A Scalable, Low-Overhead Rate Control Algorithm for Multirate Multicast Sessions

Koushik Kar, Saswati Sarkar, Leandros Tassiulas

Abstract— **In multirate multicasting, different users (receivers) within the same multicast group can receive service at different rates, depending on the user requirements and the network congestion level. Compared to unirate multicasting, this provides more flexibility to the user, and allows more efficient usage of the network resources. In this paper, we address the rate control problem for multirate multicast sessions, with the objective of maximizing the total receiver utility. This aggregate utility maximization problem not only takes into account the heterogeneity in user requirements, but also provides a unified framework for diverse fairness objectives. We propose an algorithm for this problem and show, through analysis and simulation, that it converges to the optimal rates. In spite of the non-separability of the problem, the solution that we develop is completely decentralized, scalable and does not require the network to know the receiver utilities. The algorithm requires very simple computations both for the user and the network, and also has very low overhead of network congestion feedback.**

Keywords— **Flow control, Multirate multicast, Layered multicast, Optimization.**

I. Introduction

In conventional or unirate multicasting, all receivers of the same multicast group receive service at the same rate. However, in general, different receivers belonging to the same multicast group can have widely different characteristics. Thus a single rate of transmission per multicast group is likely to overwhelm the slow receivers and starve the fast ones. Multirate transmission, where the receivers of the same multicast group can receive data at different rates, can be used to accommodate these diverse requirements. Naturally, multirate multicasting is the preferred mode of data delivery for many real-time applications, including teleconferencing and audio/video broadcasting. Multirate transmission allows a receiver to receive data at a rate that is commensurate with its requirements and capabilities, and also with the capacity of the path leading to it from the source. One way of achieving multirate transmission is through hierarchical encoding of real time signals. In this approach, a signal is encoded into a number of layers that can be incrementally combined to provide progressive refinement. This layered transmission scheme can be used for both audio and video transmissions over the internet [7], [29], and has potentials for use in ATM networks as well [15]. In the case of the internet, each layer can be transmitted as a separate multicast group and receivers can adapt to congestion by joining and leaving these groups (see [18] and [20] for internet protocols for adding and dropping layers). Note that in multirate multicasting, there is no unique multicast session rate, and one needs to consider receiver rates separately. Also note that in this case, the transmission rate of a multicast session (multicast group) on a link needs to be equal to the maximum of the rates of all receivers downstream of that link.

Compared to unirate multicasting, multirate multicasting allows more efficient use of the network resources. For efficient use of the network, an effective rate control strategy is necessary. The rate control algorithm should ensure that the traffic offered to a network by different traffic sources remain within the limits that the network can carry. Moreover, it should also ensure that the network resources are shared by the competing flows in some fair manner. It may therefore be desirable that the rate control algorithm would steer the network towards a point where some measure of global fairness is maximized.

There can be many acceptable definitions of fairness, some well-known ones being max-min fairness [5], proportional fairness [13]. Fairness definitions can be generalized in a nice way by using utilities. Utility of a user is a function connecting the bandwidth given to the user with the "value" associated with the bandwidth (note that throughout the paper, the terms "user" and "receiver" are used synonymously). The utility could be some measure of, say, the perceived quality of audio/video, the user satisfaction, or even the amount paid by the user for the bandwidth allotted to it. In this paper, we try to design the rate control algorithms such that they maximize the sum of the utilities over all receivers, subject to the link capacity constraints. This objective was proposed recently by Kelly [13]. It is easy to see that various fairness objectives can be realized within this utility maximization framework for different choices of the utility functions (see [19]). Note that in our problem, the utility functions can be different for different users (receivers). Thus this framework allows us to differentiate among receivers on the basis of their requirements and/or revenues. This is important, since receivers could have heterogeneous requirements, and the same amount of bandwidth could be valued differently by different receivers.

Recently, there has been a considerable interest in the problem of fair allocation of resources for multirate multicast sessions. However, most of the work in this area is concerned only with the notion of max-min fairness (see [23], [25], [26], [27], [9]). Although there has been a lot of research on the utility maximization problem for unicast case [14], [2], [17], [28], [16], [10], the multirate multicast case has not received significant attention. It is worth noting here that certain factors make the multirate multicast problem significantly different and considerably more complex than its unicast version. For instance, the problem in the multirate multicast case is non-separable and non-differentiable, unlike the unicast case (we discuss more on this in the subsequent sections). The multirate multicast utility maximization problem is addressed in [11]. Here, the authors propose distributed algorithms for this problem; their approach is based on dual methods. In this paper, we take a different approach, and derive a primal algorithm based on non-

K. Kar and L. Tassiulas are with the ECE Department, University of Maryland, College Park, MD 20742, USA (email: koushik@eng.umd.edu; leandros@eng.umd.edu).

S. Sarkar is with the EE Department, University of Pennsylvania, Philadelphia, PA 19104, USA (email: swati@ee.upenn.edu).

differentiable optimization methods. The algorithm that we propose is distributed, scalable, and does not require the network to know the receiver utilities. Also, both the user and the network (link/node) sub-algorithms are extremely simple, and the overhead of the communication between the network and the user is very low. Moreover, in our algorithm, per-session states need not be maintained at the network links. These features make the algorithm attractive in terms of practical deployment. On the other hand, the algorithms in [11] suffer from several practical shortcomings (they have high overhead of computation and communication, and require the network links to maintain per-session state). A detailed comparison of the algorithm proposed in this paper and those in [11] is presented in Section VIII of this paper. It is worth noting here that in this work, we do not try to address the question of what utility functions should be chosen, or how a desired fairness criterion can be mapped to user utilities. Instead, we address the question of how the globally optimal rates can be achieved once the user utility functions are appropriately chosen.

The paper is structured as follows. In Section II, the rate control problem is presented formally as an optimization problem. In Section III, we state the algorithm requirements, and outline our basic solution approach. In Section IV, we present an iterative algorithm for the rate control optimization problem. Section V presents the convergence analysis for this iterative optimization algorithm. In Section VI, we describe how this algorithm can be implemented in a real network. In Section VII, we demonstrate the convergence of our algorithm in an asynchronous network environment through simulations. We compare our approach with the existing approaches in Section VIII, and conclude in Section IX.

II. Problem Statement

First we describe the network model, and formulate the rate control problem as a convex optimization problem. In the subsequent sections, we will show how we can achieve the optimal rates for this problem.

Consider a network consisting of a set L of unidirectional links, where a link $l \in L$ has capacity c_l. The network is shared by a set of M multicast groups (sessions). Each multicast group is associated with a unique source, a set of receivers, and a set of links that the multicast group uses (the set of links forms a tree)[1]. Thus any multicast group $m \in M$ is specified by $\{s_m, R_m, L_m\}$ where s_m is the source, L_m is the set of links in the multicast tree, and R_m is the set of receivers in group m. As already mentioned, the total rate of traffic of a multicast group over any link on the tree must be equal to the maximum of the traffic rates of all downstream receivers of the group. Also note that unicast is a special case of multirate multicast (in the unicast case, the tree reduces to a single path between the source and the receiver).

Let R be the set of all receivers over all multicast groups. Also let $S_l \subseteq R$ denote the set of receivers using link $l \in L$. Each receiver r has a minimum required transmission rate $b_r \geq 0$, and a maximum required transmission rate $B_r < \infty$. Moreover, each receiver r is associated with a utility function $U_r : \Re_+ \rightarrow \Re$, which is assumed to be concave, bounded and continuously differentiable[2] in the interval $X_r = [b_r, B_r]$. Thus receiver r has a utility $U_r(x_r)$ when it is receiving traffic at a rate x_r, where $x_r \in X_r$.[3] We will refer to the variables x_r as the "receiver rates".

We are interested in maximizing the "social welfare", i.e., the sum of the utilities over all receivers, subject to the link capacity constraints, as well as the maximum/minimum rate constraints. The problem can be posed as:

$$\mathbf{P}: \qquad \text{maximize} \sum_{r \in R} U_r(x_r)$$

$$\text{subject to} \quad \sum_{m \in M} \max_{r \in S_l \cap R_m} x_r \leq c_l \qquad \forall\, l \in L \qquad (1)$$

$$x_r \in X_r \qquad \forall r \in R \qquad (2)$$

Note that $S_l \cap R_m$ is the set of receivers of group m that use link l. Thus the term $\max_{r \in S_l \cap R_m} x_r$ denotes the rate of traffic of multicast group m on link l. Also note that when $S_l \cap R_m = \phi$, the term $\max_{r \in S_l \cap R_m} x_r$ in (1) should be interpreted as zero.

Note that in the above formulation, the sets X_r are (bounded) continuous intervals. Therefore, it is assumed that the receiver rates can be continuous. In practice, however, bandwidth allocations can be limited to some discrete levels only (for example, in layered video, there will be some distinct bandwidth levels, one corresponding to each layer). However, constraining the set X_r to a set of discrete points (between the lower and the upper bounds on the receiver rates), makes the problem much harder to solve. Discretization of the rates destroys the convexity of the problem (it becomes an integer programming problem), which is crucial for developing a distributed solution. In the following, therefore, we develop a solution to the "convexified" or "relaxed" problem, as stated above. The actual rate is then computed by "rounding" the rates obtained (as a solution of the convexified problem $\mathbf{P}$) so that they correspond to the allowed discrete bandwidth levels. The rounding procedure, and the associated issues, are discussed in Section VI-C.

III. Preliminaries

In this section, we introduce some new terminology, which will help us in describing the algorithms presented in the subsequent sections of this paper. We then discuss the features that are necessary in any multirate multicast rate control algorithm for the algorithm to be practically viable. In this section, we also outline the basic solution approach that is used in deriving the optimization algorithm presented in the next section.

A. Terminology

Consider Figure 1, which shows an example of a multicast tree where s is the source node and $\{r_1, r_2, r_3, r_4\}$ is the set of receiver nodes. The rest of the nodes in the multicast tree can be classified into *junction nodes* and *non-junction nodes*, as shown in the figure. Junction nodes are the forking nodes, i.e., nodes where the multicast tree "branches off". Thus in Figure 1, $\{r_5, r_6, r_7\}$ are junction nodes. Receiver/junction nodes

[1] We assume fixed path routing. So the tree associated with each multicast group is fixed.

[2] The differentiability assumptions are only for the sake of simplicity of exposition and analysis. The algorithms and convergence results presented in this paper can be extended to non-differentiable functions by using *subgradients* [24] instead of the usual derivatives.

[3] We also assume that U_r and X_r are known only to receiver r.

of different multicast groups are considered to be logically different, even if they are physically located at the same node. In the rest of the paper, we assume that the receivers are only at the leaf nodes of the multicast tree. There is no loss of generality in assuming this, since a receiver at a non-leaf node can be replaced by creating a new leaf node and placing the receiver in it, and connecting the new leaf node to the non-leaf node (where the receiver is actually located) by a link with infinite capacity. Moreover, note that any leaf node must be a receiver node. The *parent* of a receiver/junction node r refers to the closest junction/source node in the upstream path from r towards the source. Also, by *child* of junction/source node r, we would refer to any receiver/junction node whose parent is the node r. Thus in Figure 1, r_5 is the parent of r_1, r_7 is the parent of r_5, s is the parent of r_7. Similarly, r_7 is a child of s, while r_5, r_6 are children of r_7, and so on.

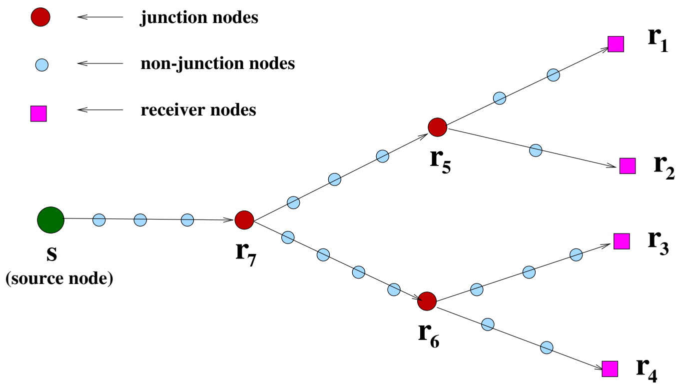

Fig. 1. An example of a multirate multicast tree

In general, we assume that the receiver decides its rate based on its utility function and the network congestion feedback. It then sends its request to its parent node. A junction node gathers all such requests (from its children nodes), takes the maximum of all the rates requested, and requests that rate from its parent node. Requests go up the tree through the junction nodes in this fashion until they reach the source node. The source sends traffic to its children nodes at their requested rates; these nodes then send traffic to their children nodes, and so on, and the traffic finally reaches the receivers at their requested rates.

