A Framework for Routing and Congestion Control for Multicast Information Flows

Saswati Sarkar and *Leandros Tassiulas*
Dept. of Electrical Engineering and Institute for Systems Research
University of Maryland, College Park, MD, USA
email addresses: swati@isr.umd.edu, leandros@isr.umd.edu

Abstract

We propose a new multicast routing and scheduling algorithm called multipurpose multicast routing and scheduling algorithm (MMRS). The routing policy load balances amongst various possible routes between the source and the destinations, relying its decisions on the message queue lengths at the source node. The scheduling amongst various sessions sharing links is devised such that the flow of a session depends on the congestion of the next hop links. MMRS is throughput optimal and computationally simple. It can be implemented in a distributed, asynchronous manner. It has several parameters which can be suitably modified to control the end to end delay, 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

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. Single destination or 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 and experimental data, distributed interactive simulation (DIS) activities such as tank battle simulations. Many distributed systems such as the V System[3] and the Andrew distributed computing environment[24], popular protocol suites like Sun's broadcast RPC service[21] and IBMs NetBIOS[12] are using multicasting. Multicasting has been used primarily in the Internet, but future ATM networks are likely to deploy multicasting in a large scale, particularly in applications like broadcast video, video-conferencing, multiparty telephony and workgroup applications[4].

[1]Corresponding Author

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 communications. 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 amongst various classes.

Significant amount of research has been directed towards 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 forking nodes. This differs from 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 (SPT's), also known as source based trees and group shared trees[33]. 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)[17]. 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[18] 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, that is 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 switch over takes place, the core based tree 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 amongst 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, 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 LAN emulation in ATM ABR services. Many envisioned applications in ATM ABR traffic are multicast in nature[29].

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[14], vat[31], nv[9], rat[10] and freephone[2] (audio packets are reordered in application play out buffer), workspace applications like Wb[34]. 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 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. Choice of the least total cost tree amongst all possible trees between a source and a group of destinations is the usual approach. 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. So computation of a good or rather near optimal route based on the total tree cost at reasonably regular intervals is computationally very expensive if the set of possible trees is even a moderately large subset of all possible trees between a source and the destinations.

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 base 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 least queue length or rather a 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 sessions which have the congestion[2] at downstream[3] 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. This spreads the congestion to the upstream nodes. Thus if a tree is congested then after some time the congestion would be reflected in the source node, more precisely the source node will soon have a large message queue for the tree and the routing policy would route message 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

[2]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.

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

computationally simply and needs only local information[4] 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 flows which fetch a greater revenue to the network and also to those which demand a particular quality of service from the nature of their application, e.g., real time traffic (voice, video) must have very low delays, but this is not a stringent requirement for data traffic. 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 feedbacks often lead to feedback *implosion*. Thus MMRS has various convenient features which render it attractive from implementational point of view.

Under some statistical assumptions on the message arrival and service process, MMRS attains maximum possible throughput in an arbitrary multicast network. Thus MMRS is optimal in some sense. 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. We have included symbol tables towards the end of this paper. The symbol tables indicate against each symbol the page where the symbol has been defined.

II. Multicast transport network model

The network is modelled by an arbitrary topology directed graph G, where $G = (V, E)$. V represents the set of nodes and E the set of directed links. There exists an edge directed from u_1 to u_2 in G iff there exists a directed link between the corresponding nodes in the network. 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 $(v_1, U_1), \ldots, (v_N, U_N)$. T_v^U denotes a directed tree with root v and leaves in U, $v \in V$, $U \subseteq V$. A directed tree $T_{v_n}^{U_n}$ can carry the traffic of session n

[4]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. So strictly speaking some amount of nonlocal information is necessary, but the scheduler can compute this "nonlocal" information from binary bits sent from the destination buffers sometime during the scheduling decision interval. We discuss this in detail later.

