between them. Source and the receivers of every session have been selected randomly. The session route consists of the shortest paths between the source and the destinations. Most of the receivers obtain their maxmin fair rates in the first few round trip travel times of the control packets. However, the control packets take longer to travel the network because of the propagation delay and the processing delay in the long source receiver paths. Thus convergence is slower than that in Figure 4. Figure 5(a) plots the attained rates of a few sample receivers. Figure 5(b) plots the maximum error of the receiver as a function of time. The maximum is taken over all currently active receivers. Both these figures indicate that the rates of all receivers converge to the respective maxmin fair rates.

We have studied the case when receivers join or leave the multicast sessions dynamically. We show results for the 10−node network of Figure 2(a) in Figure 6. All the receiver rates have converged to the maxmin fair rates by time $t = 5000$. Receiver 7 joins session 3 at time $t = 10,680$ units. Maximum error is initially high, as the new receiver starts from a low rate initially. Gradually, all receivers attain their maxmin fair rates and the maximum error decreases to 0. Receiver 9 leaves session 5 at $t = 17889$. Initially, maximum error is high, but again it drops to 0. Finally receiver 9 leaves session 4 at $t = 26,706$. The attained rates again converge to the maxmin fair rates.

6. CONCLUSION

We present a new algorithm for attaining maxmin fairness of rate allocation in unicast networks. This algorithm does not need per flow state in the intermediate nodes. Next we generalize our algorithm to address maxmin fair rate allocation in multirate multicast networks, which only requires minimum possible per flow states. The algorithms are scalable and practical for implementation. Simulation results exhibit rapid convergence to the maxmin fair rates even in presence of network topology changes.

REFERENCES

1. S. Abraham and A. Kumar: A Stochastic Approximation Approach for Max-Min Fair Adaptive Rate Control of ABR Sessions with MCRs. *Proceedings of IEEE INFOCOM '99,* New York, March 99.
2. S. Athuraliya, S. H. Low, and D. Lapsley: "Random Early Marking," *Proceedings of IEEE GLOBECOM 99,* Rio De Janeiro, December' 99.
3. T. Ballardie, P. Francis, and J. Crowcroft. Core based trees: an architecture for scalable inter-domain multicast routing, *Proceedings of ACM SIGCOMM,* Sept. 1993, pp. 85-95
4. A. Charny: An Algorithm for Rate Allocation in a Packet Switching Network with Feedback. *M.S. thesis,* Massachusetts Institute of Technology, May 1994
5. Y. T. Hou, H. H. Y. Tzeng and S. S. Panwar. A Generalized Max-Min Rate Allocation Policy and its Distributed Implementation Using the ABR Flow Control Mechanism, *Proceedings of IEEE Infocom' 98*, San Francisco, CA, March 1998
6. D. Rubenstein, J. Kurose D. Towsley: The Impact of Multicast Layering on Network Fairness *Proceedings of ACM SIGCOMM'99,* Cambridge, MA, September, 1999
7. J. Ros, W. K. Tsai: A General Theory of Constrained Max-Min Rate Allocation for Multicast Networks *IEEE ICON'00,* Singapore, 2000
8. S. Sarkar, L. Tassiulas: Distributed Algorithms for Computation of Fair Rates in Multirate Multicast Trees *Proceedings of IEEE INFOCOM'00,* Tel Aviv, Israel, 2000