B. Algorithm Requirements

In order to be practically viable, a rate control algorithm must be decentralized. Thus we would like to have a solution where the nodes in the network act like processors in a distributed computation system (where the coordinating information is exchanged in terms of congestion and rate feedbacks) and reach the system optimum without any centralized coordinator.

Closely tied to decentralization is the issue of scalability. A solution would not scale if, for example, the source or a junction node in the multicast tree has to maintain some state information for all downstream receivers of the tree. Since the number of receivers in the group can be large, this might lead to tremendous processing/storage pressure on such a node, particularly if the node is the source, or a junction node close to the source. Therefore we would like to have a solution where processing/storage overhead at a node in a multicast tree does not depend significantly on the size of the tree.

The rate control algorithm must scale not only with the size of a multicast group, but also with the total number of multicast sessions going through a link/node of the network. Therefore, we would like to have a solution where the network routers are not required to maintain state information on a per-flow (per-session) basis. However, due to the multirate nature of the traffic, some state overhead is unavoidable for routers that are junction nodes of one or more multicast sessions. This is because a junction node needs to store at least the rate information about each of its children. Thus a router has to maintain per-session information for all multirate multicast sessions for which it is a junction node. However, we would like to have a solution where the routers would not need to maintain any state information for a session for which it is a non-junction node. Thus in such a solution, no per-flow state would be required at the network nodes if all the sessions are unicast (since there are no junction nodes in the unicast case).

Conformity with existing standards is another important criterion. The rate control algorithm should be such that it can be implemented without a major modification to the existing standards. In the current networking standards like IP multicast, a junction node may not know the identity of all the downstream receivers, but will only know the downstream nodes it must forward a packet to. Therefore we require a rate control algorithm which does not require a junction node to communicate with nodes other than its immediate neighbors.

We would also prefer to have a solution where the complex computations (required for the optimization process) are limited to the end-hosts only. For practical viability, the computations that the core routers are required to perform must be kept simple.

It is also desirable that the overhead of information exchange (required in the optimization process) between the network and the end-hosts is as low as possible, such that it can be contained within a few bytes in the packet header.

The rate control algorithm that we propose in this paper satisfies all of the above criteria. It is distributed, and the user and the network algorithms are appealingly simple. The algorithm also has a low network feedback overhead. In the algorithm, the network needs to know only the receiver rates. This, however, can also be estimated by measuring the rates at the network nodes. In our algorithm, with measurement-based estimation of receiver rates, a router does not need to maintain per-session information for sessions for which it is a non-junction node; the per-session information maintained for sessions for which it is a junction node is also small. Moreover, a source/junction/receiver node only needs to communicate with its parent or children nodes, and does not need to know about the nodes further downstream/upstream. Thus the solution is scalable, and conforms well with the existing standards.

C. Solution Approach

Note that in the unicast version of the problem, the link constraints are linear and the problem $\mathbf{P}$ is separable. Separable problems are amenable to distributed solutions [4]. In our case, however, the problem $\mathbf{P}$ contains some $\max$ functions. The $\max$ functions, besides being nonlinear, couple several variables together, making the problem non-separable. Moreover, note that the $\max$ functions are non-differentiable. All these factors make the problem significantly different than its unicast version. Obtaining a solution that satisfies all the requirements described in the last subsection is an interesting and challenging problem.

The algorithm that we propose in this paper is developed using non-differentiable optimization methods, particularly those based on *subgradients*. A subgradient, defined in the context of convex/concave functions, can be viewed as a generalized gradient, and may exist even if the gradient does not (as is the case for non-differentiable functions). See Appendix I for the formal definition of subgradients and some of their important properties. The motivation, derivation and analysis of our algorithm draw from results in subgradient optimization theory, mainly those by N.Z. Shor and B.T. Poljak [24], [21]. The problem of non-separability (as well as non-differentiability) of the constraint functions can be effectively handled using subgradients. The use of subgradients thus allows us to develop a simple distributed solution to the non-separable problem $\mathbf{P}$. Our algorithm is developed in such a way that the scalability and other requirements stated above are also appropriately addressed.

IV. An Optimization Algorithm

In this section, we present an iterative optimization algorithm for the problem $\mathbf{P}$. The convergence properties of the algorithm is investigated in the next section. In Section VI we show how this algorithm can be implemented in a real network in a distributed and scalable way.

A. Notation

Before we present the algorithm, we introduce some notation that we will use. Let $\hat{R}$ be the set of all junction nodes (over all multicast groups). Let $\tilde{R} = R \cup \hat{R}$ be the set of all receiver and junction nodes (over all multicast groups). For any $r \in \tilde{R}$, let π_r denote the parent node of r. Thus in Figure 1, $\pi_{r_1} = r_5$, $\pi_{r_7} = s$, etc. For any $r \in \hat{R}$, let $C_r = \{r' : \pi_{r'} = r\}$ denote the set of all children nodes of r. Thus in Figure 1, $C_{r_7} = \{r_5, r_6\}, C_{r_5} = \{r_1, r_2\}$, etc. For any $r \in \tilde{R}$, let $\tilde{L}_r$ denote the set of all links whose immediate downstream junction/receiver node is r. In other words, $\tilde{L}_r$ is the set of all links between the nodes π_r and r in the particular multicast tree to which π_r and r belongs. Thus in Figure 1, $\tilde{L}_{r_7}$ consists of all links between s and r_7, $\tilde{L}_{r_5}$ consists of all links between r_7 and r_5, $\tilde{L}_{r_1}$ consists of all links between r_5 and r_1, and so on. Now define the set $\tilde{S}_l \subseteq \tilde{R}$ as $\tilde{S}_l = \{r : l \in \tilde{L}_r\}$. Thus $\tilde{S}_l$ consists of all junction and receiver nodes that are the immediate downstream nodes of sessions that go through link l.

For any $r \in \hat{R}$, let T_r denote the set of receiver nodes that are included in the tree rooted at r. Thus in Figure 1, $T_{r_5} = \{r_1, r_2\}, T_{r_7} = \{r_1, r_2, r_3, r_4\}$ etc. Now for each $r \in \hat{R}$, define a variable x_r such that it denotes the rate of traffic that the junction node r receives from its parent node (we will call these "junction rates" in analogy with "receiver rates"). Note that the junction rates are functions of the receiver rates. Thus for any $r \in \hat{R}$, x_r is defined as $x_r = \max_{r' \in T_r} x_{r'}$. Moreover, with this notation, for any $r \in \hat{R}$, $x_r = \max_{r' \in C_r} x_{r'}$. Also note that the capacity constraint for link l (cf., (1)) can now be simply written as $\sum_{r \in \tilde{S}_l} x_r \leq c_l$.

For any $r \in \tilde{R}$, let Q_r denote the set of all junction and receiver nodes from the source node to r, including r but excluding the source node. Thus in Figure 1, $Q_{r_5} = \{r_7, r_5\}$, $Q_{r_1} = \{r_7, r_5, r_1\}$, and so on.

B. An Iterative Algorithm

For any $r \in R$, let $x_r^{(n)}$ denote the rate of the receiver node r at the nth iterative step. Then for any $r \in \hat{R}$, $x_r^{(n)} = \max_{r' \in T_r} x_{r'}^{(n)}$ denotes the rate of the junction node r at the nth iterative step.

In our algorithm, the rate update procedure for receiver r at the nth iterative step can be summed up as follows: $x_r^{(n)}$ increases according to the "incremental utility" $U_r'(x^{(n)})$, while it decreases according to the "congestion penalty" $p_r^{(n)}$ ($p_r^{(n)}$ will be defined shortly). The quantity $p_r^{(n)}$ can be thought of as a measure of the congestion caused by r at step n, and thus determines the rate at which r "backs off" on detecting congestion in its path. As we will see later (when we describe the practical implementation of the algorithm in Section VI), the congestion penalty is basically the congestion feedback provided by the network to the receiver (user). Before we describe the rate update procedure in detail, let us define the congestion penalty formally in terms of the receiver rates and network parameters.

First we introduce a few variables that will be useful in defining the congestion penalty $p_r^{(n)}$. For each link $l \in L$, define $\varepsilon_l^{(n)}$ as a 0-1 variable denoting whether link l is congested or not at step n, i.e.,

$$\varepsilon_l^{(n)} = \begin{cases} 0 & \text{if } \sum_{m \in M} \sum_{r \in \tilde{S}_l} x_r^{(n)} \leq c_l \\ 1 & \text{if } \sum_{m \in M} \sum_{r \in \tilde{S}_l} x_r^{(n)} > c_l \end{cases} \tag{3}$$

We will refer to the variable ε_l as the "link congestion indicator" for link l. Now, for each $r \in \tilde{R}$, define $e_r^{(n)}$ as

$$e_r^{(n)} = \sum_{l \in \tilde{L}_r} \varepsilon_l^{(n)} \tag{4}$$

Therefore, $e_r^{(n)}$ indicates how many of the links in $\tilde{L}_r$ are congested at step n.

Let Ω be the set of all source nodes (over all multicast groups). Let $\tilde{R}_\Omega \subseteq \tilde{R}$ be the set of all junction and receiver nodes whose parent node is a source node. Thus $\tilde{R}_\Omega = \{r : \pi_r \in \Omega\}$. Associate a variable α_r satisfying $0 \leq \alpha_r \leq 1$ with each $r \in \tilde{R} \setminus \tilde{R}_\Omega$. We will refer to α_r as the "penalty splitting factor" associated with junction/receiver node r, the reasons for which will be clarified shortly. Let $\alpha_r^{(n)}$ denote the penalty splitting factor for r at the nth iterative step. We require these penalty splitting factors to satisfy certain conditions, as we will see later.

The definition of the congestion penalty, as will be stated shortly, can be motivated as follows. Let us interpret $\varepsilon_l^{(n)}$ as the penalty to be paid for congesting link l (by each of the multicast sessions using link l) at step n. Now consider a junction node r' belonging to any multicast group m. Then $e_{r'}^{(n)}$ is the total penalty to be paid by m for congesting the links in $\tilde{L}_{r'}$. Let this penalty be charged to r' (recall that for links in $\tilde{L}_{r'}$, r' is the closest downstream node belonging to m's multicast tree). Now let $r^{'}$ split this penalty among its children nodes. Also for any $r^{''} \in C_{r'}$, let $\alpha_{r''}^{(n)}$ be the factor that determines what proportion of this penalty is charged to $r^{''}$ (thus $r^{''}$ is charged a penalty of $\alpha_{r''}^{(n)} e_{r'}^{(n)}$). Each child node then splits the penalty charged to it amongst its children nodes (again according to the splitting factors of the nodes that are charged), and this goes on until the penalties are transferred to the receivers. It is then easy to see that the penalty charged to receiver $r \in T_{r'}$ (for congesting the links in $\tilde{L}_{r'}$) is equal to $(\prod_{r'' \in Q_r \setminus Q_{r'}} \alpha_{r''}^{(n)})\ e_{r'}^{(n)}$. Note that for any receiver node r, the penalty for congesting the links in $\tilde{L}_r$ is charged entirely to r, since it is the only downstream receiver (of that group) for those links.

Now assume that the penalties of all links of the multicast tree are split up amongst the receivers in the manner just described. Then a receiver pays a penalty for each of the links it uses (i.e., the links in the path from the source to that particular receiver). Note that for any receiver r, $\cup_{r' \in Q_r} \tilde{L}_{r'}$ represents the set of links that r uses. Therefore, the total penalty that receiver r pays is the sum of the penalties paid for the links in $\cup_{r' \in Q_r} \tilde{L}_{r'}$.

Now let us define the congestion penalty formally. For each $r \in R$, define $p_r^{(n)}$ as

$$p_r^{(n)} = \sum_{r' \in Q_r} (\prod_{r'' \in Q_r \setminus Q_{r'}} \alpha_{r''}^{(n)})\ e_{r'}^{(n)} \tag{5}$$

For $r^{'} = r$, the term $\prod_{r'' \in Q_r \setminus Q_{r'}} \alpha_{r''}^{(n)}$ should be interpreted as 1. Note that $p_r^{(n)}$ is zero if none of the links that receiver r uses is congested. Moreover, note that in the special case of an unicast session, the congestion penalty of the receiver of the session is simply the number of congested links in the path of the receiver/session.

Now we state the update procedure for the receiver rates. In the update procedure stated below, $[\,\cdot\,]_{X_r}$ denotes a projection[4] on the set X_r. For each $r \in R$, x_r is updated as follows

$$x_r^{(n+1)} = [\, x_r^{(n)} + \lambda_n\ (\ U_r^{'}(x_r^{(n)}) - K p_r^{(n)}\ )\,]_{X_r} \tag{6}$$

where K (the "penalty scaling factor") is a positive constant, and $\lambda_n > 0$ is the step-size at the nth iterative step.

