October 2002

A Framework for Routing and Congestion Control for Multicast Information Flows

Saswati Sarkar
University of Pennsylvania, swati@seas.upenn.edu

Leandros Tassiulas
University of Maryland

A Framework for Routing and Congestion Control for Multicast Information Flows

Abstract
We propose a new multicast routing and scheduling algorithm called multipurpose multicast routing and scheduling algorithm (MMRS). The routing policy load balances among various possible routes between the source and the destinations, basing its decisions on the message queue lengths at the source node. The scheduling is such that the flow of a session depends on the congestion of the next hop links. MMRS is throughput optimal. In addition, it has several other attractive features. It is computationally simple and can be implemented in a distributed, asynchronous manner. It has several parameters which can be suitably modified to control the end-to-end delay and packet loss in a topology-specific manner. These parameters can be adjusted to offer limited priorities to some desired sessions. MMRS is expected to play a significant role in end-to-end congestion control in the multicast scenario.

Keywords
Multicast, routing, scheduling, stability, throughput

A Framework for Routing and Congestion Control for Multicast Information Flows

Saswati Sarkar, *Member, IEEE,* and Leandros Tassiulas

***Abstract*—We propose a new multicast routing and scheduling algorithm called multipurpose multicast routing and scheduling algorithm (MMRS). The routing policy load balances among various possible routes between the source and the destinations, basing its decisions on the message queue lengths at the source node. The scheduling is such that the flow of a session depends on the congestion of the next hop links. MMRS is throughput optimal. In addition, it has several other attractive features. It is computationally simple and can be implemented in a distributed, asynchronous manner. It has several parameters which can be suitably modified to control the end-to-end delay and packet loss in a topology-specific manner. These parameters can be adjusted to offer limited priorities to some desired sessions. MMRS is expected to play a significant role in end-to-end congestion control in the multicast scenario.**

***Index Terms*—Multicast, routing, scheduling, stability, throughput.**

I. Introduction

MULTICASTING provides an efficient way of transmitting data from a sender to a group of receivers. A single source node or a group of source nodes sends identical messages simultaneously to multiple destination nodes. Unicast and broadcast to the entire network are special cases of multicast. Multicast applications include collaborative applications like audio or video teleconferencing, video-on-demand services, distributed databases, distribution of software, financial information, electronic newspapers, billing records, medical images, weather maps, experimental data, and distributed interactive simulation (DIS) activities such as tank battle simulations. Currently, several distributed systems such as the V System [3] and the Andrew distributed computing environment [27], popular protocol suites like Sun's broadcast Remote Procedure Cell (RPC) service [24] and IBM's NetBIOS [12] all use multicasting [5]. Multicasting has been used primarily in the Internet, but future asynchronous transmission mode (ATM) networks are likely to deploy multicasting on a large scale, particularly in applications like broadcast video, videoconferencing, multiparty telephony, and workgroup applications [4].

Manuscript received February 22, 1999; revised March 13, 2002. The work of S. Sarkar was supported in part by the National Science Foundation under Grant ANI01-06984. The material in this paper was presented in part at IEEE INFOCOM 99, New York, April 1999.

S. Sarkar is with the Department of Electrical Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: swati@ee.upenn.edu).

L. Tassiulas is with the Department of Electrical and Computer Engineering and Institute for Systems Research, University of Maryland, College Park, MD 20742 USA (e-mail: leandros@isr.umd.edu).

Communicated by V. Anantharam, Associate Editor for Communication Networks.

Publisher Item Identifier 10.1109/TIT.2002.802619.

There may be more than one possible route between a source and a group of destinations. More than one multicast session may share the same link. This gives rise to the fundamental issues of routing and scheduling in multicast communication. Until now, scheduling in multicast networks has primarily been best effort service. With increase in traffic, congestion control and class based scheduling would be required to improve performance. MMRS uses a scheduling policy which can be tuned to distinguish among various classes.

A significant amount of research has been directed toward multicast routing. Tree construction is the commonly used approach in solving the multicast routing problem. Multicast trees minimize data replication; messages need only be replicated at the forking nodes. This differs from the multicast attained through multiple unicasts where every unicast requires a copy of the message. Multiple unicasts may result in many copies of the same message traversing the same network links and thus waste network resources. Multicast trees can be broadly classified into shortest path trees (SPTs), also known as source-based trees and group shared trees [35]. SPT is currently used in distance-vector multicast routing protocol (DVMRP) [5] for Internet multicast traffic on the virtual multicast backbone (Mbone) network [8], [11] and multicast extensions for open shortest path first OSPF(MOSPF) [20]. The core-based tree (CBT) [1] uses a group shared tree also known as the center-based tree. Recently, some hybrid routing protocols like the protocol-independent multicast (PIM) [6] and the multicast Internet protocol (MIP) [21] have been proposed. These allow the system to switch modes between SPT and group shared trees.

None of these routing policies support more than one tree per source–destination pair at a time. Thus, only a single route is determined depending upon the topology and then the messages are sent along the same route till the topology or the destination group changes. PIM and MIP allow the system to switch to a different tree mode, but not on a very dynamic basis. For example, PIM supports center-based trees for low data rate sources or sparse multicast groups and allows receivers to switch over to an SPT mode when low delay is important or the multicast group is densely populated. When the switchover takes place, the CBT is modified to replace the core-based routes by the shortest path routes between a source and some destinations. Thus, these protocols do not provide for *load balancing*, i.e., having more than one possible tree simultaneously and allowing the system to dynamically choose among them, the routing decisions being taken not too infrequently. Load balancing can meet very effectively the technical challenge of minimizing the link loads given some network load and thus serve as an important weapon for congestion control. This would increase throughput and decrease delay and message loss in the network. Congestion control is critically important in various real-time resource-expensive applications in internet and various data applications and other services like local-area netsork (LAN) emulation in ATM ABR services.

Load balancing may cause out-of-order delivery of messages. Some applications do not need ordered delivery of messages, e.g., many audio and video conferencing applications like vic [17], vat [33], nv [9], rat [10], and freephone [2] (audio packets are reordered in the application play out buffer) and workspace applications like Wb [36]. Application-level protocols can be used to enforce a particular delivery order, if necessary. However, ATM applications need ordered delivery. So load balancing can be done per session. Besides, out-of-order delivery can be reduced by choosing large routing decision intervals, depending upon the requirement of the application. Load balancing would increase the number of routing table entries in the routers because more than one routing tree may be used simultaneously; however, the network can choose the number of simultaneously active trees depending upon the router memories and a tradeoff can be reached.

To do load balancing effectively, the network needs to route a message through a tree selected judiciously amongst many possible trees. The usual approach is to choose the least total-cost tree among all possible trees between a source and a group of destinations. The tree cost is usually the sum of the link costs and the link costs may be a measure of a number of possible parameters, e.g., actual or anticipated congestion, error rate, propagation delay, etc. Therefore, computation of a good or rather near-optimal route based on the total tree cost at reasonably regular intervals is computationally expensive if the set of possible trees is even a moderately large subset of all possible trees.

We propose a novel routing and scheduling policy which retains the benefits of load balancing, yet overcomes the above difficulty. We call this policy the *Multipurpose Multicast Routing and Scheduling* policy (MMRS). The routing scheme takes routing decisions at possibly random, but bounded intervals and bases the decision *only* on the message queue lengths of the different possible trees at the source node. It takes routing decisions in favor of the tree with the least queue length or rather the least scaled queue length at the source node. The scheduling policy has been so chosen that this quantity represents the congestion state of the tree. At any link, the scheduling policy gives priority to those sessions which have the congestion[1] at the downstream[2] buffers less than that at the upstream buffer, and does not serve any session at a link if all the sessions have the downstream buffers more congested than the corresponding upstream buffer at the link. Thus, any downstream congestion is eventually reflected at the source node. The source node will soon have a large message queue for the tree and the routing policy would route messages through other trees with less message queue length at the source node and hence better congestion state throughout. This attains load balancing without any intensive computation based on the state of the entire tree.

MMRS has various parameters. If the parameters are properly chosen, the scheduling policy becomes computationally simply and needs only local information[3] and not the state of the entire network. The parameters can be further adjusted to obtain low-delay and low-message-loss characteristics. The parameters can be modified suitably to give limited priority to selected flows. We discuss these in detail later. Finally, we would like to point out another significant advantage of MMRS. Since the message queue lengths at the source reflect the congestion status of the possible routes and hence that of the session, end-to-end congestion control measures may be based on this observation. Thus, the message queue lengths at the source of the session give *implicit* feedback about the congestion state of the paths followed. This is a significant advantage for multicast applications because explicit end-to-end feedbacks often lead to feedback *implosion*. Thus, MMRS has various convenient features which render it attractive from the implementational point of view.

MMRS attains maximum possible throughput in an arbitrary multicast network. MMRS retains this throughput optimality even if the routing and the scheduling decisions are not taken every slot but at bounded intervals. Also, as we point out later, MMRS is flexible and can be tuned to suit the hardware/software limitations of many real-life multicast networks.

The rest of the paper is organized as follows. We describe the multicast network model in Section II. Section III describes the general routing, scheduling and congestion control problem in multicast networks. Section IV describes MMRS in detail. We discuss some interesting aspects of MMRS in Section V. We describe our stability criterion for any network in Section VI and prove the maximum throughput property of MMRS in Section VII. We prove a necessary condition for stability in a multicast network in Section VIII.

II. Multicast Transport Network Model

The network is modeled by an arbitrary topology directed graph G, where $G = (V, E)$. Set V represents the set of nodes and E the set of directed links. 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. There are N multicast sessions $(v_1, U_1), \dots, (v_N, U_N)$. A collection $\mathcal{T}_n$ of eligible multicast trees is prespecified as the trees through which session n traffic can be transported, $|\mathcal{T}_n| = M_n$. The mth tree in $\mathcal{T}_n$ can be described by an indicator vector, $T_n^m = (t_e, e \in E)$, where $t_e = 1$ if edge e is a part of T_n^m and $t_e = 0$ otherwise. Fig. 1 illustrates the model.

We do not impose any particular structure on $\mathcal{T}_n$; $\mathcal{T}_n$ can consist of all directed trees between source v_n and the set of destinations U_n. However, this generates huge routing table entries at the routers in a virtual circuit-like scenario. This is because the

[1]Intuitively, congestion at a buffer depends on the queue length at the buffer. More technically, it is some quantity which our scheduling policy uses. We describe our scheduling policy more rigorously later.

[2]"Downstream" here means destination of a link and "upstream" means source of a link.

[3]Our scheduling policy requires the knowledge of the queue lengths at both source and destination of a link whereas the scheduler generally resides at the source. We assume that the destination communicates this information to the source. We discuss this in detail later.

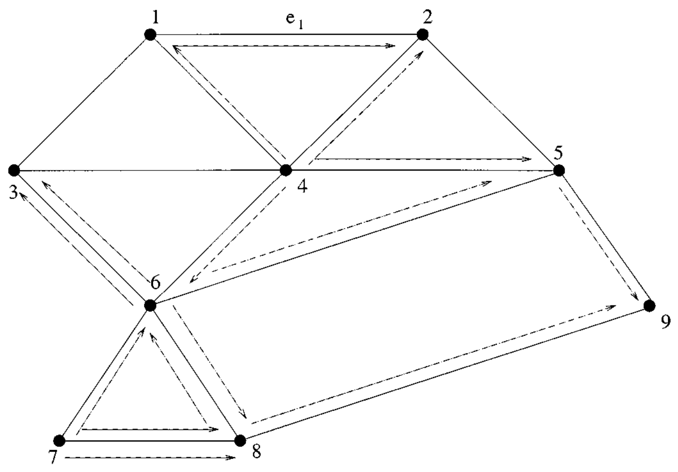


Fig. 1. There are two multicast sessions, sessions 1 and 2. Session 1 has $(v_1, U_1) = (4, \{2, 5, 9\})$ and session 2 has $(v_2, U_2) = (7, \{3, 6, 8\})$. Here, $\mathcal{T}_1 = \{T_1, T_2\}, \mathcal{T}_2 = \{T_3, T_4\}$.

$$T_1 = \{(4, 2), (4, 5), (5, 9)\}$$
$$T_2 = \{(4, 1), (1, 2), (4, 6), (6, 5), (6, 8), (8, 9)\}$$
$$T_3 = \{(7, 8), (7, 6), (6, 3)\}$$
$$T_4 = \{(7, 8), (8, 6), (6, 3)\}$$

where (v_1, v_2) represents the directed edge from v_1 to v_2. Note that neither $\mathcal{T}_1$ nor $\mathcal{T}_2$ contains all possible trees for the respective sessions, e.g.,

$$T_5 = \{(4, 3), (3, 1), (1, 2), (4, 6), (6, 5), (6, 8), (8, 9)\}$$

can also carry session 1 traffic but is not included in $\mathcal{T}_1$. The indicator vector for T_1 is $(1, 1, 1, 0, \ldots, 0)$, if $(4, 2), (4, 5), (5, 9)$ correspond to the first three edges.

total number of possible multicast trees is considerably large, and in a virtual circuit-like scenario the packets contain only route identifiers and the routers need to maintain the lookup tables for these identifiers. Therefore, it is realistic to assume that $\mathcal{T}_n$ will be a proper subset of the set of all directed trees. The actual size of this subset should depend on the available router memories. In a datagram-like scenario, packets contain the entire path to be followed in the header and the routers need only have entries regarding the currently active trees. Thus, a memory constraint may not force $\mathcal{T}_n$ to be a proper subset of the set of all directed trees. However, there may be other constraints, e.g., all multicast trees may not satisfy the requirements of a session n, e.g., session n may demand certain quality of service guarantees in terms of end-to-end delay bounds and delay jitters. For instance, during a teleconference it is important that all participants hear the speaker around the same time, else the communication lacks the feeling of an interactive face-to-face discussion [22]. In high-speed environments, the end-to-end delays depend primarily on the propagation delays, and this rules out certain trees. Therefore, $\mathcal{T}_n$ can consist of those trees which satisfy the requirements of session n, or a proper subset thereof.

Intuitively, the necessary condition for system stability is that the sum of the traffic arrival rates in all the trees of the same or different sessions passing through a link e does not exceed the link capacity, i.e.,

$$\sum_{n=1}^{N} \sum_{m=1}^{M_n} a_n^m T_n^m \le (C_1, \ldots, C_{|E|}) \quad \text{(capacity condition)}$$

where a_n^m is the traffic arrival rate in the mth tree of the nth session, T_n^m is the indicator vector for the mth tree of the nth session, and C_e is the capacity of the eth link. It is not obvious whether this condition guarantees stability in any arbitrary network. A major contribution of this paper is to prove that it is indeed so, that is, if we allocate the resources as per MMRS, the necessary condition for system stability with strict inequality, turns out to be sufficient as well. It is in this sense that MMRS maximizes throughput. We investigate this issue later.

III. Routing, Scheduling, and Congestion Control in Multicast Networks

Session n traffic can reach the destinations through one of the many possible trees in $\mathcal{T}_n$, e.g., incoming session 1 traffic can reach its destination via trees T_1 and T_2 in Fig. 1. The resource-allocation policy decides the appropriate tree. It should *load balance*, that is, respond to congestion in the currently active trees and route incoming traffic to relatively lightly loaded trees. It is also expected to compute the congestion status of the trees efficiently. This is not likely to be the case if the decision is based on any suitably defined weight of the entire tree as per the discussion in Section I.

In general, the trees of the same or different sessions would overlap on the links and at most one of them can be served at one time, e.g., trees T_3 and T_4 overlap on link $(7, 8)$ in Fig. 1. Therefore, the resource-allocation policy also decides the trees that should be scheduled in the links. Intuitively, it should "spread out" the congestion in the network, i.e., if an upstream node of a session is heavily congested, while the downstream node is not, then traffic from that session should be served on the link. This would decrease the congestion in the heavily loaded upstream node at the expense of increasing the congestion at the lightly loaded downstream node.