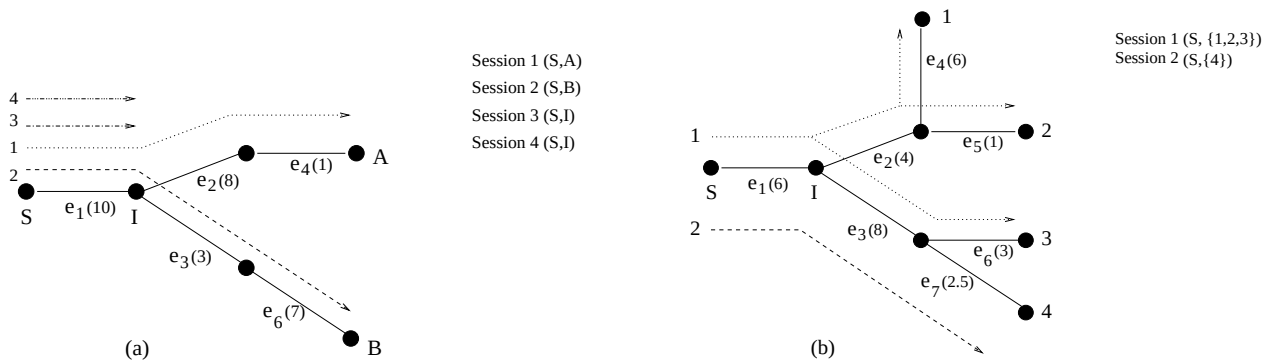

Figure 1. A session is represented by a source destination set pair in the two figures. The numbers in brackets, () denote the capacities of the respective links. For example, e_1 has capacity 10 units in Figure (a).
We consider Figure (a) first. There are 4 sessions. The rate allocation vector is (r_1, r_2, r_3, r_4). The capacity constraint for link e_1 is $r_1 + r_2 + r_3 + r_4 \leq 10$. The maxmin fair bandwidth allocation is $(1, 2.33, 2.33, 2.33)$. Session 1 is constrained in link e_4. The remaining bandwidth in link e_1 is split between sessions $2, 3$ and 4 so as to equalize their bandwidth.
Now we consider Figure (b). The capacity constraint for link e_1 is $\max(r_1, r_2, r_3) + r_4 \leq 6$ and that for link e_3 is $r_3 + r_4 \leq 8$. The maxmin fair bandwidth allocation is $(3.5, 1, 3, 2.5)$ for receivers $1, 2, 3, 4$ respectively. Session 2 is constrained in link e_7. Session 1 receives the remaining bandwidth of 3.5 units in link e_1. Receivers 2 and 3 are constrained in links e_5 and e_6 respectively. However, receiver 1 can receive the full offered bandwidth of 3.5 units.

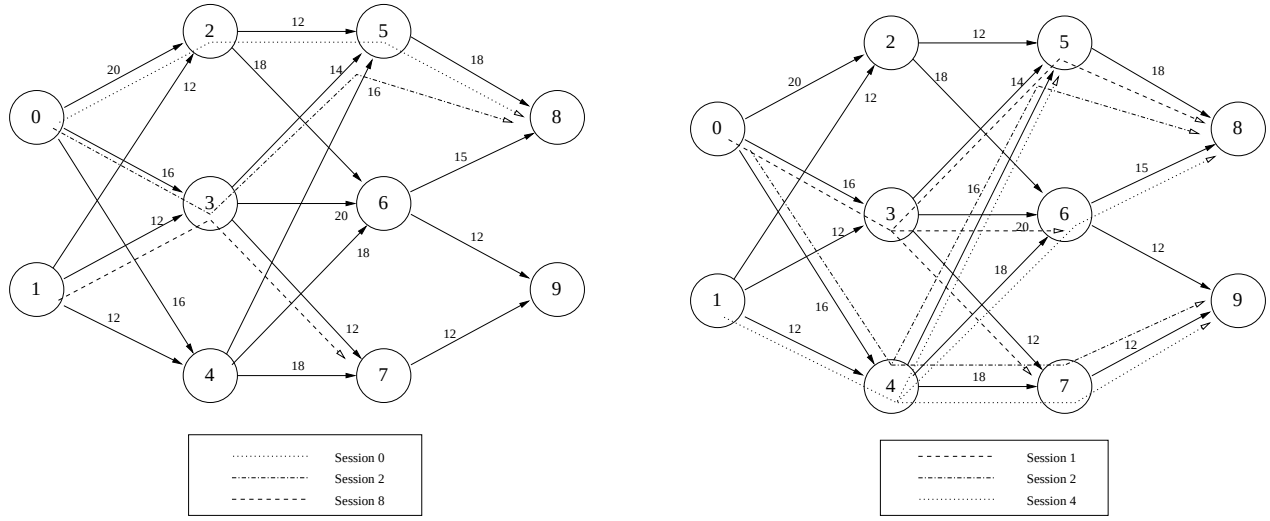

Figure 2. Figure (a) shows a unicast network of 10 nodes. The numbers next to the links denote the capacities of the respective links. The network has 14 sessions. We show only 3 sessions in the figure for convenience. The average length of a session path is 2.7 edges. On an average 2.1 sessions traverse each link.
Figure (b) shows a multicast network of 10 nodes. The network has 6 sessions and 14 receivers in all. We show only 3 sessions for convenience. The average number of receivers per session is 2.33. The average length of a session tree is 5 edges. On an average, 1.6 sessions traverse each link.