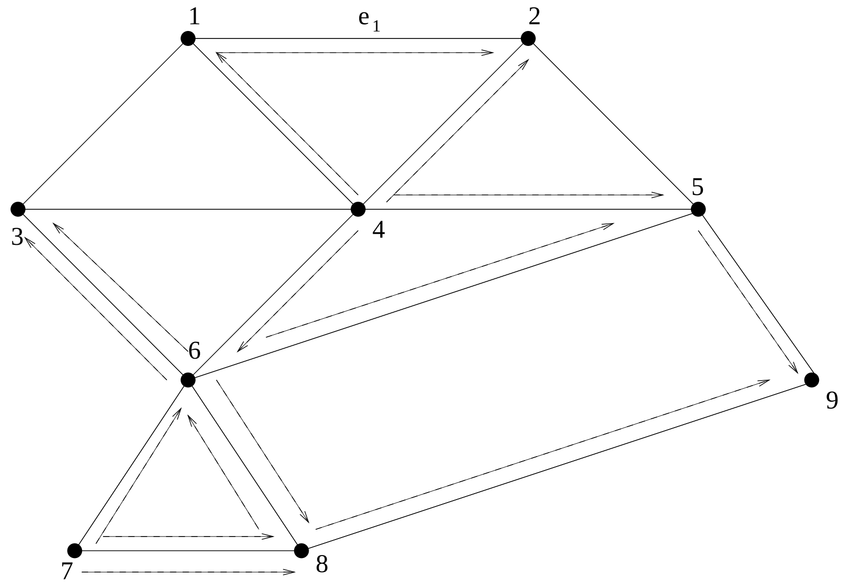

Fig. 1. A directed graph representing a network.

with source v_n and destination set U_n. A collection $\mathcal{T}_n$ of eligible multicast trees are prespecified as the trees through which session n traffic can be transported. $\mathcal{T}_n$ may include all possible trees $T_{v_n}^{U_n}$ which can carry the traffic from v_n to U_n, or a proper subset thereof. $|\mathcal{T}_n| = M_n$. 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. We now illustrate the model with an example.

Example II.1: Consider the network represented by the directed graph in figure 1. There are 9 nodes represented by the vertices and 15 links represented by the edges. There are two multicast sessions, session 1 and 2. Source nodes of sessions 1 and 2 are 4 and 7 respectively. The destination nodes are $\{2,5,9\}$ and $\{3,6,8\}$ respectively. Thus $(v_1, U_1) = (4, \{2,5,9\})$ and $(v_2, U_2) = (7, \{3,6,8\})$. $\mathcal{T}_1 = \{T_1, T_2\}$, $\mathcal{T}_2 = \{T_3, T_4\}$. $T_1 = \{(4,2),(4,5),(5,9)\}$. $T_2 = \{(4,1),(1,2),(4,6),(6,5),(6,8),(8,9)\}$. $T_3 = \{(7,8),(7,6),(6,3)\}$, $T_4 = \{(7,8),(8,6),(6,3)\}$, 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 3 edges.

We do not impose any particular structure on $\mathcal{T}_n$; $\mathcal{T}_n$ can consist of all directed trees $T_{v_n}^{U_n}$ between source v_n and the set of destinations U_n. But in a virtual circuit like scenario, where the packets contain only route identifiers and not the entire route, this would generate huge routing table entries at the routers because the total number of possible multicast trees $T_{v_n}^{U_n}$ is considerably large. So it is realistic to assume that $\mathcal{T}_n$ will be a proper subset of the set of all directed trees $T_{v_n}^{U_n}$. 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 routers need only have entries regarding currently active trees. Thus memory constraint may not force $\mathcal{T}_n$ to be a proper subset of the set of all directed trees $T_{v_n}^{U_n}$. But there may be other constraints, e.g., all multicast trees $T_{v_n}^{U_n}$ may not satisfy the requirements of session n, e.g., session n may demand certain quality of service gurantees in terms of the end-to-end delay along the individual paths from the source to each of the destination nodes and possibly a bound on the variation among the delays along the individual source destination paths. 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[19]. In high speed environments the end to end delays depend primarily on the propagation delays. So $\mathcal{T}_n$ can consist of only those trees which satisfy the requirements of session n, or a proper subset thereof.

Informally the necessary condition for system stability[5] is that the sum of arrival rates of traffic in all the trees of the same or different sessions passing through a link e does not exceed the link capacity, i.e.,

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

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 gurantees 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 Example II.1. The resource allocation policy would decide at the appropriate time which tree the traffic would follow. 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 Example II.1. So the resource allocation policy would decide at the appropriate time the trees that should be scheduled on the links. Intuitively, it should “spread out” the congestion in the network, i.e., if an upstream node

[5]We defer the formal definition of “arrival rate” and “system stability” till Section VI.

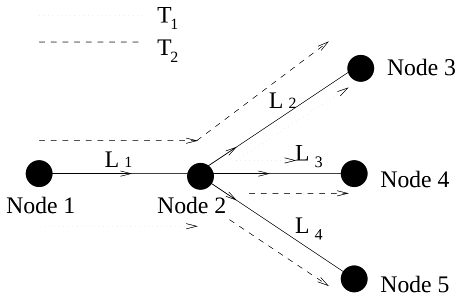

Fig. 2. Tree T_1 passes through links L_1, L_2, and L_3. Tree T_2 passes through links L_1, L_2, L_3 and L_4.

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. These decisions can also be taken based 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 [27]. 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 would discuss the policy we propose in perspective of existing work in the unicast and broadcast context in Section IX.