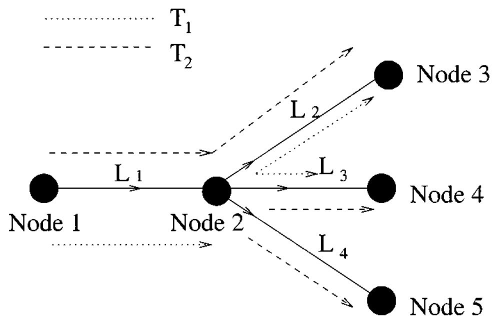
Another interesting question worth investigating is how often the routing and scheduling decisions should be taken. These decisions can be taken at intervals of fixed or bounded length. Size of the decision intervals may or may not depend on the queue lengths at the nodes.

To the best of our knowledge, there does not exist any generalized routing and scheduling policy which effectively addresses the above issues in multicast networks. Some of these issues have been addressed in the unicast scenario in [30]. However, multicast networks are inherently different from unicast networks because of "traffic multiplication." The same unit of traffic is transmitted from a multicast node across various links. Thus, the traffic flow rate in the network exceeds the arrival rate. The issue of routing is also different in the unicast context. We will discuss the policy we propose in the perspective of existing work in the unicast and broadcast context in Section IX.

The MMRS addresses all of the above issues in a flexible manner. We describe MMRS in the following section. MMRS consists of various parameters which can be adjusted to suit the requirements of various networks. Thus MMRS may also be thought of as a class of routing and scheduling policies rather than a single policy.

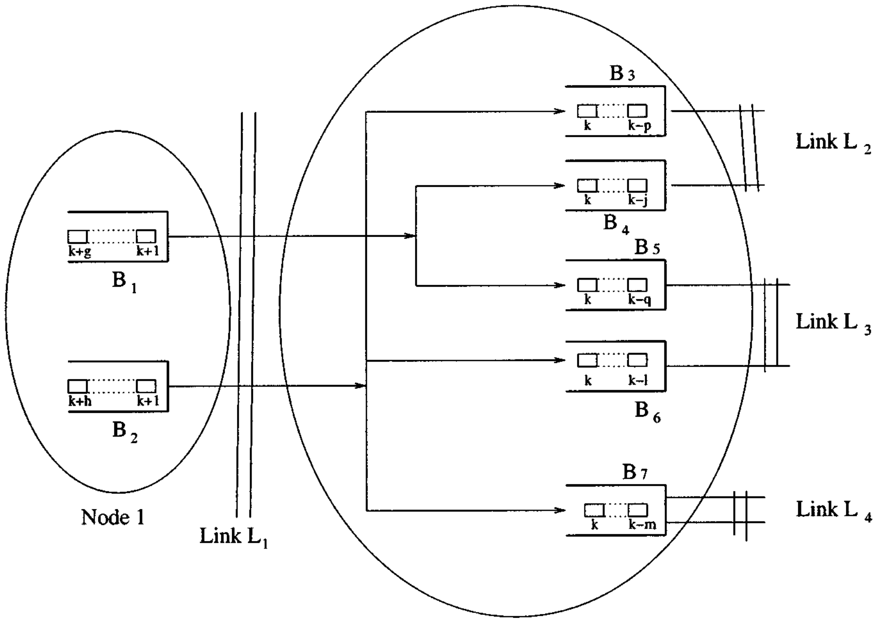
IV. Multipurpose Multicast Routing and Scheduling Policy (MMRS)

We first present an informal description of MMRS. It takes routing and scheduling decisions at intervals, the intervals sat-



Multicast Trees

(a)



Logical Buffers

(b)

Fig. 2. (a) Segments of multicast trees T_1, T_2. Tree T_1 passes through links L_1, L_2, and L_3. Tree T_2 passes through links L_1, L_2, L_3, and L_4. (b) Logical buffers at nodes 1 and 2. Here, $B_1(t)$ is actually $B_{T_1 1 L_1}(t)$ and $B_2(t)$ is actually $B_{T_2 2 L_1}(t)$. Similarly, $B_3(t)$ is $B_{T_2 2 L_2}(t)$, $B_4(t)$ is $B_{T_1 1 L_2}(t)$, $B_5(t)$ is $B_{T_1 1 L_3}(t)$, $B_6(t)$ is $B_{T_2 2 L_3}(t)$, and $B_7(t)$ is $B_{T_2 2 L_4}(t)$.

isfy some properties to be described later. The routing decision is to route an incoming session n message to the tree which has the shortest weighted queue length at the origin v_n at the decision instant (ignoring some constant bias terms for the moment). The routing policy for session n remains valid till the next routing decision instant. The scheduling decision in a link is to serve the tree which has the maximum difference between the queue lengths of its upstream buffer weighted by a scale factor and a weighted sum of the queue lengths of its downstream buffers, among all the trees with nonempty buffers contending for service in the same outgoing link (ignoring some bias terms for the moment), at the decision instant. The scheduling in a link remains the same till the next scheduling decision instant (assuming that the scheduled tree does not empty in between). The buffers we have referred to need not be *physical buffers* but are rather *logical buffers*. We explain our concept of *logical buffers* in wht follows.

We define some notations here. Every directed edge e of G has an origin vertex $o(e)$ and a destination vertex $d(e)$, e.g., $o(e_1) = 1$ and $d(e_1) = 2$ in Fig. 1. $B_{mne}(t)$ is the number of session n packets[4] traveling through the mth tree in $\mathcal{T}_n$ waiting at $o(e)$ at the end of slot t (or the beginning of slot $t+1$) for traveling to $d(e)$ through link e. The B_{mne}'s are the backlogs of the logical buffers. Fig. 2 shows some multicast trees and the corresponding logical buffers.

The logical buffers may not always represent separate memory locations, particularly for the different edges with the same origin node. The connection between the logical and the physical buffers will be discussed at the end of this section. For simplicity, we will refer to $B_{mne}(t)$'s, $m = 1, \ldots, M_n$, $e \in E$, $n = 1, \ldots, N$ as $B_1(t)$, $B_2(t)$, …, $B_M(t)$ (e.g., $B_1(t)$ denotes $B_{T_1 1 L_1}(t)$ in Fig. 2). We assume that there are

[4]For simplicity, we state MMRS for slotted arrival and service, i.e., consider packetized traffic only. It can be easily generalized to more general cases.

M logical buffers. Also, unless otherwise mentioned, buffers indicate logical buffers.

Note that the packets in the logical buffer B_i belong to the same session for every slot t. We denote this session by $n(i)$. Also, all packets in B_i are physically located at the same vertex $u(i)$ and will be transmitted through the same link $e(i)$. Every buffer has a predecessor buffer in its multicast tree, except those at the source node of a session. All packets in buffer B_i have already traversed its predecessor B_j. We denote j by $p(i)$ (predecessor of i). Also, a packet will move to a set of buffers after transmission from B_i. For example, if e_1, e_2 are in the mth tree in $\mathcal{T}_{n(i)}$, and e_1, e_2 are incident from $d(e(i))$, then a packet traversing along the mth tree will have to be transmitted across e_1 and e_2 from $d(e(i))$ and, hence, must reach buffers B_{j_1} and B_{j_2} at $d(e(i))$ after transmission from B_i, where $p(j_1) = p(j_2) = i, n(j_1) = n(j_2) = n(i)$ and $u(j_1) = u(j_2) = d(e(i))$, $e(j_l) = e_l$, $l \in \{1, 2\}$. Thus, for every buffer B_i, there exists a set of buffers, Z_i, such that any packet transmitted from buffer B_i reaches every buffer in Z_i, at the end of the transmission time. If the mth tree terminates at $d(e(i))$, $Z_i = \phi$.

Example IV.1: Refer to Fig. 2. Here, $n(1)=n(4)=n(5)=1$. Also, $n(2) = n(3) = n(6) = n(7) = 2$. This is because buffers B_1, B_4, B_5 carry session 1 packets and the other buffers carry session 2 packets. $u(1) = u(2) = 1$, $u(3) = \cdots = u(7) = 2$, $e(1) = e(2) = L_1$, $e(3) = e(4) = L_2$, $e(5) = e(6) = L_3$, $e(7) = L_4$. All packets in tree T_2 move from B_2 at node 1 to node 2 through L_1. All of these packets must be transmitted across L_2, L_3, L_4. Thus, any packet transmitted from B_2 reaches B_3, B_6, B_7. Thus, $Z_2 = \{B_3, B_6, B_7\}$. Also, $p(3) = p(6) = p(7) = 2$, since any packet in B_3, B_6, B_7 comes from B_2. Similarly, $p(4) = p(5) = 1$. $Z_1 = \{B_4, B_5\}$.

Every tree can be described by a sequence of logical buffers and every buffer B_i corresponds to a unique tree in $\mathcal{T}_{n(i)}$. We denote this tree by $m(i)$. The logical buffers corresponding to different trees are mutually disjoint.

Note that, in general, it is not necessary to choose the route immediately after packet arrival. Several routes may overlap in the first few links, and it is possible to defer the routing decision to the point where the routes diverge. Also, it is possible to envision a policy which allows route changes even after a packet is on its way. However, the routing policy we propose chooses the route of a packet immediately after the packet arrival and does not allow intermediate changes. We show that this simple policy maximizes the overall throughput.

The routing policy can be described as follows. For simplicity, let the trees be denoted by integers, i.e., $\mathcal{T}_n$ is a subset of integers. The routing vector, $\vec{\Gamma}(t)$ is defined as

$$\vec{\Gamma}(t) = (T_1(t), \ldots, T_N(t)), \qquad 1 \le T_n(t) \le M_n.$$

Here, all packets of session n arriving exogeneously at time t are routed to tree $T_n(t)$. Let O_{mn} be the set of buffers of the mth tree of $\mathcal{T}_n$ at v_n, the source node of session n, i.e.,

$$O_{mn} = \{B_i\colon n(i) = n,\ m(i) = m,\ u(i) = v_n\}.$$

Consider Fig. 2. Let T_1, T_2 be the first trees of the respective sessions. Let node 1 be the source nodes of both sessions 1 and 2. Here, $O_{11} = \{B_1\}$ and $O_{12} = \{B_2\}$. Set O_{mn} may consist of multiple buffers, e.g., if T_1 had originated from node 2 instead of node 1, then $O_{11} = \{B_4, B_5\}$. When a session n packet arrives at slot $t+1$, it is routed to the $T_n(t+1)$th tree in $\mathcal{T}_n$ and it arrives at every buffer in $O_{T_n(t+1)n}$. We update $T_n(t)$ at time instants ω_ι^n, i.e.,

$$T_n(t+1) = \begin{cases} \arg \min\limits_{1 \le m \le M_n} \left(\left(\sum\limits_{B_k \in O_{mn}} c_k B_k(t) \right) + C_{mn} \right), \\ \qquad\qquad t+1 \in \{\omega_\iota^n\}_{\iota=1}^{\infty} \\ T_n(t), \qquad \text{otherwise} \end{cases} \tag{1}$$

where c_i's are constants and C_{mn} is any constant associated with the mth tree of the nth session. Routing decisions for session n are taken at the time instants $\{\omega_\iota^n\}_{\iota=1}^{\infty}$. If $C_{mn} = 0$ for all $m \in \mathcal{T}_n$, then the routing decision taken at $\{\omega_\iota^n\}$ is to route a session n packet arriving exogeneously at a slot t in interval $[\omega_\iota^n, \omega_{\iota+1}^n)$, to the tree which has the shortest weighted queue lengths at ω_ι^n at the origin buffer v_n. C_{mn}'s are constant bias terms added to $\sum_{B_k \in O_{mn}} c_k B_k(t)$. We will discuss the significance of c_i's and C_{mn}'s later.

The routing times $\{\omega_\iota^n\}$ may be as follows.

1) At fixed intervals $\omega_{\iota+1}^n - \omega_\iota^n = T_r$. R1

2) Of bounded difference $0 \le \omega_{\iota+1}^n - \omega_\iota^n \le T_r$, $\omega_{\iota+1}^n - \omega_\iota^n$ independent and identically distributed (i.i.d.)[5] R2

3) Depend on the queue lengths. A routing decision will be taken for session n at $t+1$ if the weighted queue lengths at the origin buffer, v_n of the currently active tree ($T_n(t)$th tree) exceeds that of another tree by a certain amount ϱ_n, $\varrho_n \ge 0$. That is, $t+1 \in \{\omega_\iota^n\}_{\iota=1}^{\infty}$

if there exists $m \in M_n$ s.t.

$$C_{T_n(t)n} + \left(\sum_{B_i \in O_{T_n(t)n}} c_i B_i(t) \right) > \left(\sum_{B_i \in O_{mn}} c_i B_i(t) \right) + C_{mn} + \varrho_n. \tag{R3}$$

This means that a fresh routing decision is not taken for session n until the currently active tree is "sufficiently" congested, and the congestion is reflected in the queue lengths at the origin. In this case, routing decision always brings about a change in the currently active tree.

(Note that R1 is a subset of R2, we mention it explicitly to highlight its importance.)

Example IV.2: Refer to Fig. 2. Now let both T_1 and T_2 be session 1 trees numbered 1 and 2, respectively. (Note that in this case trees T_1 and T_2 must have the same source and destinations. The figure only shows portions of T_1 and T_2, and hence it appears that they have different destination sets.) Let node 1 be the source of session 1. Hence, $O_{11} = \{B_1\}$, $O_{21} = \{B_2\}$. Let $c_1 = c_2 = 1$, $C_{11} = C_{21} = 0$. Let Fig. 2 show the buffers just prior to ω_ι^1 (a routing decision instant for session 1). Here, $B_1(\omega_\iota^1 - 1) = g$, $B_2(\omega_\iota^1 - 1) = h$. Let $g < h$. For all t in interval $[\omega_\iota^1, \omega_{\iota+1}^1)$, $T_1(t) = 1$. Every session 1 packet which arrives in interval $[\omega_\iota^1, \omega_{\iota+1}^1)$ is routed to tree T_1. A fresh routing decision is taken at $\omega_{\iota+1}^1$. If $B_1(\omega_{\iota+1}^1 - 1) > B_2(\omega_{\iota+1}^1 - 1)$, $T_1(t) = 2$ and $T_1(t) = 1$ if $B_1(\omega_{\iota+1}^1 - 1) < B_2(\omega_{\iota+1}^1 - 1)$,

[5] $\omega_{\iota+1}^n - \omega_\iota^n$ i.i.d. $\forall\, \iota$. We do not need $\omega_{\iota+1}^{n1} - \omega_\iota^{n1}$ to be identically distributed as $\omega_{\iota+1}^{n2} - \omega_\iota^{n2}$. We also allow dependence among the residual times $\omega_{\iota+1}^{n1} - t$ and $\omega_{\iota+1}^{n2} - t$, $\forall\, n_1, n_2$, where $\omega_\iota^{n1} \le t < \omega_{\iota+1}^{n1}$ and $\omega_\iota^{n2} \le t < \omega_{\iota+1}^{n2}$.

$\omega^1_{\iota+1} \le t < \omega^1_{\iota+2}$. If $T_1(t) = 2$ at $t = \omega^1_{\iota+1}$ then all new packets of session 1 are routed to T_2 till the next routing decision is made, else all new packets are still routed to T_1. Note that $T_1(t)$ always changes value at the routing decision instants, if the routing decision intervals follow property (R3). In general, all session 1 packets arriving in $[\omega^1_\iota, \omega^1_{\iota+1})$ are routed to tree T_1 iff $c_1 g + C_{11} \le c_2 h + C_{21}$. A fresh decision is taken at $\omega^1_{\iota+1}$ on the basis of $c_1 B_1(\omega^1_{\iota+1} - 1) + C_{11}$ and $c_2 B_2(\omega^1_{\iota+1} - 1) + C_{21}$.