C. Conditions on the splitting factors

For our algorithm to work correctly, at every step n, the splitting factors α_r must satisfy the following conditions

$$\alpha_r^{(n)} \geq 0 \qquad \forall\, r \in \tilde{R} \setminus \tilde{R}_\Omega \tag{7}$$

$$\sum_{r' \in C_r} \alpha_{r'}^{(n)} = 1 \qquad \forall\, r \in \hat{R} \tag{8}$$

$$\alpha_r^{(n)} = 0 \quad \text{if } x_r^{(n)} < x_{\pi_r}^{(n)} \qquad \forall\, r \in \tilde{R} \setminus \tilde{R}_\Omega \tag{9}$$

Constraints (7) state that the splitting factors are non-negative. Constraints (8) state that the sum of the splitting factors of all of the children of a junction node must add up to 1. Constraints (9) state that the splitting factor of a node is zero if it is not receiving the same rate as its parent. Since the rate of the parent node is the maximum of the rates of its children, this implies that the splitting factor of a node is zero if its rate is not the maximum amongst the rates of all of its sibling nodes. In other words, the penalty at a node is split amongst only those children who are receiving the maximum rates.

Note that the above constraints allows us to have both fractional and integral (0 and 1) splitting factors. Choosing fractional splitting factors, however, has certain drawbacks in terms of practical implementation, as we will discuss in Section VI. Therefore, in the rest of the paper, we will only be concerned with integral splitting factors. In that case, (7) is replaced by the following constraint:

$$\alpha_r^{(n)} \in \{0, 1\} \qquad \forall\, r \in \tilde{R} \setminus \tilde{R}_\Omega \tag{10}$$

V. Convergence Analysis

In this section, we investigate the convergence of the iterative algorithm outlined in the last section. For simplicity, the convergence analysis presented here assume that the splitting factors satisfy (8)-(10). However, the results can be shown to hold even if the splitting factors satisfy the more general conditions (7)-(9).

In the following, let $x = (x_r, r \in R)$ denote the vector of the receiver rates. Let $x^{(n)}$ denote the vector of receiver rates at the nth iterative step. Let X_R denote the entire region in the $|R|$-dimensional space where x is constrained to lie due to (2), i.e., $X_R = \{(x_1, ..., x_{|R|}) : x_r \in X_r\ \ \forall r \in R\}$. Thus the set of constraints in (2) can be equivalently written as $x \in X_R$.

A. Assumptions

In the convergence analysis, we make the following assumptions on the problem $\mathbf{P}$.

Assumption 1: (Feasibility) The problem $\mathbf{P}$ is feasible, i.e., $\sum_{m \in M} \max_{r \in S_l \cap R_m} b_r \leq c_l$ for all $l \in L$.

Note that in the special case when $b_r = 0\ \forall r \in R$, the feasibility assumption is satisfied.

Assumption 2: (Bounded slope) There exists an $A < \infty$ such that $U_r'(x_r) \leq A \quad \forall x_r \in X_r$ for all $r \in R$.

For the sake of simplicity of the analysis, we make an additional assumption in this paper, as stated below. However, this assumption is not necessary for guaranteeing convergence. Refer to [12] for the convergence results in the more general case.

Assumption 3: (Strict Concavity) The utility functions U_r are strictly concave in the interval X_r. Thus for every $r \in R$, there exists a $\gamma_r > 0$ such that $-U_r^{''}(x_r) > \gamma_r$ for all $x_r \in X_r$.

Note that the above assumption also implies that the optimal solution of $\mathbf{P}$ is unique. Let x^* be the optimal solution of $\mathbf{P}$. Define the overall user utility function $U : \Re_+^{|R|} \to \Re$ as $U(x) = \sum_{r \in R} U_r(x_r)$, and let $U^* = U(x^*)$ be the corresponding optimal value.

Next we state some convergence results under various conditions of the step-sizes.

B. Exact convergence with diminishing step-sizes

Assume that the sequence of step-sizes $\{\lambda_n\}$ in (6) satisfies the following criteria

$$\lim_{n \to \infty} \lambda_n = 0 \qquad \sum_{n=1}^{\infty} \lambda_n = \infty \tag{11}$$

[4] Since $X_r = [b_r, B_r]$, thus for any scalar y, $[y]_{X_r} = \min(B_r, \max(b_r, y))$.

As an example, $\lambda_n = (1/n)$ is a sequence that satisfies (11).

Let $\bar{R} = \max_{m \in M} |R_m|$ denote the maximum number of receivers in any multicast group. The following theorem shows that our algorithm converges to the optimum if the step-sizes satisfy (11).

Theorem 1: Consider the iterative procedure stated in (3)-(6), with the splitting factors satisfying (8)-(10), and the step-sizes satisfying (11). Then for all $K > A\bar{R}$, the sequence of rate vectors $\{x^{(n)}\}$ converges to x^*, the unique optimal solution of $\mathbf{P}$.

The above theorem is proved in Appendix II. Note that from the continuity of U it follows that $\lim_{n\to\infty} U(x^{(n)}) = U^*$.

Theorem 1 states that there is a minimum value of the penalty scaling factor K beyond which our algorithm converges to the optimal solution. Note that in the unicast case, this lower bound on K is simply the maximum derivative of the utility functions.

The algorithm can be also be shown to converge if the step-sizes λ_n satisfy $\lambda_n = \Lambda\lambda^n$ where $0 < \lambda < 1$ and Λ is a "sufficiently large" constant. Note that all these step-sizes satisfy $\lim_{n\to\infty} \lambda_n = 0$. This condition is required due to the non-differentiability of the problem. In practice, however, it may not be possible (due to precision limitations) or efficient (since it could slow down the convergence rate considerably) to decrease the step-size beyond a certain value. In the next subsection, therefore, we investigate the convergence of our algorithm with constant step-sizes.

C. Approximate convergence with constant step-sizes

If the step-sizes are constant, we can guarantee convergence of the rates to a neighborhood of the optimum. The result is formally stated below. A similar result holds even in the case where the step-sizes are not constant but converge to some positive value. Let $\Phi_\delta(x^*)$ be the set of all points at a distance of δ or less from x^* (the *δ-neighborhood* of x^*), i.e., $\Phi_\delta(x^*) = \{x : ||x - x^*|| \le \delta\}$. Let $\rho(x, Y) = \min_{y \in Y} ||x - y||$ denote the Euclidean distance of a point x from any compact set Y.

Theorem 2: Let $\{x^{(n)}(\lambda)\}$ denote the sequence of rate vectors defined by (3)-(6) (and the splitting factors satisfying (8)-(10)) with $\lambda_n = \lambda \ \forall n$. Then there exists a function $\delta(\lambda) \ge 0$ satisfying $\lim_{\lambda\to 0+} \delta(\lambda) = 0$, such that for all $K > A\bar{R}$,

$$\lim_{n\to\infty} \rho(x^{(n)}(\lambda), \Phi_{\delta(\lambda)}(x^*)) = 0 \qquad \forall \lambda > 0$$

The above theorem can be proved along the same lines as Theorem 1, and the proof is omitted for brevity. The theorem states that for a constant step-size, the distance of the rate vector from a neighborhood around the optimum tends to zero, and the size of this neighborhood becomes arbitrarily small with decreasing step-size. For a given constant step-size, the size of the neighborhood depends on the parameters of the problem $\mathbf{P}$, including the utility functions. Although obtaining a general explicit expression for the size of this neighborhood is difficult, implicit expressions of $\delta(\lambda)$ in terms of λ can be calculated [12]. However, the size of the neighborhood calculated on the basis of these expressions could be very conservative. Note that the above theorem also implies that given any neighborhood around the optimum, we can choose the step-size λ to be sufficiently small so that our algorithm (with constant step-sizes) achieves rates in that neighborhood.

Note that guaranteed convergence requires bounded utility derivatives (Assumption 2). If the utility functions have unbounded derivatives (as is the case for the function $\log(x)$ at $x = 0$), then the range of achievable rates can be restricted so that the utility derivatives are bounded in the restricted range. For instance, consider the utility function $\log(x)$ where the rate x can vary over the range $[0, B]$. Since the utility derivative is unbounded at $x = 0$, we could restrict the range to $[\epsilon, B]$, where ϵ is some small positive number. Our algorithm can be applied to the problem with this restricted range, and the rates achieved will be close to optimal.

VI. Distributed Implementation

Now we describe how the algorithm described in Section IV can be implemented in an asynchronous network environment in a distributed and scalable way .

A. Protocol Description

First we describe how the protocol works. As mentioned before, in our algorithms, a source/junction/receiver node needs to communicate only with its parent and children nodes. Assume that each source/junction node sends congestion packets (CP) (containing the congestion penalty information) to its children nodes. Also assume that each receiver/junction node sends rate packets (RP) (containing the rate information) to its parent node. Thus the CPs move in the downstream direction of the tree, while the RPs move in the upstream direction (see Figure 2). The CPs that a junction node sends to its children are sent out when the junction node receives a CP from its parent. Moreover, the RP that a junction node sends to its parent is formed by merging the RPs that it receives from all of its children. As in the figure, each CP contains a congestion penalty field $\overline{p}$, while each RP contains a rate field $\overline{x}$.

A junction/receiver node communicates its rate request to its parent node through the $\overline{x}$ field of the RP. This is to let the parent node know at what rate it needs to send traffic to the corresponding child. The parent node also uses these communicated rates to determine which of the children are requesting the maximum rates, and penalize only those children. For this purpose, each junction node r maintains $C_r^{\max} (\subseteq C_r)$, the set of the children requesting the maximum rates.

A source/junction node conveys the appropriate congestion penalty to its children nodes through the $\overline{p}$ field of the CP. Note that choosing fractional splitting factors makes the penalty term fractional, and this makes it difficult to convey it to the receiver using a few bytes, without sacrificing precision. For good precision, we require that the $\overline{p}$ field be fairly large, and this results in a high protocol overhead. To avoid this problem, we can just assign integral splitting factors, i.e., 0 and 1. In this case, conditions (8)-(9) require that a splitting factor of 1 be assigned to *any* one of the children that is requesting the maximum traffic rate, while a splitting factor of 0 be assigned to all other children (whether they are requesting the maximum traffic rate or not). Note that it does not matter which one of the children (amongst those that request the maximum rate) is chosen to pay the penalty, and the child that is penalized could be different

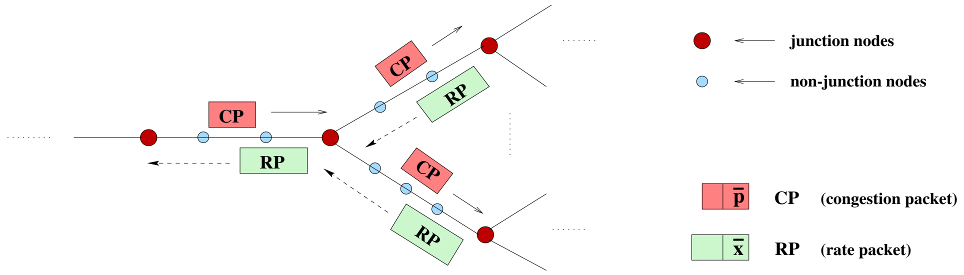

Fig. 2. Message exchanges

at different times (iterations). The algorithms described below assume this kind of penalty splitting. This ensures that the number of bytes that need to be allocated to the $\overline{p}$ field is small (we discuss more on this later).

Also assume that link l (i.e., the node associated with link l, which is usually the node where the link originates) is responsible for keeping track of the link congestion indicator variable ε_l. Moreover, for any receiver/junction node r, the node itself is responsible for keeping track of the receiver/junction rate x_r.

B. Link and Node Algorithms

On receiving RPs from all of its children nodes, a junction node computes the maximum of the rates requested, and sends an RP to its parent, setting the $\overline{x}$ field to this maximum requested rate. When an RP is going through link l, the node reads the field $\overline{x}$ and uses it to update the congestion indicator ε_l (see the link algorithm below).

When sending a CP to a child, a source node stamps 0 in the $\overline{p}$ field of the CP. Each link on the path to the child adds the link congestion indicator (0 or 1) in the $\overline{p}$ field of the CP. A junction node transfers the $\overline{p}$ field of the CP that it receives from its parent node to the CP of one of the children that has requested the maximum rate; the $\overline{p}$ fields of the CPs for the rest of the children are stamped as 0. Thus when a receiver node receives a CP, the $\overline{p}$ field contains the appropriate congestion penalty for that receiver, which it uses for updating its rate according to (6).