The multipurpose multicast routing and scheduling policy (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

We first present an informal description of MMRS. It takes routing and scheduling decisions at intervals, the intervals satisfy some properties to be described later. 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 is to schedule service at a link to 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, amongst all the trees with nonempty buffers contending for service on the same outgoing link (ignoring some bias terms for the moment), at the decision instant. The scheduling

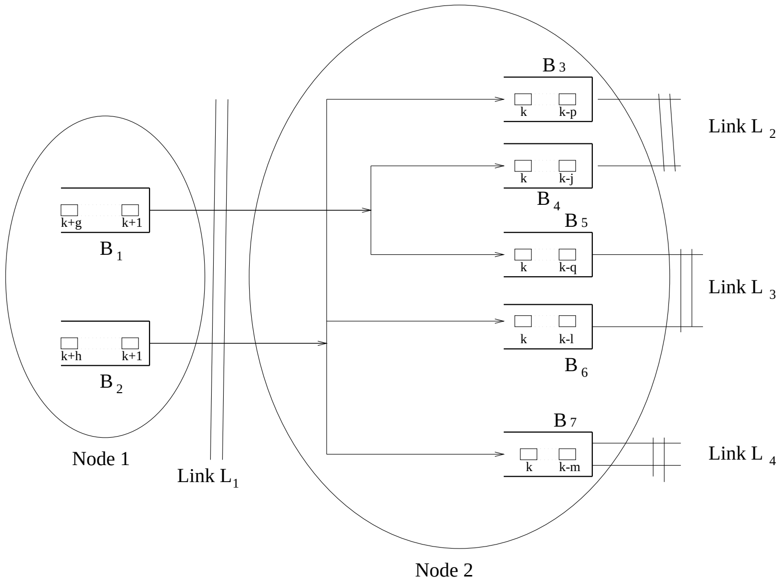

Fig. 3. Logical buffers at Node 1 and Node 2 of Figure 2.

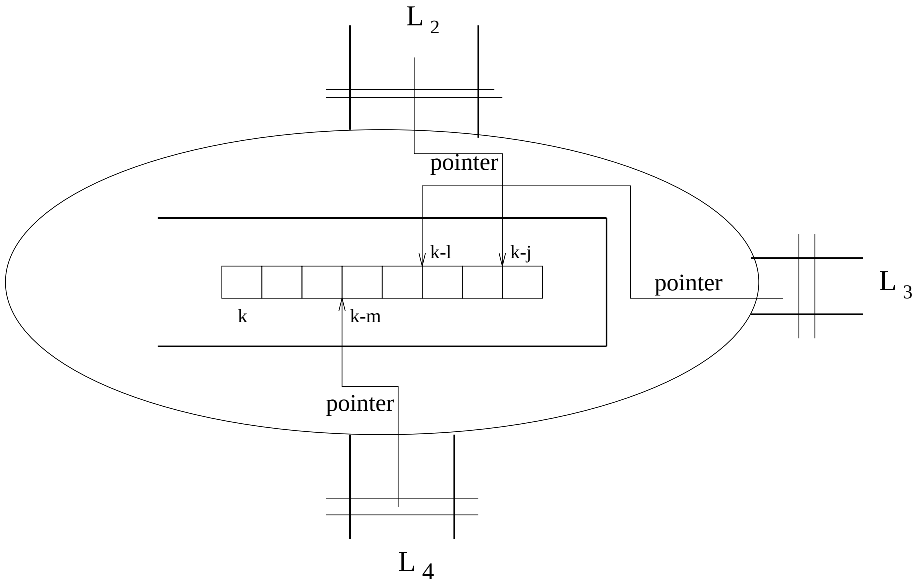

Fig. 4. A physical buffer storing T_2 packets at Node 2 of Figure 2.

policy for trees on the same outgoing 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* below. All symbols used in this section have been listed in symbol table of page 45.

Consider session n, (v_n, U_n) and edge e of G. 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 Figure 1. $B_{mne}(t)$ is the number of session n packets[6] travelling through the mth tree in $\mathcal{T}_n$ waiting at $o(e)$ at the end of slot t (or the

[6]For simplicity we state MMRS for slotted arrival and service, i.e., consider packetized traffic only. It can be easily

beginning of slot $t+1$) for travelling to $d(e)$ through link e in slot $t+1$. B_{mne}s are the backlogs of the logical buffers.

Example IV.1: Consider Figures 2 and 3. Tree T_1 of session 1 passes through link L_1, L_2, L_3 and T_2 of session 2 passes through link L_1, L_2, L_3, L_4 in Figure 2. Figure 3 shows the $B_{mne}(t)$s. $B_1(t)$ 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 $B_{T_2 2 L_4}(t)$.