Next, we describe the scheduling. Now, $\vec{E}(t+1)$ is the activation vector with M components, at the $t+1$th slot

$$E_i(t+1) = \begin{cases} 1, & \text{if a packet from } B_i \text{ is served at } e(i) \\ & \text{at slot } t+1 \\ 0, & \text{otherwise.} \end{cases}$$

In other words, $E_i(t+1) = 1$ iff the ith buffer is scheduled for packet transmission through $e(i)$ at the $(t+1)$th slot. If $Z_i \ne \phi$, this packet reaches every buffer in Z_i, and also a destination if $d(e(i))$ is a destination of session n. If $Z_i = \phi$, then $d(e(i))$ is in $U_{n(i)}$ and the packet reaches its destination. We update $\vec{E}$ at time instants Ω^e_ι, $e \in E$. Let $P_e(t)$ be the set of nonempty buffers whose packets have to be transmitted across link $e \in E$ in slot t. Thus, $P_e(t+1) = \{B_i\colon e(i) = e, B_i(t) > 0\}$

$$\begin{aligned} D_i(t+1) &= c_i B_i(t) - \sum_{B_k \in Z_i} c_k B_k(t), \qquad i = 1, \dots, M \\ s_e(t+1) &= \arg \max_{i\colon B_i \in P_e(t+1)} (l_i(t+1) + D_i(t+1)), \\ &\qquad\qquad \text{if } P_e(t+1) \ne \phi \\ s_e(t+1) &= -1, \qquad \text{if } P_e(t+1) = \phi. \end{aligned}$$

Here,[6] $D_i(t)$ is the difference between the queue length of buffer i weighted by a scale factor and a weighted sum of the queue lengths of the destination buffers of buffer i at the beginning of slot t. Term l_i can be interpreted as a $\vec{B}$ dependent or a constant bias added to D_i. We indicate this by defining $l_i(t+1)$ as $l_i(t+1) = g_i(\vec{B}(t))$, where $\vec{B}(t)$ is defined as $\vec{B}(t) = (B_1(t), \dots, B_M(t))^T$ and g_i is a function from R^M to R. Typically, all $g_i(\vec{b})$'s would be constants. We will discuss the use of $g_i(\vec{b})$'s in Section V. Let $S_e = \{i\colon e(i) = e, 1 \le i \le M\}$ (the set of buffers contending for service from link e). For example, $S_{L_1} = \{1, 2\}$, $S_{L_4} = \{7\}$ in Example IV.1.

If $t+1 \in \{\Omega^e_\iota\}^\infty_{\iota=1}$, $i \in S_e$

$$E_i(t+1) = \begin{cases} 1, & i = s_{e(i)}(t+1), \\ & l_i(t+1) + D_i(t+1) > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

If $t+1 \notin \{\Omega^e_\iota\}^\infty_{\iota=1}$, $i \in S_e$

$$E_i(t+1) = \begin{cases} E_i(t), & B_i(t) > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

A packet is transmitted across link $e(i) \in E$, at slot $t+1$, if $E_i(t+1) = 1$, for some B_i, such that $e(i) = e$, else the link idles, i.e., no packet is transmitted across the link.

[6]We had used the term "congestion" at a buffer in Section I rather loosely. The term $c_i B_i(t) + l_i(t)$ can be thought of as a measure of "congestion" (as used in Section I) at logical buffer B_i when B_i is considered as the source buffer of a link and $c_i B_i(t)$ as the measure when it is considered as a destination buffer.

The scheduling decisions for link e are taken at the time instants $\{\Omega^e_\iota\}^\infty_{\iota=1}$. The scheduling decision is to choose a buffer B_i which has maximum $D_i + l_i$ at Ω_ι amongst all the buffers nonempty at Ω_ι and contending for service from outgoing link e. If $D_i(\Omega^e_\iota) + l_i(\Omega^e_\iota) > 0$, $i \in S_e$, then the scheduling decision is to serve a packet from B_i at each slot t in the interval $[\Omega^e_\iota, \Omega^e_{\iota+1})$ unless B_i becomes empty at some t in the interval $(\Omega^e_\iota, \Omega^e_{\iota+1})$. If B_i becomes empty at some t in the interval $(\Omega^e_\iota, \Omega^e_{\iota+1})$, then the link idles till the next scheduling slot $\Omega^e_{\iota+1}$. If $D_i(\Omega^e_\iota) + l_i(\Omega^e_\iota) \le 0$, then the link idles during the entire scheduling decision interval $[\Omega^e_\iota, \Omega^e_{\iota+1})$.

Like the routing times $\{\omega^n_\iota\}$, the scheduling times $\{\Omega^e_\iota\}$ may be as follows.

1) At fixed intervals $\Omega^e_{\iota+1} - \Omega^e_\iota = T_s$. S1
2) Of bounded difference $\Omega^e_{\iota+1} - \Omega^e_\iota \le T_s$, $\Omega^e_{\iota+1} - \Omega^e_\iota$ i.i.d. S2
3) Depend on the $D_i + l_i$s, $i \in S_e$. S3

$$\begin{aligned} j_e(t) &= \arg \max_{i \in S_e} E_i(t) \\ p_e(t) &= \arg \max_{i \in P_e(t)} (D_i(t) + l_i(t)) \end{aligned}$$

A scheduling decision is taken for link e at the beginning of slot $t+1$ if any of the following conditions is satisfied.

a) Currently scheduled buffer has $D_i + l_i$ sufficiently less than that of some other contending buffer. $E_{j_e(t)}(t) = 1$,

$$\begin{aligned} &D_{j_e(t)}(t+1) + l_{j_e(t)}(t+1) \\ &\qquad \le D_{p_e(t+1)}(t+1) + l_{p_e(t+1)}(t+1) - \varsigma_{e1}. \end{aligned} \tag{S3a}$$

b) The $D_i + l_i$ of the currently scheduled buffer becomes sufficiently negative. $E_{j_e(t)}(t) = 1$

$$D_{j_e(t)}(t+1) + l_{j_e(t)}(t+1) \le -\varsigma_{e2}. \tag{S3b}$$

c) The link is currently idle but the $D_i + l_i$ of some nonempty buffer which can possibly be served by the link becomes sufficiently positive

$$E_{j_e(t)}(t) = 0 \quad D_{p_e(t+1)}(t+1) + l_{p_e(t+1)}(t+1) \ge \varsigma_{e3}. \tag{S3c}$$

$\varsigma_{e1}, \varsigma_{e2}, \varsigma_{e3}$ are prespecified positive real numbers. Again, informally this means that a scheduling decision is not taken for a link till the last scheduling decision becomes "too bad" for the current state of the network.

(Note that S1 is a subset of S2.)

We explain the scheduling policy with an example.

Example IV.3: Let Fig. 2 show the buffers just prior to $\Omega^{L_1}_\iota$ (a scheduling decision instant for link L_1).

$$\begin{aligned} B_1(\Omega^{L_1}_\iota - 1) &= g \\ B_2(\Omega^{L_1}_\iota - 1) &= h \\ B_3(\Omega^{L_1}_\iota - 1) &= j + 1 \\ B_4(\Omega^{L_1}_\iota - 1) &= p + 1 \\ B_5(\Omega^{L_1}_\iota - 1) &= q + 1 \\ B_6(\Omega^{L_1}_\iota - 1) &= l + 1 \\ B_7(\Omega^{L_1}_\iota - 1) &= m + 1. \end{aligned}$$

Let $c_i = 1$ and $g_i(\vec{b}) = 0 \,\forall i$, i.e., $l_i(t) = 0$, for all i, t.

$$D_1(\Omega^{L_1}_\iota) = g - p - q - 2$$

$$D_2(\Omega_\iota^{L_1}) = h - j - l - m - 3.$$

Let $p = q = j = l = m = 1$, $g = 5$, $h = 8$. Thus,

$$D_1(\Omega_\iota^{L_1}) + l_1(\Omega_\iota^{L_1}) = 1$$

and

$$D_2(\Omega_\iota^{L_1}) + l_2(\Omega_\iota^{L_1}) = 2.$$

Thus, the scheduling decision is to serve a packet from B_2 in every slot till the next scheduling decision instant assuming that B_2 does not empty in between. If B_2 empties in between, then L_1 idles till the next scheduling decision instant. Now let $g = h = 1$. Both $D_1(\Omega_\iota^{L_1}) + l_1(\Omega_\iota^{L_1})$ and $D_2(\Omega_\iota^{L_1}) + l_2(\Omega_\iota^{L_1})$ are negative and hence L_1 idles till the next scheduling instant, independent of the $D_i + l_i$'s in between. Now let $g_1(\vec{b}) = 4$, $g_2(\vec{b}) = 0$. In both cases, the scheduling decision is to serve a packet from B_1 in every slot till the next scheduling decision instant assuming that B_1 does not empty in between. If B_1 empties in between, then again L_1 idles till the next scheduling decision instant. This change in the bias term $g_1(\vec{b})$ gives limited priority to session 1 in link L_1. Again if $p = q = j = l = m = 1$, $g = 5$, $h = 8$, $g_i(\vec{b}) = 0 \; \forall i$, but $c_1 = c_4 = c_5 = 3$ and $c_2 = c_3 = c_6 = c_7 = 1$, then

$$D_1(\Omega_\iota^{L_1}) + l_1(\Omega_\iota^{L_1}) = 3$$
$$D_2(\Omega_\iota^{L_1}) + l_2(\Omega_\iota^{L_1}) = 2.$$

Thus, the scheduling decision is taken in favor of B_1.

Informally speaking, MMRS attains the maximum throughput for any arbitrary network (the precise technical statement is given in Section VI). We prove this in Section VII.

Whenever a packet arrives at a node (not necessarily exogeneously) the node must know the logical buffers B_i's (or B_{mne}'s) to which the packet belongs. This is necessary to keep track of the $B_i(t)$'s. Thus, the packet header must contain information specifying its session and the tree it has been routed to. Every node may know the outgoing links of every tree traversing it. This happens if there is a connection setup phase associated with every session initiation, like in a virtual-circuit scenario. In this case, the packet header just contains a number identifying the tree and the node determines the next hop edges of the packet using its lookup table and the identifier in the packet header. The logical buffers can be uniquely identified once the next hop links are known. In a datagram-like scenario, there is no connection establishment process. Thus, the nodes do not necessarily know the outgoing links of the trees passing through them. The packet header must contain explicit information about the edge sequences of the tree in addition to the tree and the session number. Immediately after the packet arrives exogeneously it is routed to a tree as per the last routing decision and the necessary information is incorporated in the packet header. The necessary information includes the tree and the session numbers but may or may not contain the explicit tree path depending on whether the nodes know the tree paths or not.

We end this section with a brief discussion on the relation between the physical and the logical buffers. A node is a multi-input multi-output multicast switch with the ability to serve packets to several outgoing links simultaneously. Fig. 2 shows a node, Node 2, with one incoming link and three outgoing links.

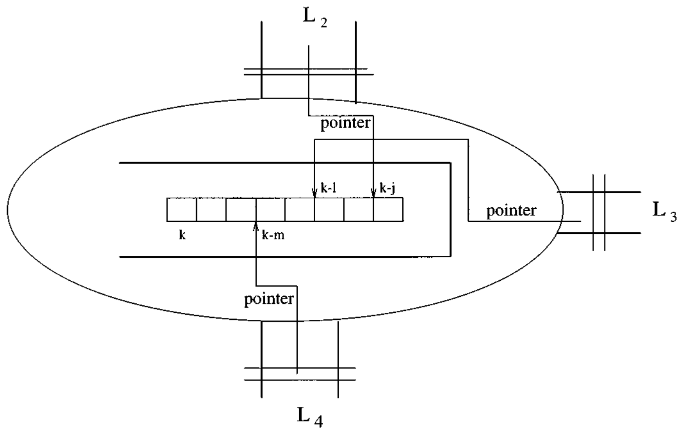


Fig. 3. A physical buffer storing T_2 packets at Node 2 of Fig. 2.

Packets reach Node 2 via trees T_1 and T_2. The node can simultaneously transmit a T_2 packet into L_4 and a T_1 packet into L_3. Both packets reach Node 2 via link L_1. The packets can be queued at the input or at the output of the node or queued in a shared memory mode. The physical buffers are the memory locations which store these packets. The relation between the physical buffers and the logical buffers depend upon whether the packets are input queued or output queued or stored in a shared memory mode.

• If the packets are input queued, then the packets are replicated[7] (if at all) only when they are transmitted to the output. There is a single physical buffer at each node for storing all packets (one copy each) of a tree traveling through the node. Upon arrival, only a single copy of the packet is stored in the node. Replication coincides with transmission to the outputs. This mode of replication is known as *replication-at-sending* (RAS) [4]. A packet remains in the buffer till it has been transmitted across all the necessary links originating from the same node. Each link uses a pointer to keeps track of the number of packets of each tree waiting for transmission. The pointer points to the first packet it needs to transmit and moves the pointer to the next packet when the first is transmitted. These numbers are precisely the logical buffer queue lengths $B_{mne}(t)$'s.

Example IV.4: Fig. 3 shows the physical buffer at Node 2 for tree T_2 of Fig. 2. It currently has packets numbered $k - j, \ldots, k$. All of them must be transmitted across link L_2. Link L_2 maintains a pointer at the first packet waiting for transmission, packet $k - j$. Link L_3 has transmitted packets $k - j, \ldots, k - l - 1$. It has to transmit packets $k - l$ to k. So it maintains a pointer at the $k - l$th packet. Similarly, L_4 has already transmitted packets $k - j, \ldots, k - m - 1$ but needs to transmit packets $k - m$ to k. It has a pointer at the $k - m$th packet. Fig. 2 shows the contents of the separate logical buffers for this tree at Node 2 (B_3, B_6, and B_7). Fig. 2 indicates that the $k + 1$th and subsequent packets of T_2 are still waiting at Node 1 for transmission across Link L_1 to Node 2.

For input queued switches, the physical and the logical buffer queue lengths can be mathematically related as follows. Let E_{Ti} be the set of outgoing links of tree T at node i, e.g.,

[7]Replication occurs only when the corresponding tree forks at the node.

$E_{T_1 2} = \{L_2, L_3\}$, $E_{T_1 1} = \{L_1\}$, $E_{T_2 2} = \{L_2, L_3, L_4\}$ in Fig. 2. Let physical buffer u store packets of tree T of session n at node i and $X_u(t)$ be the number of packets in the physical buffer u by the end of slot t (or the beginning of slot $t+1$). Then, $X_u(t) = \max_{e \in E_{Ti}} B_{Tne}(t)$.

Example IV.5: Refer to Figs. 2 and 3. Fir. 3 shows the physical buffer storing tree T_2, session 2 packets at Node 2. The physical buffer has $j+1$ packets. B_3 has $j+1$ packets, B_6 has $l+1$ packets, and B_7 has $m+1$ packets (refer to Example IV.4), where $j > l > m$. From Figs. 2 and 3

$$E_{T_2 2} = \{L_2, L_3, L_4\}$$
$$X_u(t) = \max(B_3(t), B_6(t), B_7(t)) = j+1.$$

Packets traveling along different trees must occupy different memory locations. We assume that buffers are large and there is no packet loss. In this case, there is no essential difference between storing the packets of different trees in the same or different physical buffers. Thus, we assume separate physical buffers for different trees at each node without any loss of generality.

• If the packets are output queued, then every outgoing link has a separate physical buffer. Recall that $B_{mne}(t)$ is the number of session n, tree m packets output queued at the outgoing link e of node $o(e)$ at the end of slot t. If physical buffer u stores the packets output queued at e, then $X_u(t) = \sum B_{mne}(t)$, where the summation is over all trees m of all sessions n which pass through link e. Note that here packets are replicated (again replication occurs if the corresponding tree forks at the node) immediately upon arrival and subsequently transmitted to the output queue. This mode of replication is known as *replication-at-receiving* (RAR) [4].

• The node may be a shared memory switch with the memory fully shared between all queues. There is only a single physical buffer for each tree at each node. Replication can be RAR or RAS. In the former, a multicast packet is physically replicated in front of the shared buffer, the multiple copies of the packet are stored in the buffer, each copy of the packet is queued till it is served by its requisite link. The RAR scheme has been used in several shared-memory multicast ATM switches [13]. Let the physical buffer w store the packets at node w

$$X_w(t) = \sum_{i:\, u(i)=w} B_i(t).$$

In the latter (RAS), a single instance of the multicast packet is stored in the buffer and is physically replicated only as it is transmitted to the respective output link. The RAS scheme has been recently adopted in shared-memory switches [25]. Now

$$X_w(t) = \sum_{T \in \mathcal{V}} \max_{e \in E_{Tw}} B_{Tn_T e}(t)$$

where $\mathcal{V} = \bigcup_{n=1}^{N} \mathcal{T}_n$, E_{Tw} is the set of outgoing links of tree T at node w, and n_T is the session corresponding to tree T.