Note that in real implementation, these control packets (CPs and RPs) need not be communicated as separate packets; the congestion penalty can be conveyed through a field in the data packets, while the rate information can be conveyed through a field in the acknowledgement (ACK) packets.

In the algorithms described below, the step-size for rate updates is kept constant at λ.

Link l's algorithm:

On receiving an RP:
1. Read the $\overline{x}$ field to know the current rate of that session, and forward the RP on to the next link.

On receiving a CP:
1. Add ε_l to the $\overline{p}$ field of the CP and forward it on to the next link.

Periodically:
1. Update the link congestion indicator ε_l as

$$\varepsilon_l \leftarrow \begin{cases} 1 & \text{if} \quad \sum_{r\in\tilde{S}_l} x_r > c_l \\ 0 & \text{if} \quad \sum_{r\in\tilde{S}_l} x_r \le c_l \end{cases}$$

Source node s's algorithm:

On receiving an RP:
1. Read the $\overline{x}$ field to know the new rate requested by the child.
2. Send a CP to that child, setting the $\overline{p}$ field to 0.

Receiver node r's algorithm:

On receiving a CP:
1. Read the $\overline{p}$ field of the CP to know the current congestion penalty.
2. Send an RP to the parent node, setting the field $\overline{x}$ to x_r.

Periodically:
1. Update the receiver rate as:

$$x_r \leftarrow x_r + \lambda U_r'(x_r) - \lambda K p_r$$

where p_r is an estimate of the current congestion penalty of r.
Now, if $x_r < b_r$, set $x_r \leftarrow b_r$, and if $x_r > B_r$, set $x_r \leftarrow B_r$.

Junction node r's algorithm:

On receiving a CP:
1. Send one CP to each of the children nodes, setting the $\overline{p}$ field as follows:
(a) Pick any child node in $C_r^{\max}$, and set the $\overline{p}$ field of its CP to the $\overline{p}$ field of the CP received from the parent node.
(b) Set the $\overline{p}$ fields of the CPs of all other children nodes to 0.

On receiving an RP:
1. Read the $\overline{x}$ field to know the new rate requested by the child, and do the following:
(a) Update the junction rate x_r as: $x_r \leftarrow \max_{r' \in C_r} x_{r'}$.
(b) Update $C_r^{\max}$ as: $C_r^{\max} \leftarrow \{r' : x_{r'} = x_r\}$.
2. On receiving RPs from all of the children nodes, send an RP to the parent node, setting the $\overline{x}$ field to x_r.

C. Implementation Issues

Note that in the algorithm described above, the receiver could request any rate between its minimum and maximum required rates. However, as discussed in Section II, in practice, bandwidth allocation is constrained to occur only at certain discrete levels (which are typically predetermined, and correspond to the cumulative layer bandwidths). Therefore, a source node (or junction node) can send traffic to a child node only at a rate that corresponds to a discrete level close to the requested rate. The granted discrete level can be the closest level no more than the requested bandwidth (rounding down), or it can be the closest level no less than the requested bandwidth (rounding up). The latter, however, can result in rates that are infeasible. Therefore, in practice, rounding down may be more preferable, and will result in rates that are feasible, and close to the optimal (the degree of closeness (to optimality) depends on the density of the discrete bandwidth levels).

Now let us calculate the number of bits that must be allocated to the $\overline{p}$ field of the CP. Firstly note that the value of $\overline{p}$ is upper bounded by $\bar{L}$, the maximum number of links on a source-receiver path. This is due to the fact that in the worst case, all the links from the source to a receiver can be congested, and the penalty splitting factors of all the junction/receiver nodes on that path could be one. This implies that we need to allocate $\lfloor \log_2 \bar{L} \rfloor + 1$ bits to the $\overline{p}$ field. Therefore for most real networks, including the internet, allocating just one byte for the congestion penalty field should be sufficient (note that one byte would allow 255 links on a path from the source to the receiver). Thus the overhead of the network congestion feedback to the receivers is quite small.

The implementation of the link algorithm, as described in the previous subsection, has an important drawback. Note that in the implementation described, the link has to keep track of the rates of all individual sessions that traverse that link. This implies that a router has to maintain per-session state even for sessions for which it is a non-junction node. This is certainly undesirable, as we have argued in Section III-B. However, note that the session rates can also be estimated by traffic measurements at the links. Also note that in order to determine whether a link is congested or not, we only need to know the *total* rate of traffic at that link, and not the individual session rates (see the link algorithm described above). Thus we could determine the value of the link congestion indicator just by measuring the total arrival rate at the link. Therefore, with measurement-based rate estimation, maintaining per-flow state at the links is not necessary. It is easy to see that with this modification, the distributed implementation of our algorithm (as described in the previous subsection) satisfies all the desirable features listed in Section III-B.

However, note that if rates are measured (and not communicated), there has to be estimation errors. These errors will be more significant because the traffic is sent at rates that are slightly different from the requested (computed) rates (due to rounding, as discussed above). We will discuss the effects of these estimation errors on performance in Section VII.

D. One-bit congestion feedback

As discussed, we require a byte in the packet header to carry the network congestion feedback. Although using one byte of the data packet/ACK packet header does not introduce a significant overhead, it is still interesting to investigate if the algorithm can be implemented with a single bit of network congestion feedback. In that case the algorithm could be implemented with the proposed Explicit Congestion Notification (ECN) bit.

For one-bit implementation, we could use an approach similar to Random Early Marking (REM) proposed in [1]. In this approach, there is a single bit for network congestion feedback in each packet, and this bit is marked probabilistically at each link, based on whether the link is congested or not. Each junction node transfers this bit to the child receiving traffic at the maximum rate, in a way similar to that described in Section VI-B. The receivers can then infer the congestion penalty by measuring the number of marked packets received over some time window. The packet marking process at the links and the penalty estimation process at the receivers are described more formally below.

If a link is congested (according to (3)), then a packet is marked with probability θ ($0 < \theta < 1$), chosen appropriately. If the link is not congested, the packet is not marked. The marking process in different links are independent; therefore, the probability that the packet is marked after traversing k congested links is $1 - (1 - \theta)^k$. Let ν_r be the proportion of marked packets measured over some time window at receiver r. Then the estimated congested penalty of receiver r, denoted by $\hat{p}_r$, can be calculated as

$$\hat{p}_r \quad = \quad \min\{ \frac{\log(1-\nu_r)}{\log(1-\theta)} \, , \; \bar{L} \} \tag{12}$$

where $\bar{L}$ is the maximum number of links on the path from a source to a receiver, as before.

In the next section, we evaluate the performance of this random single-bit marking based implementation of our algorithm and compare it with the original (deterministic) implementation, described previously.

VII. Simulation Results

Simulations carried out on various network topologies/scenarios confirm that our algorithm, as described in Section VI, achieves the optimal rates in an asynchronous slowly time-varying network environment. In this section, we present a few representative examples to demonstrate this fact.

Figure 3 shows the example network that we consider, which consists of two multicast groups sharing a 11-node 10-link network. We assume layered multicasting, and each multicast group can send traffic in 20 layers, each of the layers having a bandwidth of 0.25 MBps. Therefore, the maximum allowed bandwidth is 5 MBps, and bandwidth can be allocated in units of 0.25 MBps. Any particular discrete bandwidth level can be achieved by sending an appropriate number of layers. Note that layers are always sent cumulatively. Therefore, to achieve a rate of k*0.25 MBps, the lowest k layers need to be sent. Note that in layered multicasting, each data packet belongs to one particular layer. Therefore, a source/junction node can send traffic to its child at a particular discrete bandwidth level (computed by rounding down the rate requested by the child) simply by sending/forwarding only those data packets which belong to a corresponding set of cumulative layers.

In our experiments, the algorithms are implemented as described in the previous section, and the step-size of rate updates (λ) is kept fixed. However, the congestion penalty is not sent through separate control packets (CPs); instead, the congestion penalty field $\overline{p}$ is part of each data packet itself. Data packets, that travel on the forward paths, are assumed to be 400 bytes each. The rate information is carried on the upward path by RPs, as described in Section VI. Each RP is assumed to have a length of 40 bytes. Each receiver node/junction node sends out these RPs to its parent node periodically (once every 0.05 secs).

In all of the simulations described in this paper, maximum utilization of a link is set to 95%. Therefore, a link determines if it is congested or not depending on whether the overall estimated traffic on the link exceeds 95% of its capacity or not.

In the network in Figure 3, the utility functions of receivers r_4 and r_6 are $0.5\ln(1+x)$, while those of the rest are $\ln(1+x)$ (where x is expressed in MBps). The minimum rate for each receiver is 0, and the maximum rate is the capacity of the link leading to the receiver. Note that since r_5 is connected directly to the source, it behaves essentially like an unicast session. In our simulation scenario, the sequence of arrivals/departures of receivers are as follows. The receivers r_1, r_2, r_3, r_6 and r_7 arrive at time $t = 0$. Receiver r_5 joins at $t = 30$ secs, receiver r_4 joins at $t = 60$ secs, r_2 leaves at $t = 90$ secs, and r_6 leaves at $t = 120$ secs. All receivers start with an initial rate of zero. The receiver rates are updated every 0.05 secs. Note that a receiver will receive many data packets between two rate update instants, and the congestion penalty in the $\overline{p}$ field of these packets could be different. In our simulations, the congestion penalty estimate that a receiver uses in the rate computation procedure is computed by averaging the penalties over all data packets received since the last rate update. For the simulation results presented in this section, $\lambda = 0.15$ (in MBps), and $K = 1.2$.

First we consider the case where the rates are explicitly communicated to the links (the case with rate estimation at links is considered later).

Figure 4, which shows some rate plots in the time window 0-180 secs, demonstrates the performance of our algorithm in the particular example considered. Figure 4 shows the computed (requested) receiver rates of r_2, r_4, r_6 and r_7, along with the optimal rates (these 4 receivers were chosen arbitrarily, and rate plots of the other receivers also exhibit a similar trend). The rates are plotted every 0.05 secs, which is also the time interval between successive rate updates at the receivers. Note that the sudden changes in the optimal rates at $t = 30, 60, 90, 120$ are due to the arrival/departure of receivers. The plots demonstrate that the computed receiver rates track the optimal rates closely even as the optimal rates change.

Observe that in the plots in Figure 4, the computed rates do not exactly converge to the optimal rates, but fluctuate rapidly, remaining close to the optimal rates. The thickening of the receiver rate plots are due to these small but rapid fluctuations around the optimal values. Recall that in Section V, we argued that due to the non-differentiability of the problem we need step-sizes close to zero in order to guarantee exact convergence. If the step-size is constant, but small, as in the case of the plots in Figure 4, then we can only guarantee that our algorithm achieves rates that are close-to-optimal (Theorem 2). When the total traffic is close to the link capacity, the link congestion indicator fluctuates between 0 and 1, as can be expected from intuition. Moreover, when multiple children request the maximum traffic rate from a junction node, the penalty splitting factors for those children will also fluctuate between 0 and 1, as can be expected from the description of the junction node's algorithm presented in the last section. This causes the receiver penalty p_r to fluctuate, causing rate fluctuations like those seen in Figure 4. Smaller step-sizes cause smaller fluctuations, but also result in lower convergence speeds. Thus the choice of the step-size is a tradeoff between the convergence speed and the magnitude of fluctuations. In practice, a receiver could choose large step-sizes initially (to ensure fast convergence), and reduce the step-sizes once it detects that its rate is fluctuating around the same mean value (to reduce fluctuations when the rates are close to the optimal values).

Now consider the case where the links update their congestion indicators based on measured rates. In our simulations, link congestion indicators are updated after every 0.02 secs (based on the average arrival rate since the last update instant). Figure 5 shows the rate plots for this case (all other simulation conditions are similar to the case in Figure 4). The plots demonstrate that computed rates track the optimal rates even when the link rates are estimated by measurement. Comparing Figures 4 & 5, we observe that the magnitude of rate fluctuations is slightly greater (on an average) in the latter case. This is due to the errors in rate estimation at the links.