We would like to point out that the logical buffers may not always represent separate memory locations, particularly for the different edges with the same origin node. It may be necessary to transmit the same packet over multiple links originating from the same node on account of multicast, e.g., in Figure 2 any T_1 packet must be transmitted across links L_2 and L_3 from node 2. Similarly any T_2 packet at Node 2 must be transmitted across links L_1, L_2 and L_3 from node 2. It is wasteful to store copies of the same packet in different physical buffers B_{mne} meant for different links at the same node ($B_{T_1 1 L_2}(t)$ and $B_{T_1 1 L_3}(t)$ i.e., $B_4(t)$ and $B_5(t)$ respectively in Figure 3). It is necessary though to keep track of the $B_{mne}(t)$s for the implementation of the policy we shall describe shortly. This can be done without storing multiple copies of the same packet in separate memory locations. There can be a single physical buffer at each node for storing all packets (one copy each) of a tree travelling through the node. A packet remains in the buffer till it has been transmitted across all the necessary links originating from the same node. Each link can easily keep track of the number of packets of each tree, it has still to transmit, using pointers, i.e., by maintaining a pointer at the first packet it needs to transmit and sliding the pointer backwards when the packet is transmitted. These numbers are precisely the $B_{mne}(t)$s.

Example IV.2: Figure 4 shows the physical buffer at Node 2 for tree T_2 of Example IV.1. This buffer contains the tree T_2 packets at Node 2. It currently has packets numbered $k-j, \ldots, k$. All of them must be transmitted across link L_2. L_2 maintains a pointer at the first packet it has to transmit, 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. Figure 3 shows the contents of the separate logical buffers for this tree at Node 2 (B_3, B_6 and B_7). $k+1$th and subsequent packets of T_2 are still waiting at Node 1 for transmission across Link L_1 to Node 2 (Refer to figure 3).

generalized to more general cases.

For simplicity, we will refer to $B_{mne}(t)$s, $m = 1, \ldots, M_n$, $e \in E$, $n = 1, \ldots, N$ as $B_1(t), B_2(t), \ldots,$ $B_M(t)$. (e.g., $B_1(t)$ denotes $B_{T_1 1 L_1}(t)$ in figure 3.). We assume there are 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 all packets in B_i will be transmitted through the same link, $e(i)$. Since all the packets in B_i travel through the same tree, they had been at the same buffer B_j before they reached B_i. 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.3: Refer to Example IV.1. $n(1) = n(4) = n(5) = 1$. $n(2) = n(3) = n(6) = n(7) = 2$. This is because buffers B_1, B_4, B_5 carry session 1 packets and the other buffers carry session 2 packets. $u(1) = u(2) = 1$, $u(3) = \ldots = u(7) = 2$, $e(1) = e(2) = L_1$, $e(3) = e(4) = L_2$, $e(5) = e(6) = L_3$, $e(7) = L_4$. 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.

Routing policy is necessary for exogeneous arrivals. Exogeneous arrivals for session n are routed to trees in $\mathcal{T}_n$ and thereafter it is only scheduling service to contending buffers, i.e., buffers at the origin of the same link.

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 $\vec{\Gamma}(t) = (T_1(t), \ldots, T_N(t))$, $1 \leq T_n(t) \leq M_n$. Let O_{mn} be the set of buffers of the mth tree of $\mathcal{T}_n$ at v_n, the source node of seesion n, i.e., $O_{mn} = \{B_i : n(i) = n, m(i) = m, u(i) = v_n\}$. Consider Example IV.1. 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. $O_{11} = \{B_1\}$ and $O_{12} = \{B_2\}$. 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.,

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

where c_is 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 would discuss the significance of c_is and C_{mn}s later.

The routing times $\{\omega_\iota^n\}$ may be:

1. At fixed intervals $\omega_{\iota+1}^n - \omega_\iota^n = T_r$. R1
2. Of bounded difference $0 \le \omega_{\iota+1}^n - \omega_\iota^n \le T_r$, $\omega_{\iota+1}^n - \omega_\iota^n$ i.i.d.[7] 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 $t+1 \in \{\omega_\iota^n\}_{\iota=1}^{\infty}$

   $$\text{if there exists } m \in M_n \text{ s.t.} \left(\sum_{B_i \in O_{T_n(t)n}} c_i B_i(t) \right) + C_{T_n(t)n} > \left(\sum_{B_i \in O_{mn}} c_i B_i(t) \right) + C_{mn} + \varrho_n \quad \text{R3}$$