V. Discussion

We present some attractive features of MMRS in this section. First, MMRS attains globally optimum throughput even though it takes routing and scheduling decisions based on local information only. The routing decisions depend on the congestion at the source node and do not consider the congestion in the entire tree. The intuition is that the queue lengths at the source node reflect the congestion in the entire tree on account of the back-pressure-based scheduling. The scheduling decision for a link depends only on the queue lengths at the source and the destination of the link (assuming that the bias functions $g_i(\vec{b})$ depend only on the local queue length b_i). Normally, the scheduler is located at the source node of the link. Therefore, congestion at the destination of the link must be communicated to the source of the link. However, it is sufficient to communicate the scheduling decisions of the link destination to the link source, and then the source of the link can recursively compute the queue lengths at the destination. Scheduling decisions can be communicated using only a limited number of bits.

MMRS is computationally efficient (assuming that the bias functions $g_i(\vec{b})$ are easy to compute).

MMRS is adaptive as it does not assume any information about the arrival and service statistics. We have proved the throughput optimality for a class of arrival and service processes (Markov modulated arrival, and service duration one slot for each packet). However, these assumptions are not required for the implementation of the policy, and we believe that the throughput optimality holds for more general arrival and service processes.

The queue lengths at the source node reflect the congestion status in the entire tree on account of the back-pressure-based scheduling. Thus, end-to-end congestion control schemes may be applied based on the queue lengths at the source node, e.g., if the queue lengths are high then the source may be asked to slow down. For instance, if the source is a video source, then the quantization may be made coarse when these queue lengths exceed a certain threshold, and the encoding scheme can revert to a fine-grained quantization when the queue lengths fall below a certain threshold. Thus, the queue lengths at the source node provide *implicit feedback* as to the congestion state of the network. The congestion propagates to the source node after some time, but any explicit feedback will also take time to reach the source node, particularly if the network is large. Besides, usage of explicit feedback often gives rise to the problem of *feedback implosion* in the multicast scenario [34]. This solution of implicit feedback is inherently scalable. This implicit feedback may not be able to substitute the need for explicit feedback entirely, but can be used in conjunction, so that much more infrequent explicit feedback will suffice.

It is not clear whether we need global information and/or information regarding statistics of the arrival and service for the routing and scheduling decisions, if the objective is to optimize other performance considerations like delay, packet loss, etc. MMRS optimizes throughput, but we do not know whether it optimizes other performance metrics or not. MMRS is actually a class of policies. The exact policy depends on the choice of the parameters. All of these individual policies attain the optimum throughput, but the choice of the parameters is likely to have an impact on other performance considerations. The complete characterization of the parameter values for delay and loss guarantees for general networks is an interesting research issue by itself, and a topic for future research. However, we provide some general guidelines from our initial research in this direction.

One can use the c_i's to give limited priority to some sessions over others without affecting the throughput. Increasing the c_i's for a session over others decreases the delay of the packets of the session at the expense of a greater delay experienced by the packets of other sessions. Thus, one would expect the c_i's to be higher for real-time sessions like audio, video, and possibly for applications which fetch greater revenue. Also, l_i's can serve the same purpose, i.e., give limited priority to sessions over each other.

One can use the C_{mn}'s to give limited priority to some trees of a session n over some other trees in the routing decision. A multicast tree m of session n may not be a desirable route for a session for various reasons, e.g., it may be very long thereby incurring a large propagation delay. Typically, the network may want to use it only when other trees of the session are significantly congested. This purpose can be achieved by setting a high value of C_{mn} for the tree, so that $\sum_{B_i \in O_{mn}} c_i B_i(t) + C_{mn}$ is the minimum among all trees of session n, only when other trees have high congestion. Manipulation of the c_i's can serve the same purpose, but this affects the scheduling as well. The choice of the C_{mn}'s do not affect the scheduling decision.

The parameters c_i and $g_i(\vec{b})$ can be suitably modified to reduce packet loss, if necessary. If memory size is small at a node as compared to its neighbors then the c_i's and l_i's corresponding to the logical buffers at the node (e.g., logical buffers B_3, B_6, B_7 correspond to Node 2 in Fig. 2) can be set higher than the corresponding ones at the neighboring nodes. Thus, the scaled queue lengths ($c_i B_i(t) + l_i$'s) of the corresponding logical buffers would be high even if the actual queue lengths $B_i(t)$'s are not so high. The links bringing packets to the node would idle frequently and the links serving packets from the node would idle rarely. The queue length at the node would be small at the expense of larger queue lengths at its neighbors. This would reduce the overall packet loss.

We would like to mention in this context that the parameters affect the relative delay and loss performances among different sessions. However, if absolute guarantees are required, then admission control and quality of service negotiation must be used in conjunction with MMRS. For example, if the arrival rates are high for all sessions, then all sessions suffer large delays, and the system cannot offer tight delay guarantees. However, if admission control is used to regulate the number of sessions and their individual arrival rates, then this situation would not arise.

The routing and scheduling decisions can be taken every slot or at intervals. We examine the pros and cons of both approaches. Initially, assume that the decisions do not depend on queue lengths. There is a communication and computation overhead associated with each decision, and taking decisions at intervals means that the overhead is once per interval and not once every slot. Also, taking the routing decision at intervals reduce the out-of-order delivery at the endpoints. The advantage of taking decisions every slot is that the decision optimality is maintained every slot, whereas decisions deviate from the optimal ones if the decision intervals are large, and as a result there is more end-to-end delay for large decision intervals. The advantage of using queue-length-dependent decision intervals is that some unnecessary decision changes are avoided, and some necessary decision changes are made. i.e., a fresh routing/scheduling decision is taken only when the current decision becomes "unacceptably bad." This is a significant advantage when there is a cost associated with changing the current routing or scheduling decision. The tradeoff between the frequency of decision changes and overall system performance can be controlled through the parameters ϱ_n, $n = 1, \ldots, N$, and ς_i, $i = 1 \cdots M$. The disadvantage of this approach is that a computation must be performed every slot, whereas the computation is performed once every interval for queue-length-independent intervals.

We have specified a general approach so far. However, application-specific modifications may be necessary in particular cases. For example, all packets of the same session must follow the same route for ATM networks. In this case, the routing decision for a session should be taken only when the session arrives. The decision would be to choose from a list of potential trees for the same source and destinations. Normally, there are a large number of sessions between the same source and destinations. So the routing decision would depend on the source congestion for the existing sessions between the same source and destinations. If there is no other session between the same source and destination, then the routing decision would be to choose any tree among the available ones. Also, for broadcast sessions, the tree must be a spanning tree, and there are efficient algorithms for computing the minimum-weight spanning tree, unlike that for minimum-weight multicast trees which is an NP-complete problem (Steiner problem). In this case, a better strategy would be to compute the minimum-weight spanning tree periodically and route packets along the current minimum-weight spanning tree. The scheduling policy remains the same in all these cases.

VI. Formal Throughput Properties of MMRS

Intuitively, the concept of stability of a queueing system is associated with the queue length process at the *physical buffers* (buffers corresponding to actual storage locations). Again, $X_u(t)$ is the number of packets in a physical buffer u by the end of slot t (or the beginning of slot $t+1$). We define the system to be stable if there exists a family of random vectors, $\hat{X}^{(i,j)}$ $i = 0, \ldots, P-1$, $j = 0, \ldots, Q-1$, $E\hat{X}^{(i,j)} < \infty$, $\forall\, i, j, P, Q$ finite, such that $\{\vec{X}(t)\}_{t=0}^{\infty}$ can be partitioned into Q subsequences $\{\vec{X}(tQ+\theta)\}_{t=0}^{\infty}$, $\theta = 0, \ldots, Q-1$, and $\vec{X}(tQ+\theta)$ converges weakly to a random vector $\hat{X}^{(i,(j+\theta)\%Q)}$, where (i, j) are uniquely specified given the *initial phase* of the system (% is the modulo operator). The *phase* of the system should contain some information not contained in the queue lengths at the physical or the logical buffers. The exact definition of the phase depends on the particular routing and scheduling policy. For example, the phase of the system could be the residual times for the next routing and scheduling decisions for MMRS with the routing and scheduling decision intervals satisfying R2 and S2, respectively. In that case, the initial phase (phase at $t = 0$) indicates the residual times for the first routing and scheduling decisions, and given this information, (i, j) should be known uniquely. If the routing and scheduling decision intervals follow R1, R3, S1, and S3, then the residual times for the next routing and scheduling decisions can be determined from the logical buffer queue lengths and hence the system does not have any phase ($P = Q = 1$), i.e., $\{\vec{X}(t)\}_{t=0}^{\infty}$ converges weakly to a random vector $\hat{X}$, with $E\hat{X} < \infty$. The "initial phase" determines the random vector the queue lengths converge to (convergence in distribution). The proof for Lemma 1 indicates the significance of the initial phase. The intuition behind this definition is that we generally consider a system to be stable, as long as the physical buffers do not "blow up" and this does not happen if $\vec{X}(t)$ converges weakly to one of finitely many finite mean random vectors and hence those systems should be considered stable. The particular random vector $\vec{X}(t)$ converges to (in distribution), should be uniquely determined from some initial phase of the system.

The stability condition can be expressed in terms of logical buffer queue lengths as follows. Refer to the discussion at the end of Section IV describing the relation between the physical buffer and logical buffer queue lengths. It follows that, if there exists a family of random vectors, $\hat{B}^{(i,j)}$ $i = 0, \ldots, P-1$, $j = 0, \ldots, Q-1$, $E\hat{B}^{(i,j)} < \infty$, $\forall i, j, P, Q$ finite, and $\{\vec{B}(t)\}_{t=0}^{\infty}$ can be partitioned into Q subsequences

$$\{\vec{B}(tQ+\theta)\}_{t=0}^{\infty}, \qquad \theta = 0, \ldots, Q-1$$

such that $\vec{B}(tQ+\theta)$ converges weakly to a random vector

$$\hat{B}^{(i,(j+\theta)\%Q)}, \qquad i \in \{0, \ldots, P-1\},\ j \in \{0, \ldots, Q-1\}$$

i, j are uniquely determined given the initial phase of the system, then the system is stable.

Let a_n be the expected number of session n packets arriving in a slot. We call $(a_1, a_2, \ldots, a_N)$ the arrival rate vector. Informally speaking, throughput is the traffic carried by the system. A system is said to "carry" a traffic if it is stable under the traffic. We denote the arrival rate vector as the throughput of the system if it is stable. If the system is not stable, then throughput becomes meaningless, and can be arbitrarily defined as the 0 vector. We call an arrival rate vector feasible if for each session n the total traffic a_n can be split into portions a_n^m, $a_n^m \geq 0$, $m = 1, \ldots, M_n$, and $\sum_{m=1}^{M_n} a_n^m = a_n$, where M_n is the total number of trees in $\mathcal{T}_n$ and $a_n^m, m = 1, \ldots, M_n, n = 1, \ldots, N$ satisfies the capacity condition with strict inequality. Intuitively, a_n^m is the amount of traffic routed through tree m in $\mathcal{T}_n$.

We assume that each packet has a deterministic service time equal to one slot at each link. In that case, an arrival rate vector is feasible if

$$\sum_{n=1}^{N}\sum_{m=1}^{M_n} a_n^m T_n^m < (1, \ldots, 1). \tag{4}$$

T_n^m is the indicator vector of the mth tree in T_n. MMRS renders the system stable for every feasible arrival rate vector. We prove this in the next section.

We shall prove in Section VIII that the system is not stable if the arrival rate for session n (a_n) cannot be split in portions a_n^m such that the a_n^m's satisfy the capacity condition stated in Section II. This gives the necessary condition for stability. The condition for feasibility of an arrival rate vector is the same as the necessary condition for stability for all practical purposes. Technically speaking, if an arrival rate vector $(a_1, \ldots, a_N)$ satisfies the necessary condition for stability, then MMRS renders the system stable for any arrival vector $(a_1 - \epsilon, \ldots, a_N - \epsilon)$, for any arbitrarily small ϵ. Quite possibly, MMRS renders the system stable for arrival vector $(a_1, \ldots, a_N)$ itself (that is, if it satisfies the capacity condition with strict inequality). Thus, MMRS is throughput optimal for all practical purposes.

VII. Proof of Throughput Optimality of MMRS

We make the following assumptions for the purpose of analysis. Arrival and service are slotted. Each session has its own exogeneous i.i.d. arrival stream of packets $\{A_n(t)\}_{t=1}^{\infty}$, where $A_n(t)$ is the number of session-n packets arriving in slot t, $A_n(t) \leq K_n$, $\forall n, t$, K_n is a positive integer for all n. As mentioned in Section VI, each packet has a deterministic service time equal to one unit. Note that the dependence between $A_{n_1}(t)$ and $A_{n_2}(t)$ has not been ruled out for any $n_1 \neq n_2$. We have discussed the generalizations of these assumptions toward the end of this section.

Let the arrival rate vector be feasible and let MMRS be followed. We also assume certain properties of c_i's and $g_i(\vec{b})$'s. $c_i > 0\ \forall i$. g_i's can be any arbitrary function from R^M to R satisfying the following property:

$$\lim_{\|c\vec{b}\| \to \infty} \frac{g_i\left(\vec{b}\right)}{\left\|c\vec{b}\right\|} = 0, \quad \forall i, \quad \text{where } \left\|c\vec{b}\right\| = \sqrt{\sum_{i=1}^{M} c_i b_i^2}. \tag{5}$$

Note that a large class of g_i's satisfies the above property, e.g., any bounded function g, any linear function of $\sqrt{b_1}, \ldots, \sqrt{b_M}$, etc. Let $\tilde{A}_i(t)$ be the number of exogeneous packet arrivals at B_i at slot t

$$\tilde{A}_i(t) = \begin{cases} A_n(t), & \text{if } B_i \in O_{T_n(t)n(i)} \\ 0, & \text{otherwise.} \end{cases}$$

The routing matrix R can be represented as follows:

$$R_{ab} = \begin{cases} -1, & a = b \\ 1, & a \in Z_b \\ 0, & \text{otherwise.} \end{cases}$$

$$\vec{B}(t+1) = \vec{B}(t) + R\vec{E}(t+1) + \tilde{A}(t+1). \tag{6}$$

Initially assume that the routing policy satisfies either R1 or R2 and the scheduling policy satisfies either S1 or S2. We will discuss the case for other routing and scheduling policies toward the end of this section. The residual time for the next scheduling decision at link e, $\Xi_e(t)$ can be expressed as $\Xi_e(t) = \Omega_{\iota+1}^e - t$, where $\Omega_\iota^e \leq t < \Omega_{\iota+1}^e$. The residual time for the next routing decision for the nth session $\xi_n(t)$ can be described as $\xi_n(t) = \omega_{\iota+1}^n - t$, where $\omega_\iota^n \leq t < \omega_{\iota+1}^n$. Both $\xi_n(t)$ and $\Xi_e(t)$ take values in a finite set.

Let $\vec{Y}(t) = (\vec{B}(t), \vec{E}(t), \vec{\Gamma}(t), \vec{\Xi}(t), \vec{\xi}(t))$. The phase of the system is $(\vec{\Xi}(t), \vec{\xi}(t))$. It contains some information not in $\vec{B}(t)$ for any t. The main result of this section is contained in Theorem 1 stated as follows.

Theorem 1: There exists random vectors $\hat{B}^{(k,l)}$, $k = 0, 1, \ldots, P-1$, $l = 0, 1, \ldots, Q-1$, $E\hat{B}^{(k,l)} < \infty$ such that given $(\vec{\Xi}(0), \vec{\xi}(0))$

$$\{\vec{B}(tQ+i)\}_{t=0}^{\infty}, \qquad i = 0, 1, \ldots, Q-1$$

converges weakly to $\hat{B}^{(k,(l+i)\%Q)}$, k, l uniquely known given $(\vec{\Xi}(0), \vec{\xi}(0))$, $k = 0, 1, \ldots, P-1$, $l = 0, 1, \ldots, Q-1$.