Note that Figures 4 & 5 show the computed (requested) rates. The rate at which a receiver receives traffic will typically be slightly less, since it is computed by rounding down the requested rate. The rates at which the receivers receive traffic are shown in Figure 6. The rates shown are the traffic rates measured at the receiver (each point in the plot is computed by averaging the receiver rates over a period of 1 sec). The figure shows that the actual received rates are fairly close to the optimal rates. The slight difference between the optimal and the received rates are due to the rounding down procedure, as mentioned before. Note that the optimal rates plotted in the figure are the optimal rates computed based on the relaxed problem, and not the optimal rates of the actual discretized problem (which can only be computed by solving a very complex integer program). However, since the achieved rates are fairly close to the optimal rates of the convexified problem, they are expected to be close to optimal rates of the actual discretized problem (note that the optimal objective function value of the convexified problem is an upper bound on the optimal objective function value of the discretized problem).

Recall that convergence of our algorithm is guaranteed only when the constant K is "sufficiently large" (Theorems 1 & 2). However, setting K to a very large value could reduce the average throughput considerably, as we would intuitively expect. Therefore, the value of K should be chosen carefully to ensure good performance in practice. Note the value of K in this example is 1.2, which is less than the lower bound for guaranteed convergence, as stated in Theorems 1 & 2. Therefore, this example also demonstrates that in practice, the rates can converge to the optimal values even for a value of K smaller than the stated lower bound.

Note that in the scenario described above, there is a large number of layers, and bandwidth can be allocated at fine granularity. However, in practice, there can be a few layers (4-5, for example) and the bandwidth difference between layers can be large. Thus bandwidth can be allocated only at coarse granularity. Figure 7 plots the received rates when there are 5 layers, each of 1 MBps. All other simulation conditions are the same as that in Figure 6. The figure also plots the optimal rates (computed based on the relaxed problem, i.e., assuming no discreteness in bandwidth allocation), and the optimal rates rounded

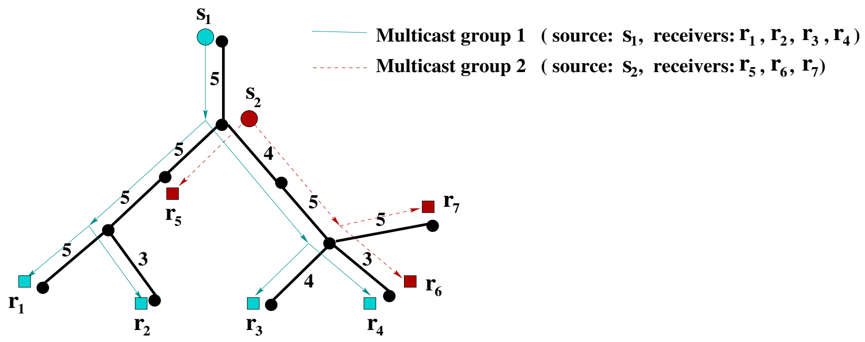

Fig. 3. An example network (The numbers associated with the links are the link capacities (in MBps). The propagation delay for each link is 1 ms.)

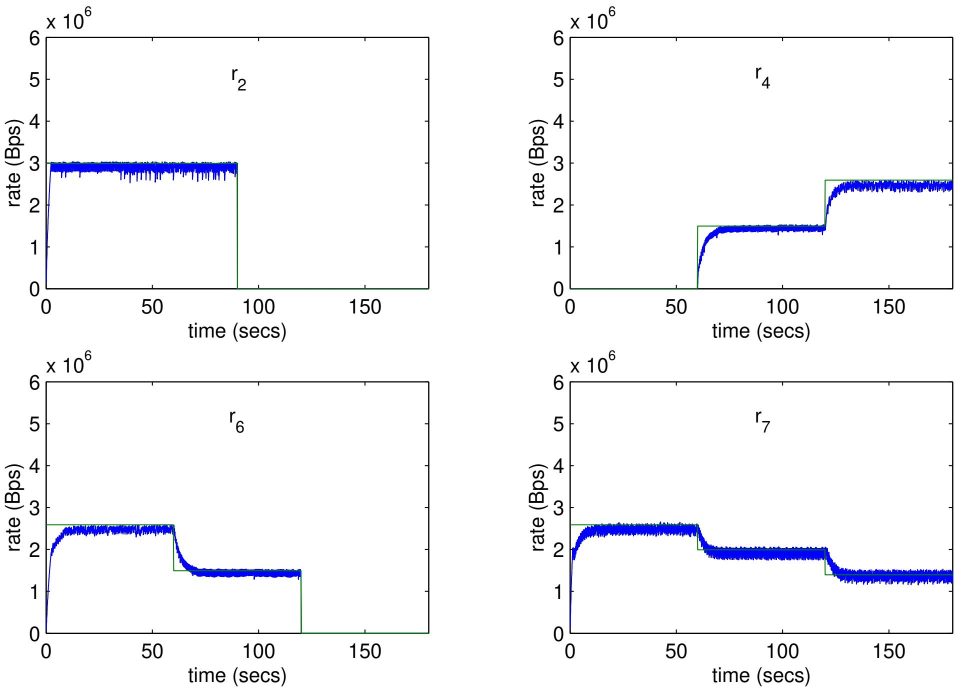

Fig. 4. Convergence of computed rates (with explicit rate communication). (The straight lines represent the optimal (theoretical) rates.)

down so that it corresponds to a cumulative layer bandwidth. As an example, if the optimal rate is 2.4 MBps, then the rounded down optimal rate in this case will be 2 MBps, whereas if the optimal rate is 1.7 MBps, the rounded down optimal rate is 1 MBps. The broken straight lines represent the optimal rounded down rates. From Figure 7, we see that the received rates lie between the optimal rates and the optimal rounded down rates. This fact was observed in most of the simulations. In a few cases, however, the rates lie between the optimal rates and the rounded up optimal rates (defined in a similar way as rounded down optimal rates). To sum up, in the case where there are few widely separated layers, the rates achieved by our algorithms lie within one layer (above or below) of the optimal rates of the relaxed problem. This error of one layer is unavoidable due to the discreteness of the problem.

In the simulations described above, all the layers have equal bandwidth. However, in practice, the layers could have widely different bandwidths, and so the discrete achievable bandwidth levels may not be evenly spaced. Figure 8 plots the received rates for the case where the discrete bandwidth levels are geometric. Here we have 5 layers, and the rates that can be allocated are 0.25 MBps, 0.5 MBps, 1 MBps, 2 MBps and 4 MBps (therefore, the first layer has a bandwidth of 0.5 - 0.25 = 0.25 MBps, the second layer has a bandwidth of 1.0 - 0.5 = 0.5 MBps, etc.). The simulation environment is the same as before. As the plots show, the observations in this case are similar to those described previously in the context of Figure 7.

Next we evaluate the performance of the random marking based single-bit implementation of our algorithm, as described in Section VI-D. Figure 9 plots the received rates in this case with $\phi = 0.25$ and $\bar{L} = 4$ (see 12). (Note that the maximum number of links on the path from a source to a receiver in the network in Figure 3 is 4). The time window chosen for averaging (for computation of the congestion penalty based on packet marks) is 0.05 secs, which is also the interval between successive rate updates at the receivers. All other simulation conditions are the same as those in Figure 6. The plots demonstrate that the achieved rates track the optimal rates in this case too. Comparison with Figure 6 reveals that the rate fluctuations in the case with random marking is greater than that with the determinis-

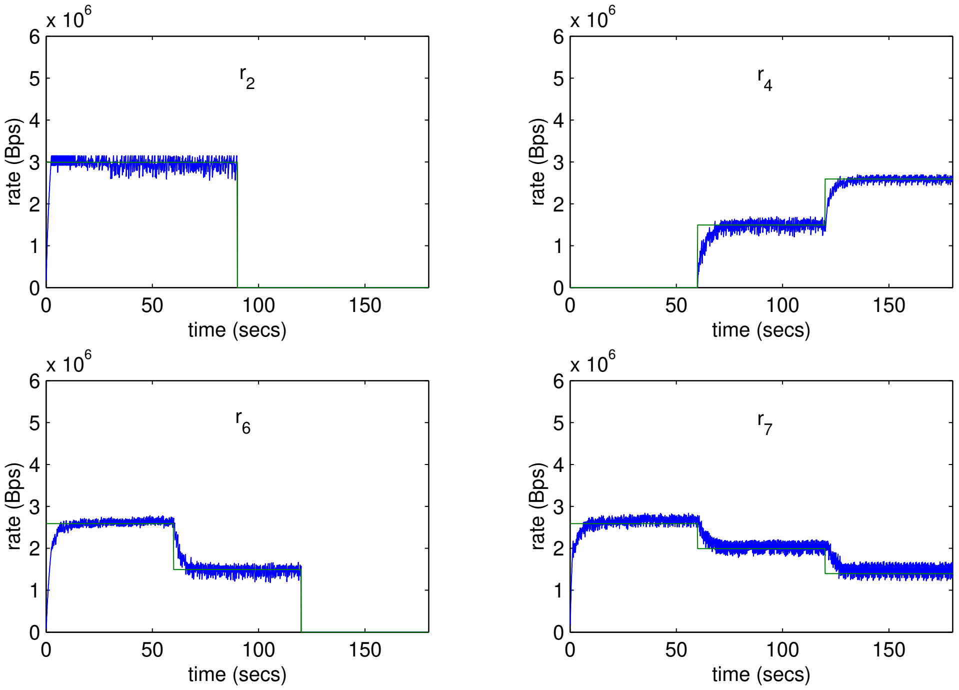

Fig. 5. Convergence of computed rates (with measurement based estimation of rates). (The straight lines represent the optimal (theoretical) rates.)

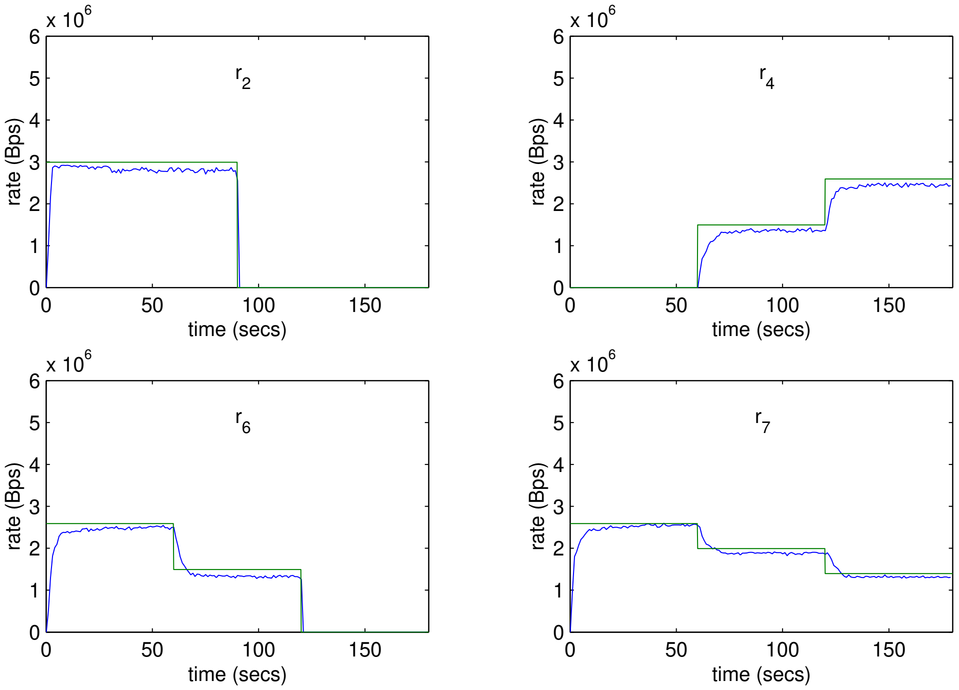

Fig. 6. Convergence of received rates with 20 layers (with measurement based estimation of rates). (The straight lines represent the optimal (theoretical) rates.)