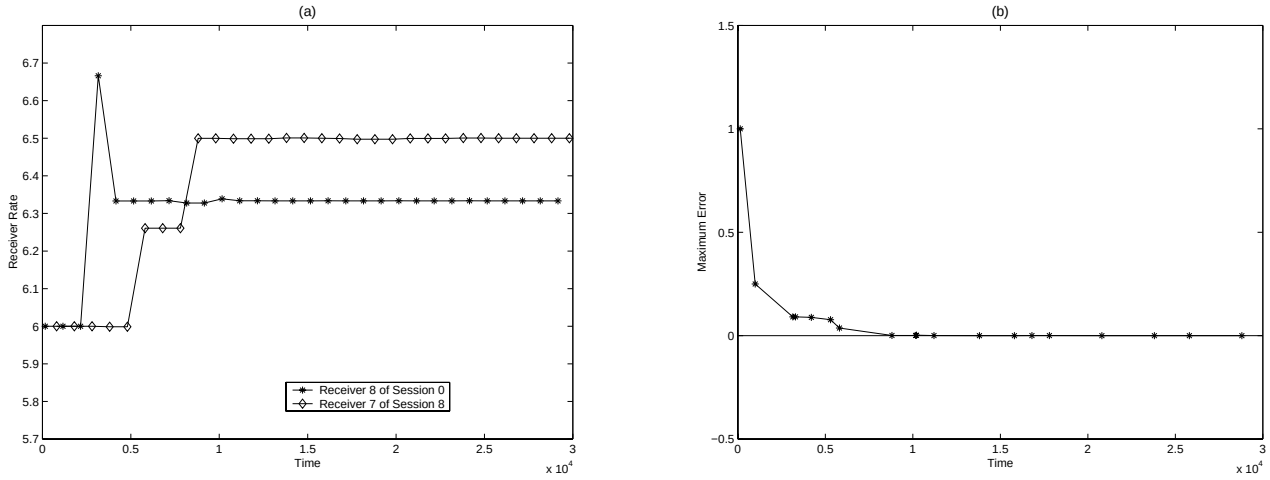

Figure 3. Figure(a) shows the attained rates of sessions 0 and 8 shown in Figure 2(a). The attained rates converge to the maxmin fair rates (6.333 for session 0, 6.5 for session 8). These sessions have slower convergence than others in the network of Figure 2(a).
Figure(b) shows that the maximum relative error rapidly decreases to 0. This indicates that the rates of all sessions converge to the maxmin fair rates.