We prove the above theorem toward the end of this section using Proposition 1 and Lemmas 1 and 2 stated in what follows.

Proposition 1: Let $\vec{Y}(t)$ be an aperiodic discrete-time countable state Markov chain with state space $\mathcal{X}$, and a single closed communication class accessible from all states. If there exists real nonnegative functions $\phi_1(\vec{y})$, $\phi_2(\vec{y})$ such that $\phi_1(\vec{y}) \geq 1$, $\phi_2(\vec{y})$ finite for all $\vec{y} \in \mathcal{X}$ and

$$E\left(\phi_2\left(\vec{Y}(t+1)\right)\Big/\vec{Y}(t) = \vec{y}\right) < \phi_2(\vec{y}) - \phi_1(\vec{y}), \quad \forall \vec{y} \in A^c$$

where A is a finite subset of $\mathcal{X}$, then $\{\vec{Y}(t)\}_{t=0}^{\infty}$ converges weakly to a random vector $\hat{Y}$, such that $E\phi_1(\hat{Y}) < \infty$.

This proposition has been stated in a manner appropriate to our context. It follows from a theorem in [18] which we describe in Appendix II. Basically, it states that if a "negative drift condition" (stated in Lemma 2) holds, then the system is stable.

Lemma 1: Given $(\vec{\Xi}(0), \vec{\xi}(0))$, $\{\vec{Y}(tQ+i)\}_{t=0}^{\infty}$, $i = 0, \ldots, Q-1$, is a discrete-time countable state aperiodic Markov chain with state space $\mathcal{X}_{(I(0),(J(0)+i)\%Q)}$. Here, $(I(0), J(0))$ is uniquely known given $(\vec{\Xi}(0), \vec{\xi}(0))$. Also, $\mathcal{X}_{(I(0),(J(0)+i)\%Q)}$ has a single closed communication class for all $(I(0), J(0))$, i. This class is accessible from all states in $\mathcal{X}_{(I(0),(J(0)+i)\%Q)}$.

We prove this lemma later in this section.

Lemma 2 (Negative Drift Condition): There exists real nonnegative functions $\phi_1(\vec{y})$, $\phi_2(\vec{y})$ such that $\phi_1(\vec{y}) \geq 1$, $\phi_2(\vec{y})$ finite for all $\vec{y} \in \mathcal{X}$, and

$$E(\phi_2(\vec{Y}((t+1)Q+i))/\vec{Y}(tQ+i) = \vec{y}) < \phi_2(\vec{y}) - \phi_1(\vec{y}), \quad \forall \vec{y} \in A^c$$

where A is a finite subset of $\mathcal{X}$.

Lemma 2 is crucial to the proof of our main result of this section, that is, the throughput optimality of MMRS. We call this lemma the negative drift condition, because it proves that for a large number of $\vec{Y}$'s, the expected conditional drift of a function $\phi_2(\vec{Y})$ is bounded above by a function $-\phi_1(\vec{Y})$ with negative values. We prove this negative drift condition in Appendix I.

Proof of Lemma 1: $\vec{Y}(t)$, $(\vec{\Xi}(t), \vec{\xi}(t))$ are discrete-time countable state Markov chains with state space $\mathcal{X}$ and $\mathcal{C}$, respectively. We assume that $\mathcal{C}$ can be partitioned in $\mathcal{C}_0, \ldots, \mathcal{C}_{P-1}$, such that the $\mathcal{C}_i$'s are closed communication classes of periodicity Q. This is a fairly general assumption on the structure of $\mathcal{C}$, which incorporates the cases when $\{\Xi_e(t)\}_{e=1}^{|E|}$, $\{\xi_n(t)\}_{n=1}^{N}$'s are mutually independent and also holds in case of most common dependencies among the $\Xi_e(t)$'s and $\xi_n(t)$'s, e.g., a subset of $\Xi_e(t)$'s, $\xi_n(t)$'s are always equal, etc. Thus, $\mathcal{C}_i$ can be partitioned in periodicity classes $\mathcal{C}_{(i,0)}, \ldots, \mathcal{C}_{(i,Q-1)}$, such that if $(\vec{\Xi}(t), \vec{\xi}(t)) \in \mathcal{C}_{(i,j)}$, $(\vec{\Xi}(t+1), \vec{\xi}(t+1)) \in \mathcal{C}_{(i,(j+1)\%Q)}$, with probability (w.p.) 1. It follows that $\mathcal{X}$ can be partitioned into $\mathcal{X}_0, \ldots, \mathcal{X}_{P-1}$, $\vec{Y}(t) \in \mathcal{X}_i$ iff $(\vec{\Xi}(t), \vec{\xi}(t)) \in \mathcal{C}_i$. Since $\mathcal{C}_i$'s are closed communication classes and the state $\vec{B} = \vec{0}$ is reachable from any $\vec{B} = \vec{B}_0$, it follows that $\mathcal{X}_i$ consists of only one closed communication class accessible[8] from every state and has periodicity Q for all i. The periodicity classes of $\mathcal{X}_i$ are $\mathcal{X}_{(i,0)}, \ldots, \mathcal{X}_{(i,Q-1)}$, where $\vec{Y}(t) \in \mathcal{X}_{(i,j)}$ iff $(\vec{\Xi}(t), \vec{\xi}(t)) \in \mathcal{C}_{(i,j)}$. Let

$$\left.\begin{array}{l} I(t) = l \\ J(t) = m \end{array}\right\} \quad \text{iff } \left(\vec{\Xi}(t), \vec{\xi}(t)\right) \in \mathcal{C}_{(l,m)}.$$

Hence the lemma is proved. □

Proof of Theorem 1: It follows from Lemmas 1 and 2 and Proposition 1 that there exists random vectors $\{\hat{Y}^{(k,l)}\}$, $E\hat{Y}^{(k,l)} < \infty$, $k \in \{0, \ldots, P-1\}$, $l \in \{0, \ldots, Q-1\}$ such that given $(\vec{\Xi}(0), \vec{\xi}(0))$, $\{\vec{Y}(tQ+i)\}_{t=0}^{\infty}$ converges in distribution to a random vector $\hat{Y}^{(I(0),(J(0)+i)\%Q)}$, $E\phi_1(\hat{Y}^{(I(0),(J(0)+i)\%Q)}) < \infty$. Since

$$\vec{Y}(t) = (\vec{B}(t), \vec{E}(t), \Gamma(t), \vec{\Xi}(t), \vec{\xi}(t))$$

given $(\vec{\Xi}(0), \vec{\xi}(0))$, $\{\vec{B}(tQ+i)\}_{t=0}^{\infty}$ converges in distribution to a random vector $\hat{B}^{(I(0),(J(0)+i)\%Q)}$, $I(0) \in \{0, \ldots, P-1\}$, $J(0) \in \{0, \ldots, Q-1\}$. We used function

$$\phi_1(\vec{y}) = \max\left(1, \frac{2\lambda'\sqrt{\sum_{i=1}^{M} c_i b_i^2}}{\kappa}\right)$$

$\kappa > 1$, $\lambda' > 0$ are constants, c_i's are strictly positive constants. Thus, $E\phi_1(\hat{Y}^{(I(0),(J(0)+i)\%Q)}) < \infty$ implies that $E\hat{B}^{(I(0),(J(0)+i)\%Q)} < \infty$. Hence the result follows. □

If the routing policy satisfies either R1 or R2 and the scheduling policy satisfies S3, then the Markov chain

$$\vec{Y}(t) = (\vec{B}(t), \vec{E}(t), \Gamma(t), \vec{\xi}(t))$$

represents the system. If the routing policy satisfies R3 and the scheduling policy satisfies either S1 or S2, then the Markov chain

$$\vec{Y}(t) = (\vec{B}(t), \vec{E}(t), \Gamma(t), \vec{\Xi}(t))$$

represents the system. Finally, if the routing policy satisfies R3 and the scheduling policy satisfies S3, the Markov chain

$$\vec{Y}(t) = (\vec{B}(t), \vec{E}(t), \Gamma(t))$$

represents the system. In the first two cases, Lemma 1 holds with the periodicity determined by that of $\vec{\xi}(t)$ and $\vec{\Xi}(t)$, respectively. The phase of the system is determined by $\vec{\xi}(t)$ and $\vec{\Xi}(t)$ accordingly. In the last case, the Markov chain $\vec{Y}(t)$ is aperiodic and has a single closed communication class accessible from all states, i.e., $Q = P = 1$. Thus, Lemma 1 holds in this case as well. As will be discussed in the Appendix I, Lemma 2 holds in all these cases. Thus, Theorem 1 holds in all these cases as well. Hence, the proof of stability for feasible arrival rate vector generalizes. Also, the proof of stability for feasible arrival rate

[8]A set $\mathcal{S}$ is accessible from a state $\vec{x}$, if $\Pr(\vec{Y}(t) \in \mathcal{S}/\vec{Y}(0) = \vec{x}) > 0$, for some t.

vector holds even if the maximum number of exogeneous arrivals per slot is unbounded, if the routing policy satisfies R3 and the scheduling policy satisfies S3 or the routing and scheduling decisions are taken every slot.

Finally, we made some statistical assumptions in the beginning of this section. However, these assumptions are not critical to the results. We have proved the throughput optimality of MMRS for more generalized arrival process in technical report [26]. There we assumed that the arrival process for session n, $\{A_n(t)\}_{t=1}^{\infty}$, is a Markov modulated process. In this case, a_n is the expectation of $A_n(t)$ under the stationary distribution of the underlying Markov process $S(t)$. We showed that if the arrival rate vector $(a_1, \ldots, a_n)$ is feasible, then a quadratic Liapunov function of $\{\vec{Y}, S\}$ has negative drift when averaged over a sufficiently long interval. As in the i.i.d. case, stability results follow subsequently from work relating drift analysis and stability of Markov chains. A similar approach has been adopted in [31] while proving stability results for Markov modulated service process. For simplicity, we omit the proof here. We believe that the maximum throughput property of MMRS does not depend on the assumptions and holds for much more general arrival and service processes.

VIII. Proof for Necessity

We proceed to prove that the system is not stable if the arrival rate for session n, a_n cannot be split in portions a_n^m such that the a_n^m's satisfy the capacity condition stated in Section II. We make the following assumptions for the purpose of analysis. Arrival and service are slotted. Each session has its own exogeneous arrival stream of packets, $\{A_n(t)\}_{t=1}^{\infty}$, where $A_n(t)$ is the number of session n packets arriving in slot t. Arrival process $\{(A_1(t), \ldots, A_N(t)\}_{t=1}^{\infty}$ can be a stationary-ergodic process or a probabilistic function of a finite-state irreducible aperiodic discrete-time Markov process.[9] As mentioned in Section VI, we assume that each packet has a deterministic service time equal to one unit at each link.

We shall use the following propositions later. The first is the well-known Birkhoff ergodic theorem and the second follows from a trivial extension of a similar result for positive recurrent discrete-time Markov chains [23].

Proposition 2: If $X(t)$ is a stationary-ergodic random process, then

$$\frac{1}{n}\sum_{i=1}^{n} X_i(\omega) \to EX = \int XdP \text{ w.p. } 1.$$

Proposition 3: If $S(t)$ is a positive recurrent periodic discrete-time Markov chain, and $A(t)$ is a random process such that $\Pr(A(t) = l/S(t) = m)$ does not depend on t and given $S(t)$, $A(t)$ is conditionally independent of all past and future $S(t)$'s and $A(t)$'s, then

$$\frac{1}{n}\sum_{i=1}^{n} A_i(\omega) \to EA \text{ w.p. } 1$$

with $EA = E(E(A(t)/S(t)))$, where the outer expectation is over the stationary distribution of $S(t)$.

[9]A probabilistic function of a finite-state irreducible aperiodic Markov process can be a stationary-ergodic process as well but not necessarily so.

We first introduce certain notation we use throughout. The vector $\tilde{f} = (f_1, \ldots, f_M)$ will be denoted as a *buffer discharge* vector. A *valid* buffer discharge vector is one in which $f_i \geq 0$, $\sum_{i\in S_e} f_i \leq 1$. Set S_e as defined before to be the set of buffers sharing the same outgoing link e. Let F be the set of valid buffer discharge vectors. The vector

$$\hat{a} = (a_1^1, \ldots, a_1^{M_1}, a_2^1, \ldots, a_2^{M_2}, \ldots, a_N^1, \ldots, a_N^{M_N})$$

will be denoted *tree arrival rate vector*. Intuitively, a_n^m denotes the arrival rate to the mth tree of the nth session. However, technically speaking we are not assuming that a_n^m is the long-term rate of arrival of packets to the mth tree of the nth session and, in fact, we do not even assume the existence of these long-term averages. The *tree arrival rate vector* is just a nomenclature. If $\tilde{a} = (a_1, a_2, \ldots, a_N)$ is the arrival rate vector

$$A(\tilde{a}) = \left\{ \hat{a}\colon \sum_{m=1}^{M_n} a_n^m = a_n,\, a_n^m \geq 0, \quad m = 1, \ldots M_n,\, n = 1, \ldots N \right\}$$

is a set of valid tree arrival rate vectors for the arrival rate vector $\tilde{a}$, i.e., we denote a tree arrival rate vector valid, if it belongs to $A(\tilde{a})$. Let

$$C = \left\{ \hat{a}\colon \sum_{n=1}^{N}\sum_{m=1}^{M_n} a_n^m T_n^m \leq (1, \ldots, 1) \right\}$$

where T_n^m is the indicator vector for the mth tree of the nth session. We call an arrival rate vector *unstable* if $C \cap A(\tilde{a}) = \phi$. We shall prove that the system cannot be stable if the arrival rate vector is unstable.

A *buffer graph* is a directed graph in which each node represents a logical buffer (node i represents buffer B_i) and there is an edge from vertex i to j, if B_j is a destination of B_i, i.e., $B_j \in Z_i$. A buffer graph consists of disconnected trees. Each multicast tree in the network corresponds to a unique set of disconnected trees in the buffer graph. Let

$$f_n^m = \min_{i\colon m(i)=m,\, n(i)=n} f_i$$

i.e., f_n^m is the minimum buffer discharge component among those corresponding to the logical buffers belonging to the mth multicast tree of the nth session. Let

$$l_T = \arg\min_{i\colon m(i)=T} f_i$$

(if more than one buffer attain this minimum, the tie is broken arbitrarily). Let o_T be the root node of the tree in the buffer graph containing node l_T. Let P_T be the unique directed path from o_T to l_T in the buffer graph. Let

$$Q(n) = \{i\colon \text{node } i \text{ lies on } P_T, T \in \mathcal{T}_n\}.$$

Observe that

$$\sum_{\substack{l\in Q(n)\\ Z_l\cap Q(n)=\phi}} f_l = \sum_{m=1}^{M_n} f_n^m \tag{7}$$

$$\sum_{l\in Q(n)} \tilde{A}_l(\theta) = A_n(\theta). \tag{8}$$

($\tilde{A}_i(t)$, as defined before, is the number of exogeneous packet arrivals at B_i at slot t.) We illustrate the concept of the buffer