   Informally this means that routing decision is not taken for session n until the currently active tree is "sufficiently" congested, and the congestion is reflected in the origin buffer lengths. 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.4: Refer to Example IV.1. Now let both T_1 and T_2 be session 1 trees numbered 1, 2 respectively. Let node 1 be the source of session 1. $O_{11} = \{B_1\}$, $O_{21} = \{B_2\}$. Let $c_1 = c_2 = 1$, $C_{11} = C_{21} = 0$. Let figure 3 show the buffers just prior to ω_ι^1 (a routing decision instant for session 1). $B_1(\omega_\iota^1 - 1) = g$, $B_2(\omega_\iota^1 - 1) = h$. Let $g < h$. $T_1(t) = 1$, for all t such that $\omega_\iota^1 \le t < \omega_{\iota+1}^1$. 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 $B_1(\omega_{\iota+1}^1 - 1) < B_2(\omega_{\iota+1}^1 - 1)$, $\omega_{\iota+1}^1 \le t < \omega_{\iota+2}^1$. If $T_1(t) = 2$ at $t = \omega_{\iota+1}^1$ 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 $[\omega_\iota^1, \omega_{\iota+1}^1)$ are routed to tree T_1 iff $c_1 g + C_{11} \le c_2 h + C_{21}$.

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

A fresh decision is taken at $\omega^1_{\iota+1}$ on the basis of $c_1B_1(\omega^1_{\iota+1}-1)+C_{11}$ and $c_2B_2(\omega^1_{\iota+1}-1)+C_{21}$.

Next we describe the scheduling. $\vec{E}(t+1)$ is the activation vector with M compnents, 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 $Z_i\neq\phi$, 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. $P_e(t+1)=\{B_i : e(i)=e, B_i(t)>0\}$

$$\begin{aligned}D_i(t+1) &= c_iB_i(t)-\sum_{B_k\in Z_i}c_kB_k(t)\;\; i=1,\ldots,M\\ s_e(t+1) &= \arg\max_{i:B_i\in P_e(t+1)}\left(l_i(t+1)+D_i(t+1)\right),\text{ if } P_e(t+1)\neq\phi\\ s_e(t+1) &= -1,\text{ if } P_e(t+1)=\phi\end{aligned}$$

$D_i(t)$ 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. 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 $\vec{B}(t)=(B_1(t),\ldots,B_M(t))^T$ and g_i is a function from R^M to R. Typically all $g_i(\vec{b})$s would be constants. We would discuss the use of $g_i(\vec{b})$s in Section V. Let $S_e=\{i : e(i)=e, 1\leq i\leq M\}$ (the set of buffers contending for service from link e). For example $S_{L_1}=\{1,2\}$, $S_{L_4}=\{7\}$ in Example IV.3.

$$\text{If } t+1\in\{\Omega^e_\iota\}^\infty_{\iota=1}, i\in S_e,\; E_i(t+1)=\begin{cases}1, & i=s_{e(i)}(t+1),\\ & l_i(t+1)+D_i(t+1)>0\\ 0 & \text{otherwise.}\end{cases}\tag{2}$$

$$\text{If } t+1\notin\{\Omega^e_\iota\}^\infty_{\iota=1}, i\in S_e,\; E_i(t+1)=\begin{cases}E_i(t), & B_i(t)>0,\\ 0 & \text{otherwise.}\end{cases}\tag{3}$$

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.

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

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

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 $D_i(\Omega^e_\iota) + l_i(\Omega^e_\iota) \leq 0$, 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:

1. At fixed intervals $\Omega^e_{\iota+1} - \Omega^e_\iota = T_s$. S1
2. Of bounded difference $\Omega^e_{\iota+1} - \Omega^e_\iota \leq T_s$, $\Omega^e_{\iota+1} - \Omega^e_\iota$ 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. $E_{j_e(t)}(t) = 1$, $D_{j_e(t)}(t+1) + l_{j_e(t)}(t+1) \leq D_{p_e(t+1)}(t+1) + l_{p_e(t+1)}(t+1) - \varsigma_{e1}$ S3a

(b) The $D_i + l_i$ of the currently scheduled buffer becomes sufficiently negative. $E_{j_e(t)}(t) = 1$, $D_{j_e(t)}(t+1) + l_{j_e(t)}(t+1) \leq -\varsigma_{e2}$ 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.