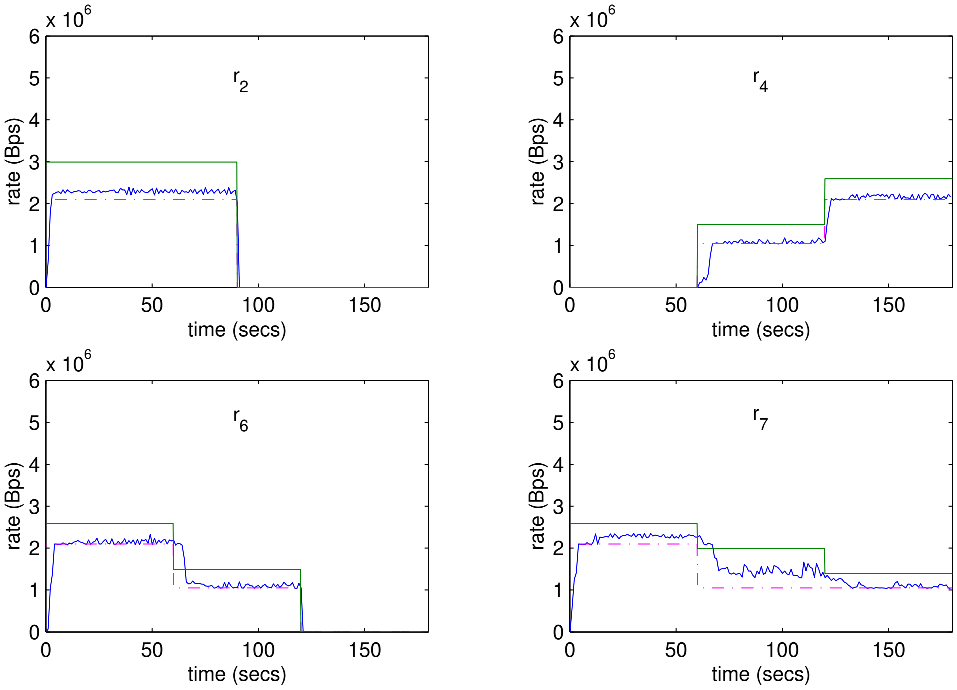

Fig. 7. Convergence of received rates with 5 layers, uniformly spaced (with measurement based estimation of rates). (The unbroken straight lines represent the optimal rates, computed based on the relaxed problem. The broken straight lines ($- \cdot -$) represent the rounded down optimal rates.)

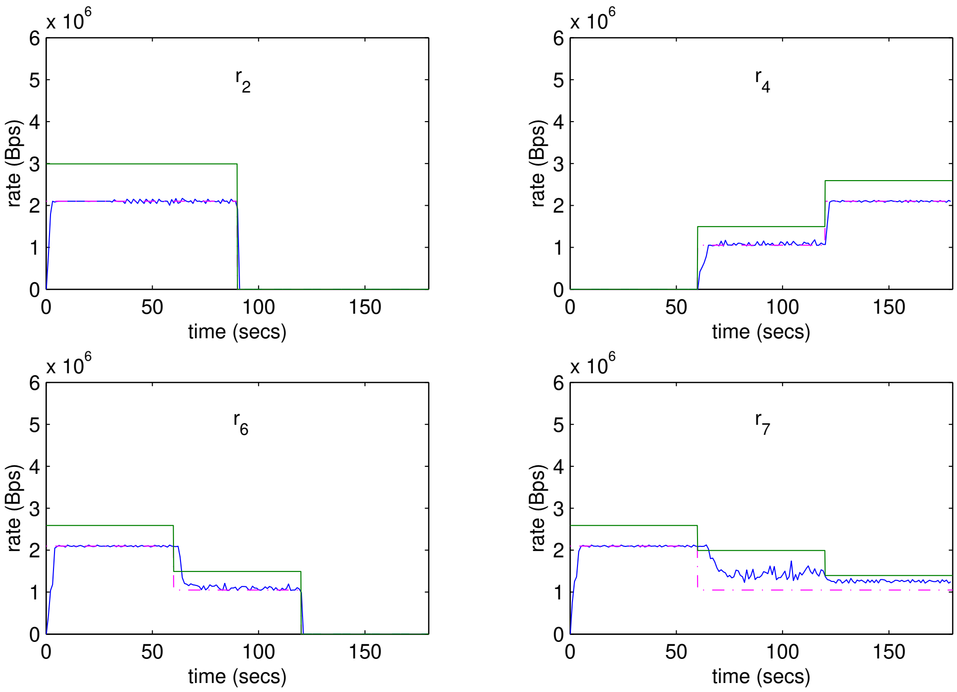

Fig. 8. Convergence of received rates with 5 layers, geometrically spaced (with measurement based estimation of rates). (The unbroken straight lines represent the optimal rates, computed based on the relaxed problem. The broken straight lines ($- \cdot -$) represent the rounded down optimal rates.)

tic algorithm, as we would intuitively expect. Figure 10 plots the received rates with random marking, but with 5 layers of 1 MBps each. Comparing with Figure 7, we see that the observations in this case are similar, although the magnitude of rate fluctuations is slightly larger with random marking. The simulation results with geometrically spaced layers are similar, and are omitted for brevity.

VIII. Related Work

In this section, we mention some of the recent work on the unicast version of the problem that we have addressed in this paper. We also compare, in detail, the algorithm presented in this paper with an alternative approach to the same problem (the multirate multicast case), presented in [11].

An aggregate utility maximization approach to flow control was suggested recently by Kelly [13]. Recently, this problem has received considerable attention in the context of unicast networks. Several flow control algorithms, both rate-based and window-based, have been proposed (see [17], [14], [16], [28], [10]). These algorithms were derived using different optimization approaches, and we will not discuss them here. Amongst these, the unicast algorithm presented in [10] is also based on subgradient optimization methods. For the special case of all unicast sessions, the algorithm presented in this paper reduces to a form that has certain similarities with the algorithm in [10] (particularly the fact that in both cases, the congestion feedback from the network to the user is the number of congested links on user's path). However, compared to the the algorithm in [10], the all unicast version of our algorithm guarantees convergence under much weaker assumptions on the receiver utility functions and the penalty scaling factor.

For the case of multirate multicast sessions, the optimization based rate control problem has not been adequately addressed. As we have already argued in earlier sections, the non-separability and non-differentiability of the problem and the multicast-specific requirements make this problem much more complex than its unicast version. In [11], the authors address the multirate multicast utility maximization problem and propose dual-based algorithms for it. The algorithms are distributed, and do not require the network to know the receiver utilities. The processing, storage and communication overheads at a junction node is proportional to its number of children. In spite of these attractive features, the algorithms in [11] suffer from certain drawbacks which limit their practical viability.

In the algorithms in [11], the network determines its congestion level based on certain "pseudo-rates" which could be different from the actual rates. The pseudo-rates can not be inferred from the actual traffic rates. These pseudo-rates need to be stored at junction nodes, and also to be communicated between a parent and children nodes, thus increasing the storage and communication overheads significantly. More importantly, each link has to keep track of the pseudo-rate of each of the sessions going through it (in order to update the link congestion indicator). Therefore, the network links need to maintain persession state. Hence these algorithms do not scale as the number of sessions traversing a link increases. Moreover, the pseudo-rates are communicated between a parent and its children nodes, thus increasing the storage and communication overheads significantly. As we have argued before, in our algorithm, no persession information needs to be maintained at the non-junction nodes. Moreover, we do not have any extra overhead of storing and communicating pseudo-rates.

In the algorithms proposed in [11], the congestion information ("congestion prices") that the network needs to communicate to the users are real numbers that could vary over a wide range. This poses a difficulty in communicating the price to the end-host using a small number of bits. While one can use some probabilistic marking policies (following the approach in [1]) to convey the congestion information is a single bit, it is not clear if such policies can provide theoretical convergence guarantees (note that even if the algorithm converges in that case, the convergence would be in some probabilistic sense). On the other hand, our algorithm has guaranteed deterministic convergence, and would require, in practice, no more than one byte in the packet header for conveying the congestion information.

In terms of computational overhead too, our algorithm is significantly better than those proposed in [11]. In the latter, the junction nodes are required to solve a maximization problem. This could impose a considerable computational overhead on the core routers of the network. In our case, however, the algorithms of the junction (as well as the non-junction) nodes are extremely simple, and therefore, the computational overhead on the core routers is small. Moreover, in certain cases, the receiver algorithm too could be much simpler in our case as compared to that in [11] (note that the algorithms in [11] require the receiver to compute a maximizer, whereas in our case, the receiver only needs to compute a derivative).

It is also worth noting here that the algorithm presented here guarantees convergence for a wider class of utility functions as compared to the algorithms in [11]. Our algorithm guarantees convergence for linear utility functions, and also a wide range of non-differentiable utility functions, which is outside the framework of the algorithms in [11].

IX. Concluding Remarks and Future Work

In this paper, we considered the rate control problem for multirate multicast sessions, with the purpose of maximizing the aggregate user utility. This utility maximization problem integrates various fairness criteria in a common framework. We presented a simple rate control algorithm that achieves the optimal rates for this problem, and can thus be used to achieve various fairness criteria (by choosing the utility functions appropriately). The algorithm is distributed, and scalable, both in terms of the size of each session and the number of sessions in the network. An attractive feature of the algorithm is that the computational burden on the core routers, as well as the end-hosts, is low. Moreover, the overhead of communication between the user and the network is also small.

Acknowledgements

Very recently, Deb and Srikant [8] have developed an algorithm that is quite similar to the one presented in this paper. Their work, done independently and completed in parallel with our work, is also based on primal subgradient techniques. However, while congestion on a link is measured in terms of the packet loss rate in their case, link congestion is indicated by

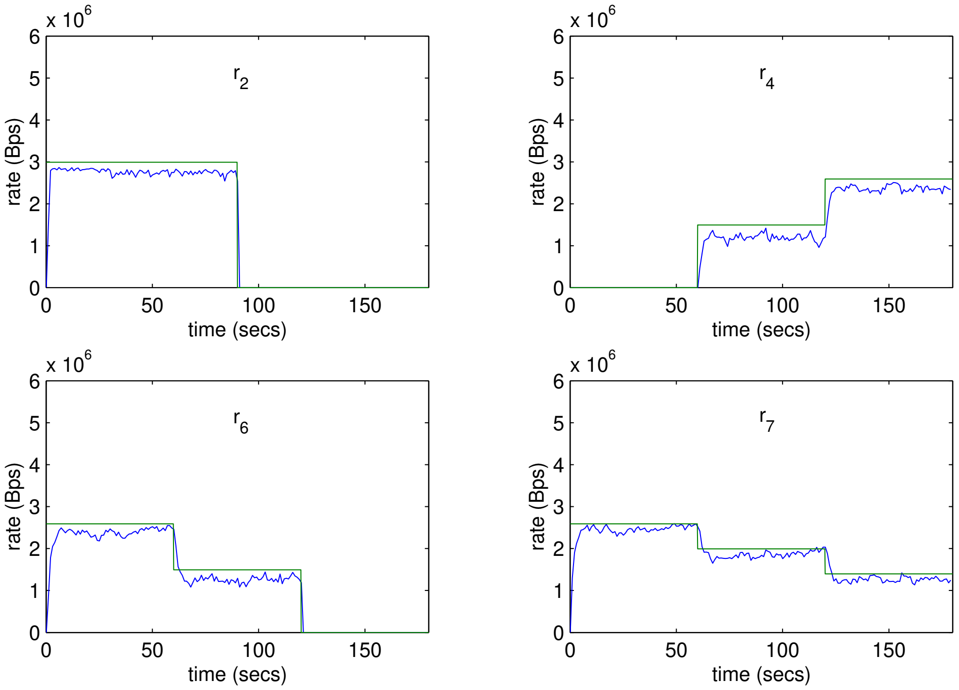

Fig. 9. Convergence of received rates with 20 layers, and random single-bit marking (with measurement based estimation of rates). (The straight lines represent the optimal (theoretical) rates.)

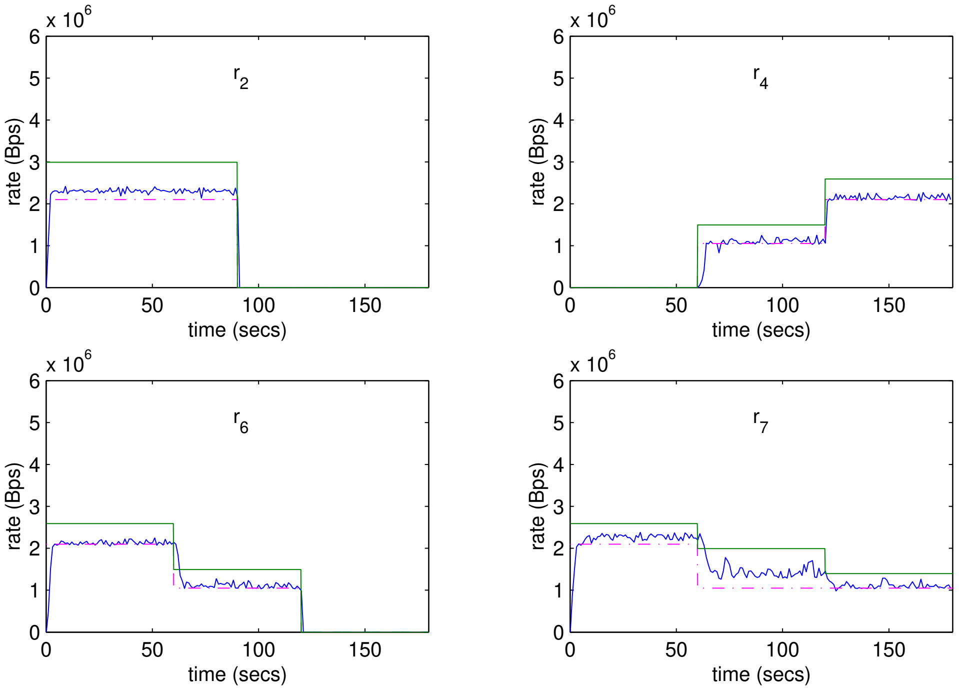

Fig. 10. Convergence of received rates with uniformly spaced 5 layers, and random single-bit marking (with measurement based estimation of rates). (The unbroken straight lines represent the optimal rates, computed based on the relaxed problem. The broken straight lines $(- \cdot -)$ represent the rounded down optimal rates.)

a single bit (congested/uncongested) in our case. Whereas the approach taken in our paper can be considered to be a generalization of the approach presented in [10] for the unicast case, the approach in [8] can be viewed as a generalization of the approach proposed in [14], [28] for unicast sessions.