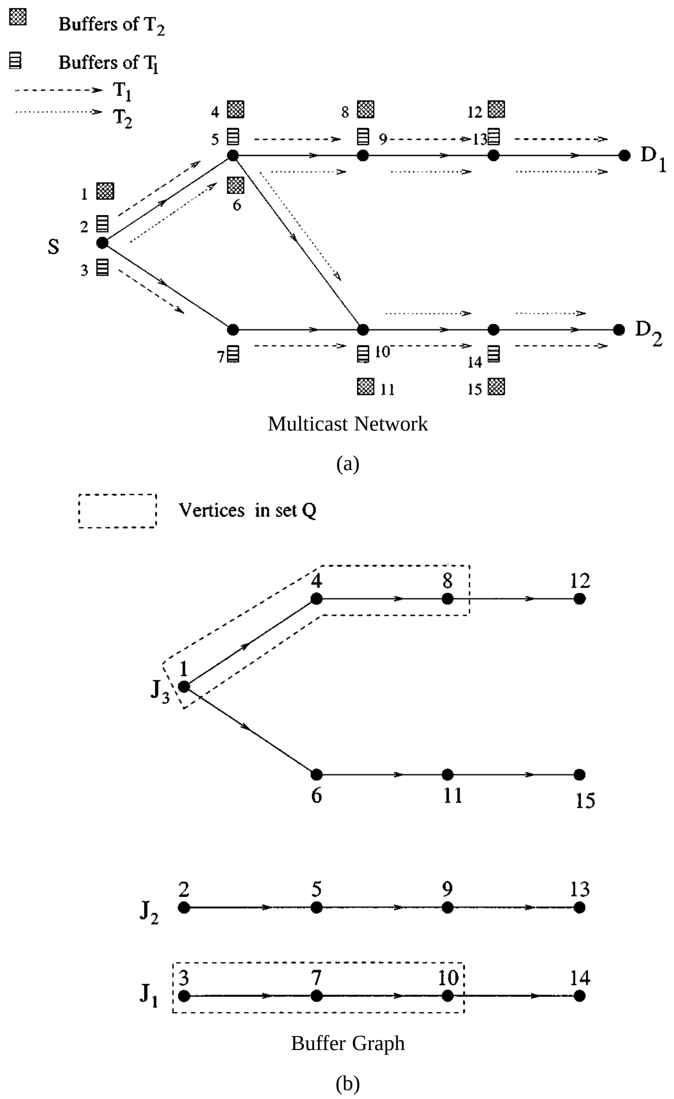


Fig. 4. (a) A network with a single session, source S and destinations D_1, D_2 and (b) the corresponding buffer graph. Logical buffers B_1, B_4, B_6, B_8, B_{11}, B_{12}, B_{15} correspond to tree T_2 and B_2, B_3, B_5, B_7, B_9, B_{10}, B_{13}, B_{14} correspond to tree T_1 in the multicast network. Trees J_1, J_2 in the buffer graph correspond to tree T_1 and tree J_3 corresponds to tree T_2. Let $f_8 = 0.3, f_{10} = 0.1. f_i = 0.4, i \notin \{8, 10\}$. Then, $Q = \{3, 7, 10, 1, 4, 8\}$.

graph in Fig. 4. Note that any exogeneous arrival in the multicast network in Fig. 4 is routed to either tree T_1 or T_2. If it is routed to T_1, it arrives at buffers B_2 and B_3. If it is routed to tree T_2 it arrives at buffer B_1. Observe that Q contains vertices $1, 3$. Thus, the total number of exogeneous arrivals of the session at any slot equals that at the logical buffers corresponding to set Q. Similarly, buffers $8, 10$ are the only ones in Q that have none of their destinations in Q and $f_8 + f_{10} = f_{T_2} + f_{T_1}$.

Lemma 3: If the arrival rate vector is unstable, then there exists $\epsilon > 0$, such that for every valid buffer discharge vector, $\tilde{f}$, there exists a session $n(\tilde{f})$ such that

$$\sum_{\substack{l \in Q(n(\tilde{f})) \\ Z_l \cap Q(n(\tilde{f})) = \phi}} f_l \le a_{n(\tilde{f})} - \epsilon$$

where Z_l is the set of destinations of buffer B_l.

Proof of Lemma 3: Let $\tilde{f} \in F$. Let $\tilde{a}$ be *unstable*. We introduce some notations

$$U_{\tilde{a}\hat{a}\tilde{f}} = \max_{m,\,n}(a_n^m - f_n^m), \qquad \hat{a} \in A(\tilde{a})$$

and

$$V_{\tilde{a}\tilde{f}} = \min_{\hat{a} \in A(\tilde{a})} U_{\tilde{a}\hat{a}\tilde{f}}.$$

$V_{\tilde{a}\tilde{f}}$ is well defined as the set $A(\tilde{a})$ is closed and bounded for every $\tilde{a}$ and $U_{\tilde{a}\hat{a}\tilde{f}}$ is a continuous function of $\hat{a}$ for every $\tilde{a}$ and $\tilde{f}$. $V_{\tilde{a}\tilde{f}} > 0$. Otherwise, there exists $\hat{a} \in A(\tilde{a})$, such that $U_{\tilde{a}\hat{a}\tilde{f}} \le 0$, i.e., $a_n^m \le f_n^m, \forall m, n$

$$\sum_{n=1}^{N} \sum_{m=1}^{M_n} a_n^m T_n^m \le \sum_{n=1}^{N} \sum_{m=1}^{M_n} f_n^m T_n^m$$
$$\le \left(\ldots, \sum_{i \in S_e} f_i, \ldots\right). \tag{9}$$

Thus,

$$\sum_{n=1}^{N} \sum_{m=1}^{M_n} a_n^m T_n^m \le (1, \ldots, 1).$$

Inequality (9) follows because the eth component of

$$\sum_{n=1}^{N} \sum_{m=1}^{M_n} f_n^m T_n^m$$

is the sum of the f_n^m's of those multicast trees of the network which pass through e and every multicast tree passing through e has one logical buffer which sends its traffic across e, and, finally, as per definition $f_n^m \le f_i$ where B_i is a buffer of the mth tree of the nth session, i.e., $m(i) = m$, $n(i) = n$.

However, then $\hat{a} \in C$ and we know that $\hat{a} \in A(\tilde{a})$. This contradicts the assumption that $\tilde{a}$ is unstable. Thus, $V_{\tilde{a}\tilde{f}} > 0$.

Next we show that there exists a session n such that

$$a_n - \sum_{m=1}^{M_n} f_n^m > 0.$$

Consider $\hat{a}$ which attains $V_{\tilde{a}\tilde{f}}$. Let $U_{\tilde{a}\hat{a}\tilde{f}}$ be attained by trees $T_1 \ldots, T_p$ of session n. $U_{\tilde{a}\hat{a}\tilde{f}} = V_{\tilde{a}\tilde{f}} > 0$. Hence, $a_n^{T_i} - f_n^{T_i} > 0, i = 1, \ldots, p$. If there does not exist a session q such that

$$a_q - \sum_{m=1}^{M_q} f_q^m > 0$$

then

$$\sum_{m=1}^{M_q} a_q^m - \sum_{m=1}^{M_q} f_q^m \le 0, \qquad \forall q, \hat{a} \in A(\tilde{a}).$$

It follows that there exists a tree T_r of the nth multicast session of the network, such that $a_n^{T_r} - f_n^{T_r} < 0$. $a_n^{T_i}$, $i = 1, \ldots, p$ could be decreased, increasing $a_n^{T_r}$, still maintaining the sum of the a_n^ms equal to a_n, yet decreasing the $\max_m(a_n^m - f_n^m)$. If the process, is repeated with other sessions attaining $U_{\tilde{a}\hat{a}\tilde{f}}$, we would obtain a $\hat{a}' \in A(\tilde{a})$, such that $U_{\tilde{a}\hat{a}'\tilde{f}} < U_{\tilde{a}\hat{a}\tilde{f}} = V_{\tilde{a}\tilde{f}}$ which contradicts the definition of $V_{\tilde{a}\tilde{f}}$.

Thus,

$$\max_n\left(a_n - \sum_{m=1}^{M_n} f_n^m\right) > 0, \qquad \text{for all } \tilde{f} \in F.$$

F is a closed and bounded set.

$$\max_n \left(a_n - \sum_{m=1}^{M_n} f_n^m \right)$$

is a continuous function of $\tilde{f}$. Hence,

$$\min_{\tilde{f}\in F} \max_n \left(a_n - \sum_{m=1}^{M_n} f_n^m \right) \quad \text{exists.}$$

Let

$$\epsilon = \min_{\tilde{f}\in F} \max_n \left(a_n - \sum_{m=1}^{M_n} f_n^m \right).$$

Since

$$\max_n \left(a_n - \sum_{m=1}^{M_n} f_n^m \right) > 0, \qquad \text{for all } \tilde{f} \in F, \epsilon > 0.$$

Thus,

$$\max_n \left(a_n - \sum_{m=1}^{M_n} f_n^m \right) \geq \epsilon > 0, \qquad \text{for all } \tilde{f} \in F.$$

Let $n(\tilde{f})$ be the session which attains the above maximum for $\tilde{f}$. Then

$$a_{n(\tilde{f})} - \sum_{m=1}^{M_{n(\tilde{f})}} f_{n(\tilde{f})}^m \geq \epsilon > 0.$$

The result follows from (7). □

Theorem 2: If the arrival rate vector $\tilde{a}$ is unstable, then

$$\sum_{i=1}^{M} B_i(t) \to_{t\to\infty} \infty \text{ a.s..}$$

Proof of Theorem 2: Let the arrival rate vector $\tilde{a}$ be unstable. Let $W \subseteq \{1, \ldots, M\}$

$$\sum_{i\in W} B_i(t) = \sum_{i\in W} \left(B_i(t-1) + \left(R\vec{E}(t) \right)_i + \tilde{A}_i(t) \right) \quad \text{(from (6))} \quad (10)$$

$$\sum_{i\in W} \left(R\vec{E}(t) \right)_i = \sum_{i\in W} ((|Z_i \cap W| - 1)E_i(t))$$
$$\geq - \sum_{i\in W,\, Z_i\cap W=\phi} E_i(t) \quad (11)$$

$$\sum_{i\in W} B_i(t) \geq \sum_{\theta=1}^{t} \left(\left(\sum_{i\in W} \tilde{A}_i(\theta) \right) - \left(\sum_{\substack{i\in W,\\ Z_i\cap W=\phi}} E_i(\theta) \right) \right). \quad (12)$$

The last inequality follows from recursive substitution using relations (10) and (11). Note that (6) holds for any arbitrary scheduling policy. However, $\vec{E}(t)$ is not updated as per (2) and (3) for any arbitrary scheduling policy. $\vec{\Gamma}(t)$ is also not updated as per (1) for any arbitrary routing policy. We do not assume these relations in this proof.

Let

$$\vec{\lambda}(t) = \frac{1}{t} \sum_{\theta=1}^{t} \vec{E}(\theta).$$

Note that

$$\sum_{i\in S_e} \lambda_i(t) = \frac{1}{t} \sum_{\theta=1}^{t} \sum_{i\in S_e} E_i(\theta) \leq 1, \qquad \forall \theta.$$

The last inequality follows as

$$\sum_{i\in S_e} E_i(\theta) \leq 1, \qquad \forall \theta.$$

Also, $\lambda_i(t) \geq 0$ Thus, $\vec{\lambda}(t) \in F$, i.e., $\vec{\lambda}(t)$ is a valid buffer flow vector

$$\begin{aligned}
\sum_{i=1}^{M} B_i(t) &\geq \sum_{i\in Q(n(\vec{\lambda}(t)))} B_i(t) \\
&\geq \sum_{\theta=1}^{t} \left(\left(\sum_{i\in Q(n(\vec{\lambda}(t)))} \tilde{A}_i(\theta) \right) - \left(\sum_{\substack{i\in Q(n(\vec{\lambda}(t)))\\ Z_i\cap Q(n(\vec{\lambda}(t)))=\phi}} E_i(\theta) \right) \right) \\
&\qquad \text{(from (12))} \\
&= \left(\sum_{\theta=1}^{t} A_{n(\vec{\lambda}(t))}(\theta) \right) - t \left(\sum_{\substack{i\in Q(n(\vec{\lambda}(t)))\\ Z_i\cap Q(n(\vec{\lambda}(t)))=\phi}} \lambda_i(t) \right) \\
&\qquad \text{(from (8) and the definition of } \vec{\lambda}(t)) \\
&\geq \sum_{\theta=1}^{t} A_{n(\vec{\lambda}(t))}(\theta) - t a_{n(\vec{\lambda}(t))} + t\epsilon \qquad \text{(from Lemma 3)} \\
&\geq t\epsilon - t \left| \frac{1}{t} \left(\sum_{\theta=1}^{t} A_{n(\vec{\lambda}(t))}(\theta) \right) - a_{n(\vec{\lambda}(t))} \right|. \qquad (13)
\end{aligned}$$

Since there are only a finite number of sessions, it follows from trivial extensions of Propositions 2 and 3 to vector-valued random processes that given any $\delta > 0$, there exists a t_0 such that

$$\left| \frac{1}{t} \left(\sum_{\theta=1}^{t} A_n(\theta) \right) - a_n \right| < \delta \text{ a.s.}$$

$\forall n \in \{1, \ldots N\}, \forall t \geq t_0$. Letting $\delta = \epsilon/2$, it follows from (13) that a.s.

$$\sum_{i=1}^{M} B_i(t) \geq \frac{t\epsilon}{2}, \qquad \forall t \geq t_0.$$

The result follows. □

If $X_u(t)$'s represent the physical buffer queue lengths, then it follows from the discussion in Section VI that

$$\sum_u X_u(t) = \sum_j \max_{i\in W_j} B_i(t)$$

where $W_1, W_2, \ldots$ constitute a partition of $\{1, \ldots, M\}$, $W_i \neq \phi$, $\forall i$. It follows that

$$\sum_u X_u(t) \geq \frac{1}{\max_j |W_j|} \sum_{i=1}^{M} B_i(t).$$

Hence, if the arrival rate vector is unstable

$$\sum_u X_u(t) \rightarrow_{t\rightarrow\infty} \infty \text{ a.s.}$$

(Theorem 2). □

IX. Conclusion

As discussed in Section I, most of the existing research in multicast routing have advocated the use of a single multicast tree per session. A significant amount of research has been directed toward the construction and the nature of the tree, e.g., whether the tree should be an SPT or a CBT and how to form these trees for a source and a set of destinations. No existing routing protocol provides for load balancing. With increase of traffic, load balancing is likely to become an important tool for congestion control. We have assumed that the set of possible multicast trees are known for a session and have focused on load balancing via dynamic selection of the appropriate tree for an incoming packet. The scheduling in current multicast network is still best effort service. We have proposed a scheduling based on local information. The throughput attained by existing multicast routing protocols such as DVMRP [5], CBT [1], PIM [6], MIP [21] are not known. We have proposed a throughput optimal routing and scheduling. Throughput-optimal algorithms in generalized multicast networks were not known before. However, some previous work exists for broadcast networks. A throughput optimal algorithm for broadcasts in a mesh network has been proposed in [19]. This happens to be a special case of ours. Also, [32] proposes a routing policy which attains at least 50% of the maximum possible throughput in an arbitrary broadcast network. Since unicast and broadcast are special cases of multicast, MMRS applies to both unicast and broadcast networks as well.

Kumar *et al.* presents general techniques for analyzing performance and stability criterion of scheduling policies in a broad class of networks in [14], [16]. Also, [15] presents a class of throughput-optimal scheduling strategies for unicast networks. However, most of these techniques apply for networks with a single and a predetermined route for the session. Throughput-optimal routing and scheduling policies have been presented for arbitrary unicast networks with any number of sessions in [29], [30]. Our scheduling policy is somewhat similar to the parametric back pressure policy (PBP) proposed in [30]. But the intricacies of *traffic multiplication* in the multicast scenario cannot be captured by the system model introduced in [30] because as opposed to traffic from a buffer j reaching one of many buffer in a given set $\mathcal{R}_j$ in the unicast scenario considered there, traffic from a buffer may need to reach multiple buffers in the multicast case. Thus, scheduling needs to be modified suitably to take this into account. Besides, the routing policies are inherently different in the two cases. In MMRS routing, the entire path which the packet follows is decided once for each packet and immediately after its arrival and, as we have discussed before, this decision is computationally simple. In PBP, the routing decision is taken freshly at every buffer. None of these decisions determine the entire path alone, but all of these decisions cumulatively determine the entire path. This makes MMRS simpler to implement. We also introduce the use of some scale factors and queue-length dependent or constant-bias terms in making scheduling decisions. These scale factors and bias terms can be used to allow limited priority to sessions over one another, reduce overall delay, packet loss, etc. Neither [29] nor [30] uses these scale factors and bias terms.

Appendix I
Proof of the Negative Drift Condition (Lemma 2)

We first introduce some notations. H is the set of possible activation vectors $\vec{E}$'s. Note that

$$H = \left\{ \vec{\gamma}: \gamma_i \in \{0, 1\}, \sum_{i\in S_e} \gamma_i \le 1 \right\}$$

where $S_e = \{i: e(i) = e, 1 \le i \le M\}$ (the set of buffers contending for service from link e).

$$V(t) = \sum_{i=1}^{M} c_i B_i^2(t).$$

We prove the negative drift condition (Lemma 2) using Lemmas 4–6.