$E_{j_e(t)}(t) = 0, \qquad D_{p_e(t+1)}(t+1) + l_{p_e(t+1)}(t+1) \geq \varsigma_{e3}$ S3c

$\varsigma_{e1}, \varsigma_{e2}, \varsigma_{e3}$ are prespecified positive real numbers. Again informally this means that 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.5: Refer to Example IV.1. Let Figure 3 show the buffers just prior to $\Omega^{L_1}_\iota$ (a scheduling decision instant for link L_1). $B_1(\Omega^{L_1}_\iota - 1) = g$, $B_2(\Omega^{L_1}_\iota - 1) = h$, $B_3(\Omega^{L_1}_\iota - 1) = j+1$, $B_4(\Omega^{L_1}_\iota - 1) = p+1$, $B_5(\Omega^{L_1}_\iota - 1) = q+1$, $B_6(\Omega^{L_1}_\iota - 1) = l+1$, $B_7(\Omega^{L_1}_\iota - 1) = m+1$. Let $c_i = 1$ and $g_i(\vec{b}) = 0\ \forall i$, i.e., $l_i(t) = 0$, for all i, t. $D_1(\Omega^{L_1}_\iota) = g - p - q - 2$. $D_2(\Omega^{L_1}_\iota) = h - j - l - m - 3$. Let $p = q = j = l = m = 1$, $g = 5$, $h = 8$. Thus $D_1(\Omega^{L_1}_\iota) + l_1(\Omega^{L_1}_\iota) = 1$ and $D_2(\Omega^{L_1}_\iota) + l_2(\Omega^{L_1}_\iota) = 2$. 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 $D_1(\Omega^{L_1}_\iota) + l_1(\Omega^{L_1}_\iota)$ and $D_2(\Omega^{L_1}_\iota) + l_2(\Omega^{L_1}_\iota)$ 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 the above 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 on 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_is (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 through it. This is so if there is a connection setup phase associated with every session initiation, like in virtual circuit scenario. In that case the tree information may be contained in a number identifying the tree and the node can determine the outgoing edges along which the packet must travel from the tree and the session numbers. This enables the node to determine the B_{mne}s to which the packet belongs. In 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 can be the tree and the session numbers or the explicit tree path, the tree number and the session number depending upon whether the nodes know the tree paths or not.

V. Discussion

- It is interesting to note that the congestion in the entire tree need not be taken into account for routing. Any congestion downstream will ultimately cause congestion in the origin buffers and will prevent arrivals at the tree and this would help to clear the downstream congestion. One can think of policies which take congestion in the entire tree into account while routing and thus respond early to downstream congestion. These policies can not generate greater throughput. It is not clear how MMRS would compare with those with respect to other performance criteria, e.g., delay, expected queue lengths at the buffers, etc. As we discussed in Section I, taking routing decision on the basis of congestion in the entire trees is computationally expensive even if the routing decision intervals are long. Besides it is unrealistic to assume, particularly in large networks that the routers will have access to the necessary current global information.

- One can use the c_is to give limited priority to some sessions over others without affecting the throughput. It is generally expected that c_is for all buffers of a particular session would be equal. However they may be chosen differently in accordance with the physical buffer sizes, so as to bring down the packet loss, as explained later. Increasing the c_is for a session over others decreases the delay of the packets of the session at the expense of greater delay experienced by packets of other sessions. Thus one would expect the c_is to be higher for real time sessions like audio, video and possibly for applications which fetch greater revenue. l_is can serve the same purpose, i.e., give limited priority to sessions over each other. A judicious choice of function $g_i(\vec{b})$s can decrease the expected delay of the packets in the entire network.
- If $g_i(\vec{b})$ is a function of b_i only and not of the entire vector $\vec{b}$, then MMRS can be implemented in a decentralized manner, because the scheduler at every link need only know the queue lengths of the logical buffers at the source and destination of the link at the scheduling instants. If the scheduler of link e is located at the source, and $E_j(\Omega_{\iota j}^{e(j)})$ is available to it before Ω_ι^e, for each buffer B_j at the destination node of e, $d(e)$, where $\Omega_{\iota j}^{e(j)}$ is the last scheduling decision slot for $e(j)$ before Ω_ι^e, it can compute $B_j(t)$ for each buffer B_j at its destination in a recursive fashion and take the scheduling decision accordingly at Ω_ι^e. Since $E_j(t) \in \{0,1\}$, it may be relatively simple to communicate the necessary $E_j(t)$s instead of the corresponding $B_j(t)$s, especially if $\Omega_\iota^e - \Omega_{\iota j}^{e(j)}$ is small for some j. Note that $\Omega_{\iota j}^{e(j)} < \Omega_\iota^e$, i.e., $E_j(t)$ at the last scheduling decision slot strictly before Ω_ι^e is necessary for recursive computation. So there is at least 1 slot for propagation of a binary number. The recursive computation takes negligible time.
- If $g_i(\vec{b})$ is a computationally efficient function of b_i, then MMRS can be implemented in real time. Typically we would expect the l_is to be constants independent of b_is.
- One can use C_{mn}s to give limited priority to some trees of a session n over some other trees, while taking 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 for the session only when other trees of the session are significantly congested. This purpose can be achieved by setting a high threshold C_{mn} for the tree, so that $\sum_{B_i \in O_{mn}} c_i B_i(t) + C_{mn}$ is the minimum amongst all trees of session n, only when other trees have high congestion. Manipulation of c_is can serve the same purpose, but c_is also affect scheduling whereas C_{mn}s affect routing in the desirable manner without affecting scheduling. In fact MMRS retains its throughput optimality, even if C_{mn}s are replaced by queue length dependent bias, e.g., $f_{mn}(\vec{b})$ as long as $\lim_{||c\vec{b}|| \to \infty} \frac{f_{mn}(\vec{b})}{||c\vec{b}||} = 0, \forall m, n$, but again computation of $f_{mn}(\vec{b})$ may require global information while taking routing decisions, or computation may be expensive depending on the nature of the functions. So we suggest the usage of constant and queue length independent bias while taking routing decisions.
- The scheduling policy spreads the congestion in the system. A link would not serve a tree if the