References

[1] S. Athuraliya, S. Low, D. Lapsley,"Random Early Marking", *Proceedings of the First International Workshop on Quality of future Internet Services (QofIS) 2000*, Berlin, Germany, September 2000.

[2] Y. Bartal, J. Byers and D. Raz, "Global Optimization using Local Information with Applications to Flow Control", *Proc. of the 38th Ann. IEEE Symp. on Foundations of Computer Science (FOCS)*, Miami, USA, October 1997.

[3] D. P. Bertsekas, "Necessary and Sufficient Conditions for a Penalty Method to be Exact", *Math. Programming 9*, pp. 87-99, 1975.

[4] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, 1995.

[5] D. P. Bertsekas, R. Gallagher, *Data Networks*, Prentice Hall, Second Edition, 1992.

[6] D. P. Bertsekas, J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Athena Scientific, 1989.

[7] T. Bially, B. Gold, 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.

[8] S. Deb, R. Srikant,"Congestion Control for Fair Resource Allocation in Networks with Multicast Flows", *Proceedings of Conference on Decision and Control (CDC) 2001*, Orlando, USA, December 2001.

[9] E. Graves, R. Srikant, D. Towsley, "Decentralized Computation of Weighted Max-Min Fair Bandwidth Allocation in Networks with Multicast Flows", *Proceedings of Tyrrhenian International Workshop on Digital Communications (IWDC) 2001*, Taormina, Italy, September 2001.

[10] K. Kar, S. Sarkar, L. Tassiulas, "A Simple Rate Control Algorithm for Maximizing Total User Utility", *Proceedings of Infocom 2001*, Anchorage, USA, April 2001.

[11] K. Kar, S. Sarkar, L. Tassiulas, "Optimization Based Rate Control for Multirate Multicast Sessions", *Proceedings of Infocom 2001*, Anchorage, USA, April 2001.

[12] K. Kar, S. Sarkar, L. Tassiulas, "A Low-Overhead Rate Control Algorithm for Maximizing Aggregate Receiver Utility for Multirate Multicast Sessions", Technical Report ISR TR 2000-52, Institute for Systems Research, University of Maryland, 2000.

[13] F. P. Kelly,"Charging and Rate Control for Elastic Traffic", *European Transactions on Telecommunications*, vol. 8, no. 1, 1997, pp. 33-37.

[14] F. Kelly, A. Maulloo, D. Tan,"Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability ", *Journal of Operations Research Society*, vol. 49, no. 3, 1998, pp. 237-252.

[15] F. Kishino, K. Manabe, Y. Hayashi, H. Yasuda, "Variable Bit-Rate Coding of Video Signals for ATM Networks", *IEEE Journal on Selected Areas In Communications*, vol. 7, no. 5, June 1989.

[16] R. La, V. Anantharam,"Charge-Sensitive TCP and Rate Control in the Internet", *Proceedings of Infocom 2000*, Tel Aviv, Israel, March 2000.

[17] S. Low, D. E. Lapsley, "Optimization Flow Control, I: Basic Algorithm and Convergence", *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, December 1999.

[18] X. Li, S. Paul, M. Ammar, "Layered Video Multicast with Retransmission (LVMR): Evaluation of Hierarchical rate control", *Proceedings of IEEE Infocom 1998*, March 1998.

[19] L. Massoulie, J. Roberts, "Bandwidth Sharing: Objectives and Algorithms", *Proceedings of IEEE Infocom 1999*, New York, USA, March 2000.

[20] S. McCanne, V. Jacobson, M. Vetterli, "Receiver-Driven Layered Multicast", *Proceedings of ACM Sigcomm '96*, Stanford, CA, September 1996.

[21] B. T. Poljak, "A General Method of Solving Extremum Problems", *Soviet Math Doklady*, vol. 8, no. 3, 1967, pp. 593-597.

[22] R. T. Rockafellar, *Convex Analysis*, Princeton Univ. Press, 1970.

[23] D. Rubenstein, J. Kurose, D. Towsley. "The Impact of Multicast Layering on Network Fairness", *Proceedings of ACM Sigcomm '99, Cambridge, MA, September, 1999*.

[24] N. Z. Shor, *Minimization Methods for Non-differentiable Functions*, Springer-Verlag, 1985.

[25] S. Sarkar, L. Tassiulas, "Fair Allocation of Utilities in Multirate Multicast Networks", *Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing*, 1999.

[26] S. Sarkar, L. Tassiulas, "Distributed Algorithms for Computation of Fair Rates in Multirate Multicast Trees", *Proceedings of IEEE Infocom 2000*, Tel Aviv, Israel, March 2000.

[27] S. Sarkar, L. Tassiulas, "Fair Allocation of Discrete Bandwidth Layers in Multicast Networks", *Proceedings of IEEE Infocom 2000*, Tel Aviv, Israel, March 2000.

[28] S. Kunniyur, R. Srikant,"End-to-End Congestion Control Schemes: Utility Functions, Random Losses and ECN Marks", *Proceedings of Infocom 2000*, March 2000.

[29] T. Turletti, J. C. Bolot, "Issues with Multicast Video Distribution in Heterogeneous Packet Networks", *Packet Video Workshop*, 1994.

Appendix I: Subgradients and their properties

Definition 1: [24] *(Subgradient and Subdifferential)* Consider a convex and continuous function f defined on a convex set $F \subseteq \Re^k$. Then a vector $w_0 \in \Re^k$ is called a *subgradient* of f at a point $x_0 \in F$ if it satisfies

$$f(x) - f(x_0) \quad \geq \quad \langle w_0, x - x_0 \rangle \qquad \forall x \in F$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product.

The *subdifferential* of f at $x_0 \in F$, denoted by $\partial f(x_0)$, is the set of all subgradients of f at x_0, i.e.,

$$\partial f(x_0) = \{w_0 \in \Re^k : f(x) - f(x_0) \geq \langle w_0, x - x_0 \rangle \quad \forall x \in F\}$$

In general, subgradient at a point may be non-unique. However, if $\nabla f(x_0)$ exists, then $\partial f(x_0) = \{\nabla f(x_0)\}$.

Next we state two properties of subgradients (see Theorems 1.12 & 1.13 of [24]), which will be useful in our analysis.

Lemma 1: Let I be a finite index set. Let $f_i, i \in I$, be convex, continuous functions defined on a convex set F. Let $x_0 \in F$, and $w_{i0} \in \partial f_i(x_0), i \in I$.
(a) Let $f(x) = \sum_{i \in I} a_i f_i(x)$, where $a_i \geq 0$, $i \in I$. Then $\sum_{i \in I} a_i w_{i0} \in \partial f(x_0)$.
(b) Let $f(x) = \max_{i \in I} f_i(x)$. Define $\tilde{I}(x) = \{i \in I : f_i(x) = f(x)\}$. Then $w_{i0} \in \partial f(x_0)$, for all $i \in \tilde{I}(x_0)$.

Appendix II: Proof of Theorem 1

First we state a few lemmas that would be used in the proof of Theorem 1. For each $l \in L$, define $g_l : \Re_+^{|R|} \rightarrow \Re$ as $g_l(x) = \max_{r \in S_l \cap R_m} x_r - c_l \;\; = \;\; \sum_{r \in \tilde{S}_l} x_r - c_l$. Thus the capacity constraint for link l can be simply written as $g_l(x) \leq 0$.

Now consider the following problem

$$\tilde{\mathbf{P}}: \qquad \begin{array}{ll} \text{maximize} & \displaystyle\sum_{r \in R} U_r(x_r) - K \sum_{l \in L} \max\{0, g_l(x)\} \\ \text{subject to} & \quad x_r \in X_r \quad \forall r \in R \end{array}$$

where K is a non-negative constant.

Let $\tilde{x}^*$ denote the optimal solution of $\tilde{\mathbf{P}}$ (note that the uniqueness of this optimum is guaranteed by Assumption 3).

Lemma 2: If $K > A\bar{R}$, then $\tilde{x}^* = x^*$.

The above result is fairly intuitive. Comparing problems $\mathbf{P}$ and $\tilde{\mathbf{P}}$, we see that the link constraints in $\mathbf{P}$ have been transferred to the objective function in $\tilde{\mathbf{P}}$. The term $K \max\{0, g_l(x)\}$ can be interpreted as the penalty associated with the violation of the capacity constraint of link l. Thus the above lemma states that when the penalty associated with constraint violations is sufficiently large, the optimal solution set of the unconstrained problem $\tilde{\mathbf{P}}$ becomes the same as that of $\mathbf{P}$. Results similar to the one stated above have been observed in the optimization literature. For example, see [3] (and the references therein) and [24] (Theorem 4.2), where the above result is shown to hold for a slightly different lower bound on K. A rigorous proof of Theorem 2 is quite complex, and is stated in [12].

Now define a function $\tilde{U} : \Re_+^{|R|} \rightarrow \Re$ as $\tilde{U}(x) = \sum_{r\in R} U_r(x_r) - K\sum_{l\in L}\max\{0, g_l(x)\}$. Thus $\tilde{\mathbf{P}}$ is the problem of maximizing $\tilde{U}(x)$ subject to $x \in X_R$. Let $u_r^{(n)} = U_r^{'}(x_r^{(n)})$, and $v_r^{(n)} = u_r^{(n)} - Kp_r^{(n)}$. Let $u^{(n)} = (u_r^{(n)}, r \in R)$ denote the vector of the utility derivatives, and $p^{(n)} = (p_r^{(n)}, r \in R)$ denote the vector of the congestion penalties. Let $v^{(n)} = (v_r^{(n)}, r \in R) = u^{(n)} - Kp^{(n)}$. Note that since $p^{(n)}$ depends on the penalty splitting factors $\alpha_r^{(n)}, r \in \tilde{R} \setminus \tilde{R}_\Omega$ (cf. (5)), so does $v^{(n)}$.

Lemma 3: If $\alpha_r^{(n)}, r \in \tilde{R} \setminus \tilde{R}_\Omega$ satisfy (8)-(10), then $v^{(n)} \in \partial\tilde{U}(x^{(n)})$.

Proof: Define $P_l(x) = \max\{0, g_l(x)\}$, and $P(x) = \sum_{l\in L} P_l(x)$. Therefore, $\tilde{U}(x) = U(x) - KP(x)$.

We will first show that $p^{(n)} \in \partial P(x^{(n)})$. Define $g_{lm}(x) = \max_{S_l\cap R_m} x_r$. Then $g_l(x) = \sum_{m\in M} g_{lm}(x)$. Now for every $l \in L$, let $\beta_l^{(n)} = (\beta_{l,r}^{(n)}, r \in R)$ be a $|R|$-dimensional vector whose components are given as

$$\beta_{l,r}^{(n)} = \begin{cases} 0 & \text{if } r \notin S_l \\ \prod_{r''\in Q_r\setminus Q_{r'}} \alpha_{r''}^{(n)} & \text{if } r \in S_l \end{cases} \quad (13)$$

where $r^{'}$ is junction/receiver node immediately downstream of link l in the multicast tree to which r belongs. Note that since the splitting factors are either 0 or 1, the components $\beta_{l,r}^{(n)}$ and are also either 0 or 1. Then combining (5) and (4), it is easy to see that $p^{(n)}$ can be written as

$$p^{(n)} = \sum_{l\in L} \beta_l^{(n)} \varepsilon_l^{(n)} \quad (14)$$

Now for every $l \in L$ and every $m \in M$, let $\tilde{\beta}_{lm}^{(n)} = (\tilde{\beta}_{lm,r}^{(n)}, r \in R)$ be a $|R|$-dimensional vector whose components are given as

$$\tilde{\beta}_{lm,r}^{(n)} = \begin{cases} \beta_{l,r}^{(n)} & \text{if } r \in R_m \\ 0 & \text{if } r \notin R_m \end{cases} \quad (15)$$

Next we show that $\tilde{\beta}_{lm}^{(n)} \in \partial(\max_{r\in S_l\cap R_m} x_r^{(n)}) = \partial g_{lm}(x^{(n)})$. Consider any $l \in L$ and any $m \in M$. Now consider the two cases:

(i) $S_l \cap R_m = \phi$: In this case, it is easy to see that $\tilde{\beta}_{lm}^{(n)}$ is a zero vector. Thus $\tilde{\beta}_{lm}^{(n)} \in \partial(\max_{r\in S_l\cap R_m} x_r^{(n)})$, trivially.