Lemma 4: There exists a function $\psi: R \rightarrow R$ associated with the Markov chain $\vec{Y}(t)$ such that

$$\vec{D}(t+1)^T \vec{E}(t+1) \ge \max_{\vec{\gamma}\in H} \vec{D}(t+1)^T \vec{\gamma} - \psi(t)$$

where ψ satisfies the property that for any $\delta' > 0$, there exists a constant $L(\delta')$ such that $\psi(t) \le \delta' \sqrt{V(t)}$, if $V(t) \ge L(\delta')$.

Lemma 4 states that the dot product between the vector of difference of scaled backlogs of source and destination buffers $(\vec{D}(t))$ and the activation vector does not differ significantly from the maximum possible value of such a dot product. The difference is upper-bounded by a function associated with the Markov chain $\vec{Y}(t)$ whose growth rate is less than that of $V(t)$. The lemma clearly holds if the bias terms ($l_i(t)$'s) are zero, and the scheduling decisions are taken every slot, because then the scheduling policy activates buffers such that $\vec{D}(t)^T \vec{E}(t) = \max_{\vec{\gamma}\in H} \vec{D}(t)^T \vec{\gamma}, \forall t$. For the more general case, the proof follows from the fact that the scheduling decisions are taken not too infrequently and when a link is scheduled, the trees with large difference of source destination buffer backlogs are preferred over others (bias terms are also taken into account while making a decision but they are small compared to $V(t)$, for large $V(t)$). Refer to [26] for the detailed proof.

Lemma 5: There exists a constant ε_n for each session n such that

$$\sum_{B_i \in O_{T_n(t+1)n}} c_i B_i(t) \le \sum_{B_i \in O_{mn}} c_i B_i(t) + \varepsilon_n, \quad \forall m, 1 \le m \le M_n.$$

Lemma 5 states that the sum of the scaled backlogs of source buffers of the currently active tree of a session is not significantly greater than that of any other tree of the session, if at all. The scaled backlogs of the source buffers of the currently active tree of a session can exceed that of any other tree of the session by at most a constant. Clearly, this lemma holds if the routing decisions are taken every slot, and the bias terms are zero. The result holds in the more general case as well, because the routing decisions are taken not too infrequently. Also, the routing decision for a session prefers the trees with small source queue lengths over others (again, scale factors and constant bias terms are taken into account). The detailed proof can be found in technical report [26].

Lemma 6: For any t_1, t_2, $|t_2-t_1| \le \Lambda$, Λ any finite constant, given any $\delta > 0$, there exists a finite $L(T, \delta)$ such that

$$(1-\delta_2)V(t_1) \le V(t_2) \le (1+\delta_2)V(t_1), \quad \text{if } V(t_1) \ge L(\Lambda, \delta).$$

Lemma 6 states that the relative difference between $V(t_1)$ and $V(t_2)$ becomes negligible as $V(t_1)$ increases if the length of the interval $|t_2-t_1|$ is bounded. Intuitively, the result holds, because there can be at most one packet departure from and bounded number of arrivals (by assumption) to any buffer in a slot. The formal proof follows.

Proof of Lemma 6: Recall the definition of O_n in (24)

$$|B_i(t+T) - B_i(t)| \le \begin{cases} T, & B_i \notin O_{n(i)} \\ K_{n(i)}T, & B_i \in O_{n(i)}. \end{cases} \quad (14)$$

This follows from the fact that there can be at most one arrival to a nonorigin buffer (buffer not at v_n for some n) and at most $K_{n(i)}$ arrivals to a buffer at the origin node of its session, in one slot. At most, one packet can depart from a buffer in a slot. Hence, $|B_i(t+T) - B_i(t)| \le \sigma T$, where $\sigma = \max_{1\le n\le N} K_n$. Let, $\Upsilon = \sum_{i=1}^{M} c_i$. It follows that

$$\begin{aligned} &\frac{|V(t_1)-V(t_2)|}{V(t_1)} \\ &\quad \le \frac{2\sum_{i=1}^{M} c_i B_i(t_1)\sigma|t_2-t_1| + M\Upsilon\sigma^2(t_2-t_1)^2}{\sum_{i=1}^{M} c_i B_i(t_1)^2} \\ &\quad \le \delta \quad \forall t_1 \text{ s.t. } V(t_1) \ge L(\Lambda, \delta), \quad \text{since } |t_2-t_1| \le \Lambda. \end{aligned}$$

The result follows. □

Proof of Lemma 2: Let

$$\phi_2(\vec{y}) = \sum_{i=1}^{M} c_i y_i^2, \; \phi_2(\vec{Y}(t)) = V(t)$$

where $V(t) = \sum_{i=1}^{M} c_i B_i^2(t)$

$$\phi_2(\vec{y}) < \infty, \quad \forall \vec{y} \in \mathcal{X}, \; \mathcal{X} \text{ is the state space of } \vec{Y}(t). \quad (15)$$

Now

$$V(t) = (K\vec{B}(t))^T(K\vec{B}(t))$$

where $K = \operatorname{diag}(\sqrt{c_1}, \ldots, \sqrt{c_N})$. It follows that

$$V(t+1) - V(t) = (K(\vec{B}(t+1) - \vec{B}(t)))^T (K(\vec{B}(t+1) + \vec{B}(t))).$$

Substituting the expressions for $\vec{B}(t)$ from (6), and further simplification, we have

$$\begin{aligned} &V(t+1) - V(t) \\ &\quad = \left(R\vec{E}(t+1) + \tilde{A}(t+1)\right)^T K^T K \left(R\vec{E}(t+1) + \tilde{A}(t+1)\right) \\ &\qquad + 2\left(K^T K\vec{B}(t)\right)^T \left(R\vec{E}(t+1) + \tilde{A}(t+1)\right) \\ &E\left(V(t+1) - V(t) \middle/ \vec{Y}(t)\right) \\ &\quad = E\left(\left(R\vec{E}(t+1) + \tilde{A}(t+1)\right)^T K^T K \right. \\ &\qquad \left. \cdot \left(R\vec{E}(t+1) + \tilde{A}(t+1)\right) \middle/ \vec{Y}(t)\right) \\ &\qquad + 2E\left(\left(K^T K\vec{B}(t)\right)^T \left(R\vec{E}(t+1) + \tilde{A}(t+1)\right) \middle/ \vec{Y}(t)\right). \end{aligned} \quad (16)$$

For some finite positive constant α

$$\begin{aligned} &E\left(\left(R\vec{E}(t+1) + \tilde{A}(t+1)\right)^T K^T K \right. \\ &\quad \left. \cdot \left(R\vec{E}(t+1) + \tilde{A}(t+1)\right) \middle/ \vec{Y}(t)\right) \le \alpha, \quad \forall \vec{Y}(t). \end{aligned} \quad (17)$$

Since, $\vec{E}(t+1)$ is uniquely known given $\vec{Y}(t)$

$$\begin{aligned} &E\left(\left(K^T K\vec{B}(t)\right)^T \left(R\vec{E}(t+1) + \tilde{A}(t+1)\right) \middle/ \vec{Y}(t)\right) \\ &\quad = \left(K^T K\vec{B}(t)\right)^T \left(R\vec{E}(t+1) + E\left(\tilde{A}(t+1) \middle/ \vec{Y}(t)\right)\right). \end{aligned} \quad (18)$$

Since $\vec{\Gamma}(t+1)$ is uniquely known given $\vec{Y}(t)$,

$$E\left(\tilde{A}_i(t+1) \middle/ \vec{Y}(t)\right) = \begin{cases} a_n, & \text{if } B_i \in O_{T_n(t+1)n(i)} \\ 0, & \text{otherwise.} \end{cases} \quad (19)$$

$$\left(K^T K\vec{B}(t)\right)^T R\vec{E}(t+1) = -\vec{D}(t+1)^T \vec{E}(t+1)$$

$$\left(\text{since } \left(K^T K\vec{B}(t)\right)^T R = \vec{D}(t+1)^T\right)$$

$$\left(K^T K\vec{B}(t)\right)^T R\vec{E}(t+1) \le -\max_{\vec{\gamma}\in H} D(t+1)^T\vec{\gamma} + \psi(t)$$

(from Lemma 4) (20)

$$R\vec{f} = -\hat{a} \quad (21)$$

where

$$f_i = a_{n(i)}^{m(i)} \quad (22)$$

and

$$\hat{a}_i = \begin{cases} f_i, & B_i \in O_{m(i)n(i)} \\ 0, & \text{otherwise} \end{cases}$$

where $\vec{f}$ and $\hat{a}$ are column vectors with f_i and $\hat{a}_i$ as the ith components, respectively, $1 \le i \le M$.

If $i \in O_{m(i)n(i)}$, there does not exist j such that $p(i) = j$. Thus, $(R\vec{f})_i = -f_i = -\hat{a}_i$. If $i \notin O_{m(i)n(i)}$, $(R\vec{f})_i = -f_i + f_{p(i)}$, and $n(i) = n(p(i))$, $m(i) = m(p(i))$. Thus $f_i = f_{p(i)} = a_{n(i)}^{m(i)}$. Thus $(R\vec{f})_i = 0 = -\hat{a}_i$, if $i \notin O_{m(i)n(i)}$.

If the arrival rate vector is feasible

$$\sum_{i\in S_e} f_i = \sum_{n=1}^{N}\sum_{m=1}^{M_n} a_n^m(T_n^m(e)) < 1$$

(see (4)), where $T_n^m(e) = 1$, if the mth tree of the nth session passes through link e and 0 otherwise. Also, $\vec{\gamma} \in H$, iff $\sum_{i\in S_e} \gamma_i \le 1$ and $\gamma_i \in \{0, 1\}$. This means that if the arrival rate vector is feasible, $\vec{f} = \sum_{\vec{\gamma}\in H} \lambda_{\vec{\gamma}}\vec{\gamma}$ for some coefficients $\lambda_{\vec{\gamma}}$, $\vec{\gamma} \in H$, such that $\sum_{\vec{\gamma}\in H} \lambda_{\vec{\gamma}} < 1$, $\lambda_{\vec{\gamma}} \ge 0$

$$\hat{a} = -\sum_{\vec{\gamma}\in H} \lambda_{\vec{\gamma}} R\vec{\gamma} \tag{23}$$

$$\begin{aligned}\left(K^TK\vec{B}(t)\right)^T & E\left(\tilde{A}(t+1)\Big/\vec{Y}(t)\right)\\ &= \sum_{n=1}^{N} a_n \sum_{B_i\in O_{T_n(t+1)n}} c_iB_i(t)\\ &\le \sum_{n=1}^{N}\sum_{B_i\in O_n} f_ic_iB_i(t) + \varepsilon \qquad (24)\\ O_n = \bigcup_{1\le l\le M_n} O_{\mathrm{ln}}, &\qquad \varepsilon = \sum_{i=1}^{n} a_n\varepsilon_n.\end{aligned}$$

The last inequality follows using Lemma 5 and (22)

$$\begin{aligned}\sum_{n=1}^{N}\sum_{B_i\in O_n} f_ic_iB_i(t) &= \left(K^TK\vec{B}(t)\right)^T \hat{a}\\ &= \sum_{\vec{\gamma}\in H} \lambda_{\vec{\gamma}}\left(-\left(K^TK\vec{B}(t)\right)^T R\right)\vec{\gamma}\\ &\qquad\qquad \text{(from (23))}\\ &= \sum_{\vec{\gamma}\in H} \lambda_{\vec{\gamma}}\vec{D}^T(t+1)\vec{\gamma}\\ &\le \lambda \max_{\vec{\gamma}\in H} \vec{D}^T(t+1)\vec{\gamma}\\ \lambda &= \sum_{\vec{\gamma}\in H}\lambda_{\vec{\gamma}}\\ \lambda &< 1. \qquad (25)\end{aligned}$$

From (18), as well as inequalities (20), (24), and (25), we have

$$\begin{aligned}E\left(\left(K^TK\vec{B}(t)\right)^T\left(R\vec{E}(t+1)+\tilde{A}(t+1)\right)\Big/\vec{Y}(t)\right)&\\ \le -(1-\lambda)\max_{\vec{\gamma}\in H}\vec{D}^T(t+1)\vec{\gamma} + \psi(t) + \varepsilon.& \qquad (26)\end{aligned}$$

Let $m = \arg\max_{1\le i\le M} \sqrt{c_i}B_i(t)$. It follows that

$$\begin{aligned}c_mB_m(t) \ge \left(\min_{1\le i\le M}\sqrt{c_i}\right)\sqrt{\frac{V(t)}{M}},&\\ \text{since } \sqrt{c_m}\,B_m(t) \ge \sqrt{\frac{V(t)}{M}}.& \qquad (27)\end{aligned}$$

Consider a function s: $Z \to P(Z)$, Z is the set of logical buffers, $P(Z)$ is the power set of Z, $s(B_i) = Z_i$. Consider a sequence of buffers constructed as follows. The first element is B_m. The second set of elements are those in $s(B_m)$. The next set of elements consists of buffers in $s(B_j)$, for all B_j's in the set $s(B_m)$, and so on. This sequence is finite and would end in the B_j's for which $s(B_j) = Z_j = \phi$. Let

$$Q_i(t) = c_iB_i(t) - \sum_{B_j\in Z_j} c_jB_j(t).$$

Observe that

$$c_mB_m(t) = \sum_{i:\,B_i\in\text{sequence}} Q_i(t).$$

Let the sequence have X terms, $X \le M$. Thus,

$$\begin{aligned}c_mB_m(t) &\le X \max_{1\le i\le X} Q_i(t)\\ &\le M \max_{1\le i\le M}\left(c_iB_i(t) - \sum_{B_k\in Z_i} c_kB_k(t)\right).\end{aligned}$$

Hence,

$$c_mB_m(t) \le M \max_{\vec{\gamma}\in H} \vec{D}^T(t+1)\vec{\gamma} \tag{28}$$

$$\max_{\vec{\gamma}\in H} \vec{D}^T(t+1)\vec{\gamma} \ge \left(\min_{1\le i\le M}\sqrt{c_i}\right)\sqrt{\frac{V(t)}{M^3}}. \tag{29}$$

The last inequality follows from (27) and (28). From Lemma 4

$$\left.\begin{aligned}&\psi(t) \le \lambda'\sqrt{V(t)}, && \text{if } V(t) \ge L(\lambda')\\ &\lambda' = \frac{1}{2}\,\frac{(1-\lambda)\min_{1\le i\le M}\sqrt{c_i}}{M^{3/2}}, && L(\lambda') \text{ is a constant}\\ & && \text{depending upon } \lambda'\end{aligned}\right\}. \tag{30}$$

Since $\lambda < 1$, (30) follows from Lemma 4. From inequalities (26), (29), and (30)

$$\begin{aligned}E\left(\left(K^TK\vec{B}(t)\right)^T\left(R\vec{E}(t+1)+\tilde{A}(t+1)\right)\Big/\vec{Y}(t)\right)&\\ \le -\lambda'\sqrt{V(t)} + \varepsilon, \qquad \text{if } V(t) \ge L(\lambda').& \qquad (31)\end{aligned}$$

Using (16), (17), and (31)

$$\begin{aligned}E\left(V(t+1) - V(t)\Big/\vec{Y}(t)\right) \le \alpha + 2\varepsilon - 2\lambda'\sqrt{V(t)},&\\ \text{for all sufficiently large } V(t).& \qquad (32)\end{aligned}$$