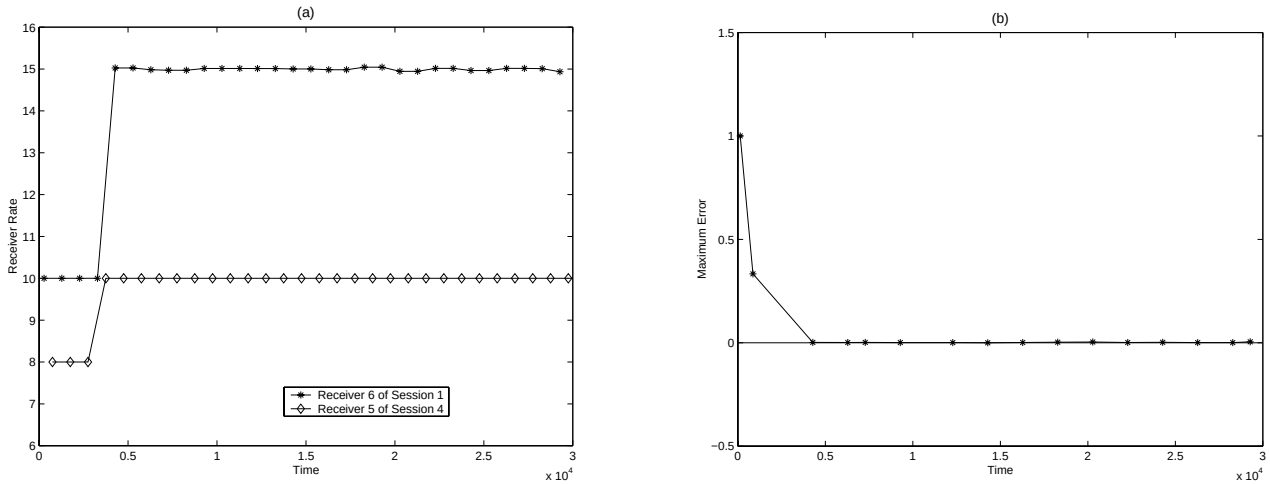

Figure 4. Figure(a) shows the attained rates of receiver 6 of session 1 and receiver 5 of session 8 shown in Figure 2(b). The attained rates converge to the maxmin fair rates (15 and 10 for the two respectively). These receivers have slower convergence than others in the network of Figure 2(b).
Figure(b) shows that the maximum relative error rapidly decreases to 0. This indicates that the rates of all receivers converge to the maxmin fair rates.

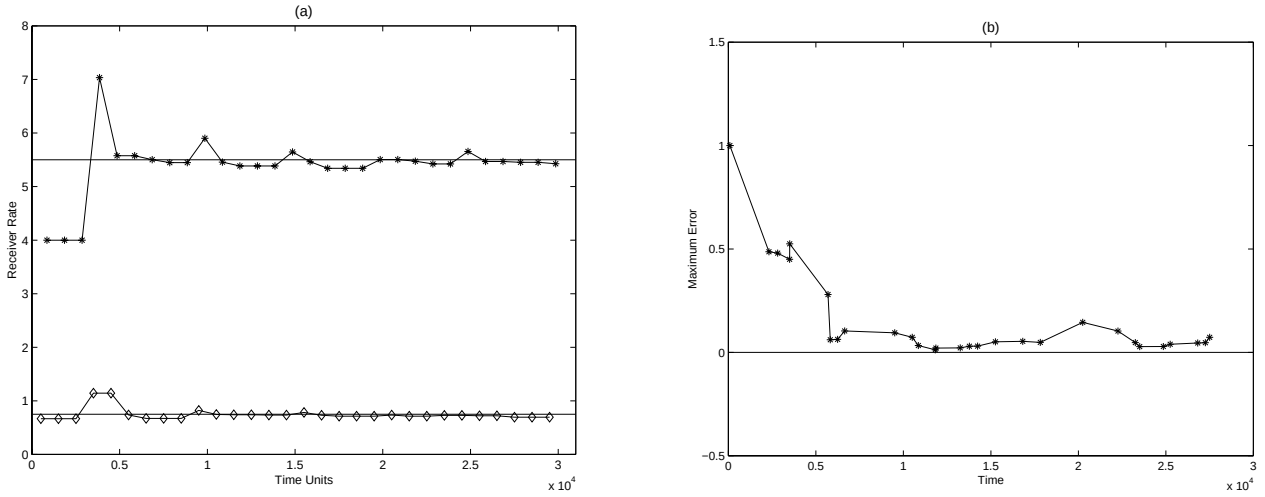

Figure 5. We show the performance of our policy in a randomly generated multicast network with 100 nodes and 603 links. The average number of receivers per session is 8.4. The average length of a session tree is 28 edges. On an average, 0.45 sessions traverse each link. Figure(a) shows the attained rates of two sample receivers in a random network. The attained rates converge to the maxmin fair rates (5.5 and 0.75 for the two respectively).
Figure(b) shows that the maximum relative error rapidly decreases to 0. This indicates that the rates of all receivers converge to the maxmin fair rates.

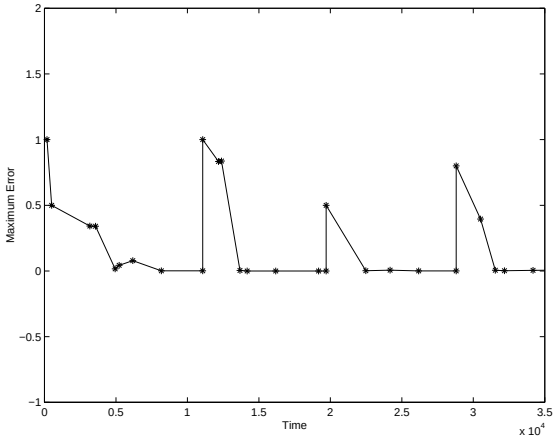

Figure 6. We plot the maximum error in the network of Figure 2(b). We allow receivers to join and leave the network dynamically. The figure indicates that the rates converge to the maxmin fair rates after every join and leave.