corresponding buffer at the destination of the link has a large message queue length as compared to the source buffer, even if it has to idle. Again if a source buffer at a link has a large queue length as compared to the destination buffer, then the corresponding tree would be served by the link, even if there exists other contending trees with destination buffers not so overcrowded as compared to source buffers. This reduces the congestion at an already congested buffer at the expense of that at a possibly lightly loaded buffer. This would reduce packet loss. The parameters c_is and $g_i(\vec{b})$s can be suitably modified to reduce packet loss even further, if necessary. If a physical buffer is small in size as compared to those upstream[8] and downstream[9] then the c_is corresponding to the relevant logical buffers, (e.g., logical buffers B_3, B_6, B_7 correspond to the physical buffer at Node 2 for tree T_2 in Example IV.2, Figures 3 and 4) can be set higher than the corresponding ones at upstream and downstream. Thus the scaled queue lengths ($c_i B_i(t)$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 physical buffer would idle frequently and the links serving messages stored in the physical buffer would idle rarely. The queue length at the physical buffer would be small at the expense of larger queue lengths at larger sized physical buffers upstream and downstream. This would reduce the overall packet loss. If a physical buffer is small as compared to its downstream ones only, the l_is of the relevant logical buffers can be chosen to be large positive constants. Similarly l_is of the relevant logical buffers should be made small positive or even negative constants if a physical buffer is large as compared to its downstream ones to bring down the packet loss. Note that choice of c_i for a logical buffer B_i, affects the queue length of the corresponding physical buffer and its upstream and downstream ones, whereas that of l_i affects the queue length of the corresponding physical buffer and its downstream ones only.

- As explained above the scheduling policy spreads the congestion in the system. Any congestion downstream will be reflected in large queue lengths at the source node of the corresponding session. The queue lengths at the source node can serve as an useful indicator of the congestion state of the source destination paths used by the session. 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 fine grained quantization when the queue lengths fall below a certain threshold. The video quality obviously suffers when the quantization is coarse, but this produces graceful degradation in perceptual image quality during periods of congestion.

[8]A physical buffer receives messages from one or more physical buffers. These are its upstream physical buffers. If a node maintains different memory partitions for different trees passing through it as in Example IV.2, then the physical buffers are separate for diffrent trees and a physical buffer has only one upstream physical buffer.

[9]A physical buffer sends packets to some other physical buffers. These physical buffer are its downstream physical buffer.

This degradation is less than that produced by cell or packet loss[32]. Thus the queue lengths at the source node provide *implicit feedback* as to the congestion state of the network. Of course 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[32]. This solution of implicit feedback is inherently scalable. This implicit feedback may not be able to substitute the need for explicit feedback entirely, but may be used in conjunction, so that much more infrequent explicit feedback may serve the purpose.