Let

$$\phi_1(\vec{Y}(t)) = \max\left(1, \frac{2\lambda'\sqrt{V(t)}}{\kappa}\right), \qquad \kappa > 1$$

where λ' is the same as that in (30)

$$\begin{aligned}\phi_1(\vec{Y}(t)) &\ge 1, \qquad \forall\vec{Y}(t) \in \mathcal{X} \qquad (33)\\ t_k &= (t+1)Q + i - k - 1\\ (t+1)Q + i &= t_0 + 1\\ tQ + i &= t_{Q-1}\end{aligned}$$

$$\begin{aligned}E&\left(V(t_0+1) - V(t_{Q-1})\Big/\vec{Y}(t_{Q-1})\right)\\ &= E\left(V(t_{Q-1}+1) - V(t_{Q-1})\Big/\vec{Y}(t_{Q-1})\right)\\ &\quad + \sum_{k=0}^{Q-2} E\left(V(t_k+1) - V(t_k)\Big/\vec{Y}(t_{Q-1})\right)\\ E&\left(V(t_k+1) - V(t_k)\Big/\vec{Y}(t_{Q-1})\right)\end{aligned}$$

$$= \sum_{\tilde{y}\in\mathcal{X}} E\left(V(t_k+1) - V(t_k) \Big/ \vec{Y}(t_k) = \tilde{y}\right) \cdot \Pr\left(\vec{Y}(t_k) = \tilde{y} \Big/ \vec{Y}(t_{Q-1})\right) \quad (34)$$

since $\vec{Y}(t)$ is Markovian and $t_{Q-1} \le t_k$

$$E(V(t_k+1) - V(t_k)/\vec{Y}(t_k)) < 0, \quad \text{if } V(t_k) > \max\left(L(\lambda'), \frac{\alpha+2\varepsilon}{2\lambda'}\right)$$

(see (32)). Since $|t_k - t_{Q-1}| < Q$ (bounded), it follows from Lemma 6 that the above holds w.p. 1 if $V(t_{Q-1})$ is sufficiently large. Thus, every term in the summation in (34) is nonpositive if $V(t_{Q-1})$ is sufficiently large. Hence, for all sufficiently large $V(t_{Q-1})$, using (32)

$$E\left(V(t_0+1) - V(t_{Q-1}) \Big/ \vec{Y}(t_{Q-1})\right) \le \alpha + 2\varepsilon - 2\lambda'\sqrt{V(t_{Q-1})} \quad (35)$$

$$\begin{aligned}
E\left(\phi_2\left(\vec{Y}(t_0+1)\right)\Big/\vec{Y}(t_{Q-1})\right) - \phi_2\left(\vec{Y}(t_{Q-1})\right)
&= E\left(V(t_0+1) - V(t_{Q-1}) \Big/ \vec{Y}(t_{Q-1})\right)\\
&\le \alpha + 2\varepsilon - 2\lambda'\sqrt{V(t_{Q-1})}\\
&\le -\frac{2\lambda'}{\kappa}\sqrt{V(t_{Q-1})} \quad \text{since } \kappa > 1, \quad \lambda' > 0, \\
&\qquad \text{for all sufficiently large } V(t_{Q-1})\\
&= -\phi_1\left(\vec{Y}(t_{Q-1})\right), \quad \text{for all sufficiently large } V(t_{Q-1}).
\end{aligned}$$

$$E(\phi_2(\vec{Y}((t+1)Q+i))/\vec{Y}(tQ+i)) \le \phi_2(\vec{Y}(tQ+i)) - \phi_1(\vec{Y}(tQ+i)), \forall \vec{Y}(tQ+i) \in A^c$$

where $A = \{\vec{y}\colon \phi_2(\vec{y}) \le \beta, \vec{y} \in \mathcal{X}\}$. Note that A is a finite set because $\{\vec{B}(tQ+i)\colon V(tQ+i) \le \mu\}$ is a finite set for all $\mu \in R$ and $\vec{E}(tQ+i)$, $\vec{\Gamma}(tQ+i)$, $\vec{\Xi}(tQ+i)$, $\vec{\xi}(tQ+i)$ take values in finite sets only

$$E\left(\phi_2\left(\vec{Y}((t+1)Q+i)\right)\Big/\vec{Y}(tQ+i) = \vec{y}\right) \le \phi_2(\vec{y}) - \phi_1(\vec{y}), \quad \forall \vec{y} \in A^c, \ |A| < \infty. \quad (36)$$

Equations (15), (33), (36), and the fact that A is a finite set $\forall \beta$ show that the functions ϕ_1, ϕ_2 satisfies the properties mentioned in Lemma 2. □

Appendix II
Justification of Proposition 1

Here we show that Proposition 1 follows from the f-norm ergodic theorem of Meyn and Tweedie [18]. The f-norm ergodic theorem is stated next.

Theorem 3 [18, pp. 330–331]: Let a discrete-time Markov chain $\vec{Y}(t)$ (state space $\mathcal{X}$) be ψ-irreducible and aperiodic, and let $\phi_1 \ge 1$ be a function on $\mathcal{X}$. Then the following conditions are equivalent.

1) The chain is positive recurrent with invariant probability measure π and

$$\pi(\phi_1) = \int \pi(d\vec{x})\phi_1(\vec{x}) < \infty.$$

2) There exists some petite set C and some extended-valued nonnegative function ϕ_2 satisfying $\phi_2(\vec{x}_0) < \infty$, for some $\vec{x}_0 \in \mathcal{X}$, and

$$\nabla\phi_2(\vec{x}) \le -\phi_1(\vec{x}) + b1_C(\vec{x}), \qquad \vec{x} \in \mathcal{X}$$

where

$$\nabla\phi_2(\vec{x}) = \int P(\vec{x}, d\vec{y})\phi_2(\vec{y}) - \phi_2(\vec{x}), \qquad \vec{x} \in \mathcal{X}.$$

Let $S_V = \{\vec{x}\colon \phi_2(\vec{x}) < \infty\}$. Any of these conditions imply that for any $\vec{x} \in S_V$

$$\|P^n(\vec{x}, .) - \pi(.)\|_{\phi_1} \to 0, \qquad \text{as } n \to \infty$$

where $P^t(\vec{x}, A) = \Pr(\vec{Y}(t) \in A/\vec{Y}(0) = \vec{x})$, $A \in \mathcal{B}(\mathcal{X})$, and $\|\nu\|_f = \sup_{g\colon |g|\ge f} |\nu(g)|$, $\nu(g) = \int g d(\nu)$ for any signed measure ν and any measurable function f.

1) A Markov chain state space $\mathcal{X}$ is φ-irreducible, if there exists a measure φ on $\mathcal{B}(\mathcal{X})$ such that whenever $\varphi(A) > 0$, $L(\vec{x}, A) > 0$, for all $\vec{x} \in \mathcal{X}$, where $L(\vec{x}, A)$ is the probability that the chain reaches A starting from $\vec{x}$. Clearly, if there exists a single closed communication class $\mathcal{S}$ accessible from all other states, then the chain is φ-irreducible for all φ with $\varphi(\mathcal{X}\backslash S) = 0$. If a chain is φ-irreducible for some φ, then it is called ψ-irreducible, where ψ is a unique "maximal" irreducibility measure among the φ's for which the chain is φ-irreducible.

2) A set $A \in \mathcal{B}(\mathcal{X})$ is called ν_a-petite set if $K_a(\vec{x}, B) \ge \nu_a(B)$, for all $\vec{x} \in A$, $B \in \mathcal{B}(\mathcal{X})$, where ν_a is a nontrivial measure on $\mathcal{B}(\mathcal{X})$ and

$$K_a(\vec{x}, B) = \sum_{n=0}^{\infty} P^n(\vec{x}, B)a(n)$$

where $\{a(n)\}$ is a distribution on Z_+. Clearly, a singleton $\{\vec{x}\}$ is always a petite set $(a(1) = 1, a(n) = 0, n \ne 1, \gamma_a(A) = P(x, A), \forall A \in \mathcal{B}(\mathcal{X}))$. Petiteness of any finite set follows from the fact that the union of two petite sets is petite [18, Proposition 5.5.5 (ii), p. 122].

3) A chain is recurrent if it is ψ-irreducible and the expected number of visits to a set A, starting from a state $\vec{x}$, is ∞ for all $\vec{x} \in \mathcal{X}$, and for all A for which $\psi(A) > 0$. A chain is positive recurrent if it is ψ-irreducible, recurrent and admits an invariant probability measure π.

Thus, a Markov chain with a single closed communication class accessible from any other state is ψ-irreducible. It follows from 2) that the ϕ_2 function of Proposition 1 satisfies the requirements of item 2) of Theorem 3, with the petite set being the finite set A of Proposition 1 and b be a real number satisfying

$$b > \max_{\vec{y}\in A}\left(E\left(\phi_2\left(\vec{Y}(t+1)\right)\Big/\vec{Y}(t) = \vec{y}\right) - \phi_2(\vec{y}) + \phi_1(\vec{y})\right).$$

(Note that the maximum exists finitely because A is a finite set and the ϕ_1, ϕ_2 functions are everywhere finite.) $S_{\phi_2} = \mathcal{X}$. It follows that any Markov chain which satisfies the requirements of Proposition 1 admits an invariant probability measure π with $E(\phi_1(\vec{Y}(t)) < \infty$, where the expectation is taken with respect to probability measure π

$$g_{(t,\vec{x})}(\vec{y}) = \begin{matrix} 1, & \text{if } P^t(\vec{x}, \vec{y}) \ge \pi(\vec{y}) \\ -1, & \text{otherwise.} \end{matrix}$$

Clearly, $|g_{(t,\vec{x})}(\vec{y})| = 1 \leq \phi_1(\vec{x})$ for all t and $\vec{x}, \vec{y} \in \mathcal{X}$. Also,

$$\int g_{(t,\vec{x})}\, d(\nu) = \sum_{\vec{y} \in \mathcal{X}} |P^t(\vec{x}, \vec{y}) - \pi(\vec{y})|.$$

Thus, it follows from the later parts of the theorem that the sequence of probability measures $\{\pi_t\}$, where

$$\pi_t(A) = \Pr(\vec{Y}(t) \in A / \vec{Y}(0) = \vec{x}), \qquad A \in \mathcal{B}(\mathcal{X})$$

converges to π with the convergence metric being $\sum_{\vec{y} \in \mathcal{X}} |\pi_t(\vec{y}) - \pi(\vec{y})|$. Thus, Proposition 1 follows.

REFERENCES

[1] T. Ballardie, P. Francis, and J. Crowcroft, "Core based trees: An architecture for scalable inter-domain multicast routing," in *Proc. ACM SIGCOMM*, Ithaca, NY, Sept. 1993, pp. 85–95.

[2] J. Bolot and A. Vega Garcia, "Control mechanisms for packet audio in the internet," in *Proc. IEEE Infocom'96*, San Francisco, CA, Apr. 1996, pp. 232–239.

[3] D. Cheriton and S. Deering, "Host groups: A multicast extension for datagram internetworks," in *Proc. ACM/IEEE 9th Data Communications Symp.*, New York, Sept. 1985, pp. 172–179.

[4] F. Chiussi, Y. Xia, and V. Kumar, "Performance of shared-memory switches under multicast bursty traffic," *IEEE J. Select. Areas Commun.*, vol. 15, Apr. 1997.

[5] S. Deering and D. Cheriton, "Multicast routing in datagram internetworks and extended LANs," *ACM Trans. Computer Syst.*, vol. 8, no. 2, pp. 54–60, Aug. 1994.

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

[7] C. Diot, W. Dabbous, and J. Crowcroft, "Multipoint communication: A survey of protocols, functions, and mechanisms," *IEEE J. Select. Areas Commun.*, vol. 15, Apr. 1997.

[8] H. Eriksson, "Mbone: The multicast backbone," *Commun. Assoc. Comput. Mach.*, vol. 37, no. 8, pp. 54–60, Aug. 1994.

[9] R. Frederick, "NV-X11 video conferencing tool," XEROX PARC, *Unix Manual Page*, 1992.

[10] V. Hardeman, M-A. Sasse, and I. Kouvelas, "Successful multi-party audio communication over the internet," *Commun. Assoc. Comput. Mach.*, vol. 41, no. 5, pp. 74–80, 1997.

[11] C. Huitema, *Routing in the Internet*. Englewood Cliffs, NJ: Prentice Hall, 1995, ch. 11.

[12] IBM Corp., "Technical Reference PC Network," Doc. 6322916.

[13] T. Kozaki, N. Endo, Y. Sakurai, O. Matsubara, M. Mizukami, and K. Asano, "32×32 shared buffer type ATM switch VLSI's for B-ISDN's," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 1239–1247, Oct. 1991.

[14] S. Kumar and P. R. Kumar, "Performance bounds for queueing networks and scheduling policies," *IEEE Trans. Automat. Contr.*, vol. 39, pp. 1600–1611, Aug. 1994.

[15] S. Kumar and P. R. Kumar, "Fluctuation smoothing policies are stable for stochastic re-entrant lines," *J. Discr. Event Dyn. Syst.: Theory Applic.*, vol. 6, no. 4, pp. 361–370, Oct. 1996.

[16] P. R. Kumar and S. P. Meyn, "Duality and linear programs for stability and performance analysis of queueing networks and scheduling policies," *IEEE Trans. Automat. Contr.*, vol. 41, pp. 4–17, Jan. 1996.

[17] S. McCanne and V. Jacobson, "Vic: A flexible framework for packet video," in *Proc. ACM Multimedia*, Nov. 1995.

[18] S. Meyn and R. Tweedie, *Markov Chains and Stochastic Stability*. New York: Springer-Verlag, 1993.

[19] E. Modiano and A. Ephremides, "Efficient algorithms for performing packet broadcasts in a mesh network," *IEEE/ACM Trans. Networking*, vol. 4, pp. 638–648, Aug. 1996.

[20] J. Moy, "Multicast routing extensions for OSPF," *Commun. Assoc. Comput. Mach.*, vol. 37, no. 8, pp. 61–66, Aug. 1994.

[21] M. Parsa and J. J. Garcia-Luna-Aceves, "A protocol for scalable loop-free multicast routing," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 316–331, Apr. 1997.

[22] G. Rouskas and I. Baldine, "Multicast routing with end-to-end delay and delay variation constraints," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 346–358, Apr. 1997.

[23] S. Ross, *Stochastic Processes*. New York: Wiley.

[24] Sun Microsystems, *Remote Procedure Call Reference Manual* Mountain View, CA, Oct. 1984.

[25] H. Saito, H. Yamanaka, H. Yamada, M. Tuzuki, H. Koudoh, Y. Matsuda, and K. Oshima, "Multicasting function and its LSI implementation in a shared multibuffer ATM switch," in *Proc. IEEE INFOCOM'94*, Toronto, ON, Canada, June 1994, pp. 315–322.

[26] S. Sarkar and L. Tassiulas, "A framework for routing and congestion control in multicast networks," Inst. Syst. Res. and Univ. Maryland, College Park, Tech. Rep. TR98-64, 1998.

[27] M. Satyanarayanan and F. Siegal, "MultiRPC: A parallel remote procedure call mechanism," Carnegie-Mellon Univ., Pittsurg, PA, Tech Rep. CMU-CS-86-139, Aug. 1986.

[28] A. Shaikh and K. Shin, "Destination-driven routing for low-cost multicast," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 373–381, Apr. 1997.

[29] L. Tassiulas and A. Ephremides, "Stability properties of controlled queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Automat. Contr.*, vol. 37, pp. 1936–1946, Dec. 1992.

[30] L. Tassiulas, "Adaptive back-pressure congestion control based on local information," *IEEE Trans. Automat. Contr.*, vol. 40, pp. 236–250, Feb. 1995.

[31] ——, "Scheduling and performance limits of networks with constantly changing topology," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1067–1073, May 1997.

[32] E. Varvarigos and A. Banarjee, "Routing schemes for multiple random broadcasts in arbitrary network topologies," *IEEE Trans. Parallel Distrib. Syst.*, vol. 7, pp. 886–895, Aug. 1996.

[33] Vat web server. [Online]. Available: http://www-nrg.ee.lbl.gov/vat/

[34] B. Vickers, M. Lee, and T. Suda, "Feedback control mechanisms for real-time multipoint video services," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 512–530, Apr. 1997.

[35] L. Wei and D. Estrin, "The tradeoffs of multicast trees and algorithms," in *Proc. Int Conf. Computer and Communication Networks*, San Francisco, CA, Sept. 1994.

[36] White board software. [Online]. Available: ftp://ftp.ee.lbl.gov/conferencing/wb/