(ii) $S_l \cap R_m \neq \phi$: From (8) and (10), it is easy to see that only one component of $\tilde{\beta}_{lm}^{(n)}$ is 1, and all the rest are 0. (Note that for $\tilde{\beta}_{lm,r}^{(n)}$ to be 1, the splitting factors of all the junction/receiver nodes in the downstream path from the link l to receiver r has to be 1. It is easy to see that this will happen for exactly one receiver of multicast group m). Let $r^{'} \in R_m$ be such that $\tilde{\beta}_{lm,r'}^{(n)} = 1$. Then using (9), it is also easy to show that $x_{r'}^{(n)} = \max_{r\in S_l\cap R_m} x_r^{(n)}$. Then, from Lemma 1 (b), it follows that $\tilde{\beta}_{lm}^{(n)} \in \partial(\max_{r\in S_l\cap R_m} x_r^{(n)})$.

From cases (i) and (ii), it follows that $\tilde{\beta}_{lm}^{(n)} \in \partial g_{lm}(x^{(n)})$ for all $l \in L$ and $m \in M$.

Note that $\beta_l^{(n)} = \sum_{m\in M} \tilde{\beta}_{lm}^{(n)}$. Hence, from Lemma 1 (a), it follows that $\beta_l^{(n)} \in \partial(\sum_{m\in M} g_{lm}(x^{(n)}) = \partial g_l(x^{(n)})$. Using this fact, and Lemma 1 (b), it is easy to show that $\beta_l^{(n)}\varepsilon_l^{(n)} \in \partial(\max\{0, g_l(x^{(n)})\}) = \partial P_l(x^{(n)})$. Then from (14), and using Lemma 1 (a), it follows that $p^{(n)} \in \partial(\sum_{l\in L} P_l(x^{(n)})) = \partial P(x^{(n)})$.

It is straightforward to show that $u^{(n)} \in \partial U(x^{(n)})$. Therefore, using Lemma 1 (a), $v^{(n)} = u^{(n)} - Kp^{(n)} \in \partial U(x^{(n)}) - KP(x^{(n)}) = \partial\tilde{U}(x^{(n)})$. □

Proof of Theorem 1: We will first show that the sequence $\{x^{(n)}\}$ converges to $\tilde{x}^*$. .

Choose an arbitrary $\delta > 0$. Let $\delta' = (\delta/2)$. For any $\epsilon^{'} > 0$, define $D_{\epsilon'}$ as $D_{\epsilon'} = \{x : x \in X_R, \tilde{U}(x) \geq \tilde{U}^* - \epsilon^{'}\}$. It follows from Theorem 27.2 of [22] that there exists an $\epsilon = \epsilon(\delta') > 0$ such that

$$D_\epsilon \quad \subset \quad \{x : ||x - \tilde{x}^*|| \leq \delta'\} \quad (16)$$

Consider an n for which $x^{(n)} \notin D_\epsilon$. Therefore, $\tilde{U}(x^{(n)}) < \tilde{U}^* - \epsilon$. Since $v^{(n)} \in \partial\tilde{U}(x^{(n)})$ (from Lemma 3), and using the definition of a subgradient (Definition 1), we obtain

$$\left\langle v^{(n)}, x^{(n)} - \tilde{x}^* \right\rangle \leq \tilde{U}(x^{(n)}) - \tilde{U}(\tilde{x}^*) < -\epsilon \quad (17)$$

Note that $||u^{(n)}||$ is upper-bounded (from Assumption 2), and so are K and $||p^{(n)}||$. Therefore, $||v^{(n)}|| = ||u^{(n)} - Kp^{(n)}||$ is also upper-bounded. Let $||v^{(n)}|| \leq \tilde{A}$ for all n. Also note that the rate update procedure for the receiver nodes, as stated in (6), can be compactly stated as: $x^{(n+1)} = [\ x^{(n)} + \lambda_n v^{(n)}\ ]_{X_R}$. Using these facts, and (17), we obtain,

$$\begin{aligned} ||x^{(n+1)} - \tilde{x}^*||^2 &= ||[\ x^{(n)} + \lambda_n v^{(n)}\ ]_{X_R} - \tilde{x}^*||^2 \\ &\leq ||x^{(n)} + \lambda_n v^{(n)} - \tilde{x}^*||^2 \quad (18) \\ &= ||x^{(n)} - \tilde{x}^*||^2 + \lambda_n^2||v^{(n)}||^2 \\ &\quad +2\lambda_n \left\langle x^{(n)} - \tilde{x}^*, v^{(n)} \right\rangle \\ &< ||x^{(n)} - \tilde{x}^*||^2 + \tilde{A}^2\lambda_n^2 - 2\epsilon\lambda_n \quad (19) \end{aligned}$$

Note that (18) follows from the fact that $\tilde{x}^* \in X_R$ (use projection theorem).

Since $\lambda_n \rightarrow 0$, $\lambda_n \leq (\epsilon/\tilde{A}^2)$ when n is sufficiently large. For all such n, from (19), we get

$$||x^{(n+1)} - \tilde{x}^*||^2 \quad < \quad ||x^{(n)} - \tilde{x}^*||^2 - \epsilon\lambda_n \quad (20)$$

Now, for the sake of contradiction, let us assume that there exists a $N'_\epsilon < \infty$ such that $x^{(n)} \notin D_\epsilon$ for all $n \geq N'_\epsilon$. Therefore, there exists $N_\epsilon \geq N'_\epsilon$ be such that (20) holds for all $n \geq N_\epsilon$. Summing up the inequalities obtained from (20) for $n = N_\epsilon$ to $N_\epsilon + m$, we obtain

$$||x^{(N_\epsilon+m+1)} - \tilde{x}^*||^2 \quad < \quad ||x^{(N_\epsilon)} - \tilde{x}^*||^2 - \epsilon\sum_{n=N_\epsilon}^{N_\epsilon+m} \lambda_n \quad (21)$$

which implies that $||x^{(N_\epsilon+m+1)} - \tilde{x}^*|| \rightarrow -\infty$ as $m \rightarrow \infty$, since $\sum \lambda_n$ diverges. This is impossible, since $||x^{(N_\epsilon+m+1)} - \tilde{x}^*|| \geq 0$. Hence our assumption was incorrect. Hence, there exists an infinite sequence $n_{1,\epsilon} < n_{2,\epsilon} < n_{3,\epsilon} < ...$ such that $x^{(n_{i,\epsilon})} \in D_\epsilon$ for all $i = 1, 2, 3, ...$. This implies that there exists an i_1 such that (20) holds for all $n \geq n_{i_1,\epsilon}$. Also, since $\lambda_n \to 0$, there exists and i_2 such that $\lambda_n \leq (\delta' / \tilde{A})$ for all $n \geq n_{i_2,\epsilon}$.

Let $i' = \max(i_1, i_2)$. We show that $||x^{(n)} - \tilde{x}^*|| \leq \delta$ for all $n \geq n_{i',\epsilon}$. Pick any $n \geq n_{i',\epsilon}$. There can be three cases:

Case 1 : $n = n_{j,\epsilon}$ for some $j \geq i'$: In this case, $x^{(n)} \in D_\epsilon$. From (16), it trivially follows that $||x^{(n)} - \tilde{x}^*|| \leq \delta' < \delta$.

Case 2 : $n = n_{j,\epsilon} + 1$ for some $j \geq i'$: In this case, $x^{(n)} = x^{(n_{j,\epsilon}+1)} = [\, x^{(n_{j,\epsilon})} + \lambda_{n_{j,\epsilon}} v^{(n_{j,\epsilon})} \,]_{X_R}$. Thus

$$\begin{aligned} ||x^{(n)} - x^{(n_{j,\epsilon})}|| &= ||[\, x^{(n_{j,\epsilon})} + \lambda_{n_{j,\epsilon}} v^{(n_{j,\epsilon})} \,]_{X_R} - x^{(n_{j,\epsilon})}|| \\ &\leq ||x^{(n_{j,\epsilon})} + \lambda_{n_{j,\epsilon}} v^{(n_{j,\epsilon})} - x^{(n_{j,\epsilon})}|| \\ &= \lambda_{n_{j,\epsilon}} ||v^{(n_{j,\epsilon})}|| \;\leq\; \tilde{A}\lambda_{n_{j,\epsilon}} \;\leq\; \delta' \end{aligned} \tag{22}$$

From (22) and the fact that $||x^{(n_{j,\epsilon})} - \tilde{x}^*|| \leq \delta'$ (Case 1), we get

$$\begin{aligned} ||x^{(n)} - \tilde{x}^*|| &\leq ||x^{(n_{j,\epsilon})} - \tilde{x}^*|| + ||x^{(n)} - x^{(n_{j,\epsilon})}|| \\ &\leq \delta' + \delta' \;=\; 2\delta' \;=\; \delta \end{aligned} \tag{23}$$

Case 3 : $n_{j,\epsilon} + 1 < n < n_{j+1,\epsilon}$ for some $j \geq i'$: Note that $x^{(n')} \notin D_\epsilon$ for all n' satisfying $n_{j,\epsilon} < n' < n_{j+1,\epsilon}$. From (20), it follows that $||x^{(n'+1)} - \tilde{x}^*|| < ||x^{(n')} - \tilde{x}^*||$. Summing up these inequalities obtained for $n' = n_{j,\epsilon} + 1$ to $n - 1$, we obtain $||x^{(n)} - \tilde{x}^*|| < ||x^{(n_{j,\epsilon}+1)} - \tilde{x}^*||$. Since $||x^{(n_{j,\epsilon}+1)} - \tilde{x}^*|| \leq \delta$ (Case 2), it follows that $||x^{(n)} - \tilde{x}^*|| \leq \delta$.

From cases 1, 2, 3,, if follows that $||x^{(n)} - \tilde{x}^*|| \leq \delta$ for all $n \geq n_{i',\epsilon}$. By virtue of the arbitrariness of δ, it follows that $\lim_{n\to\infty} ||x^{(n)} - \tilde{x}^*|| = 0$. Now, from Lemma 2, it follows that if $K > A\bar{R}$, then $\lim_{n\to\infty} ||x^{(n)} - x^*|| = 0$. $\square$

Koushik Kar received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, in 1997. He received the M.S. degree in electrical and computer engineering from the University of Maryland, College Park, in 1999, where he is currently pursuing his Ph.D. degree. His research interests include scheduling in high-speed switches, routing and congestion control, and fairness and pricing issues in communication networks.

Saswati Sarkar received Master of Engineering in electrical and communication engineering from the Indian Institute of Science in 1996, and Ph.D. in electrical and computer engineering from University of Maryland, College Park, in 2000. She is currently an Assistant Professor in the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia. Her research interests are in resource allocation and performance analysis in communication networks.

Leandros Tassiulas was born in 1965 in Katerini, Greece. He received the Diploma in electrical engineering from the Aristotelian University of Thessaloniki, Greece, in 1987, and the M.S. and Ph. D. degrees in electrical engineering from the University of Maryland, College Park in 1989 and 1991, respectively. From 1991 to 1995 he was an Assistant Professor in the Department of Electrical Engineering, Polytechnic University, Brooklyn, NY. In 1995 he joined the Department of Electrical Engineering, University of Maryland, where he is now an Associate Professor. He holds a joint appointment with the Institute for Systems Research and is the member of the Center for Satellite and Hybrid Communication Networks, established by NASA. His research interests are in computer and communication networks, with emphasis on wireless communications (terrestrial and satellite systems) and high speed network architectures and management, in control and optimization of stochastic systems in parallel and distributed processing. Dr. Tassiulas coauthored a paper that received the INFOCOM 1994 Best Paper Award. He received a National Science Foundation (NSF) Research Initiation Award in 1992, and NSF Faculty Early Career Development Award in 1995, and an Office of Naval Research Young Investigator Award in 1997.