- MMRS is completely adaptive, in the sense that its implementation is independent of the statistics of the arrival process. If the parameters c_i, g_i, C_{mn}s are selected to minimize delay, etc., then the knowledge of the statistics of the arrival process may help in making better choices.
- Routing and scheduling decisions can be taken at every slot, i.e., $\omega^n_{\iota+1} - \omega^n_\iota = 1$, $\Omega^e_{\iota+1} - \Omega^e_\iota = 1$ or at random or deterministic intervals. The exact choice must be made on a case by case basis, but we briefly discuss the points to consider while making such a choice. The advantages of taking scheduling decision at intervals are as follows. If the scheduling decision instants are the same for all the links, then taking scheduling decisions at intervals would help decentralized scheduling decisions. This would facilitate the communication of $E_i(\Omega_\iota)$s of the downstream buffers at $d(e)$ of a link e to the scheduler at the source, $o(e)$, before $\Omega_{\iota+1}$. Taking scheduling decision at intervals is also advantageous from computational complexity point of view if the relevant $g_i(\vec{b})$s for the link are computationally intensive functions of $\vec{b}$. This advantage is not there if $g_i(\vec{b})$s are constants or computationally simple functions. Besides reconfigurring the system every slot on account of rescheduling may not be possible for the system hardware. This observation applies for routing decisions also. By taking routing decisions at sufficiently long intervals, out of order delivery of packets of the same session can be substantially reduced, if necessary, without affecting the throughput. The penalty for taking routing and/or scheduling decision at intervals is possible increase in source to destination delay for the sessions.
- The size of the decision intervals may or may not depend on queue lengths. Former is the case when routing, scheduling decision intervals follow property (R3), (S3) respectively. Latter takes place when routing decision intervals follow property (R1) or (R2) and scheduling decision intervals follow property (S1) or (S2). Again the choice should be made on a case by case basis. The following are the points to consider. If (R3) or (S3) is followed (decision interval depending on queue lengths), then some queue length dependent computation must be performed every slot to determine whether routing or scheduling decision should be taken at the slot. If the intervals are deterministic or random but independent of queue lengths, ((R1), (R2), (S1), (S2)) then there is a computational overhead once per decision interval and this computation is for making the decision and not just for calculating the interval size. The advantage of (R3) over

(R1), (R2), or (S3) over (S1), (S2) is that some unnecessary switchings are avoided, and some necessary switchings 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. How undesirable can the system allow the current decision to be before a switch is made, can be precisely controlled through the parameters ϱ_n, $n = 1, \ldots, N$ and ς_i, $i = 1 \ldots M$.

- Taking scheduling decisions for the entire system at the same instants may facilitate decentralized scheduling, because this generally allows sufficient time for communication of $E_i(\Omega)$s. The synchronization of scheduling decision intervals is not difficult to achieve if the scheduling intervals satisfy property (S1). This can also be achieved by synchronizing the random number generators, if the scheduling intervals are generated randomly as per property (S2). However implementation would become difficult if the scheduling intervals are generated as per property (S3) and in order to attain uniform scheduling instants for the entire system, the entire system is rescheduled if any of the properties (S3a) to (S3c) are satisfied for any link. It is so because this would require propagation of a lot of information in a short time to all links in a large network. There is no particular advantage associated with taking routing decisions for all sessions simultaneously, and hence these decisions can be taken at the same or different instants.
- MMRS can be used for broadcasting. In this case the collection of multicast trees for the broadcast sessions may include all spanning trees rooted at the source node. In that case it may be infeasible to maintain simultaneously a buffer for each one of those trees in the source node. If that is so the minimum weight spanning tree has to be computed from time to time for each broadcast session and this tree should be used for routing the information. There exists efficient algorithms for computation of minimum weight spanning trees unlike that for minimum weight multicast trees which is an NP complete problem (Steiner problem). If the collection of trees for the broadcast session includes only a few spanning trees routed at the respective source nodes, then a logical buffer can be maintained for each one of those trees at the respective source nodes and our usual routing policy applies. Also in parallel with the broadcast session some sessions may multicast to some destination nodes. Usual routing policy applies to those sessions as well. 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). Let $X_u(t)$ be 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 $\{\vec{X}(td + \theta)\}_{t=0}^{\infty}$,

$\theta = 0, \ldots, Q-1$, and $\vec{X}(td+\theta)$ 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. The *phase* of the system should contain some information not contained in the queue lengths at the physical or the logical buffers. For example the phase of the system could be the residual times for the next routing, scheduling decisions for MMRS with routing, scheduling decision intervals satisfying R1 or R2 and S1 or S2 respectively. In that case the initial phase (phase at $t = 0$) indicates the residual times for the first routing, scheduling decisions, and given this information, (i,j) should be known uniquely. If the routing and scheduling decision intervals follow R3, S3, then the residual times for the next routing, scheduling decisions can be determined from the logical buffer queue lengths and hence should not represent the phase of the system. The system may not have any phase (as in the last case) and then $P = Q = 1$, i.e., $\{\vec{X}(t)\}_{t=0}^{\infty}$ should converge weakly to a random vector $\hat{X}$, with $E\hat{X} < \infty$. 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 that would 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. Next we relate the stability of the system to logical buffer queue lengths. For this purpose, we describe the possible relations between physical buffer and logical buffer queue lengths, $\vec{X}(t)$ and $\vec{B}(t)$ respectively, in the following discussion.

A node is a possibly multi-input multi-output multicast switch with the ability to serve packets to several outgoing links simultaneously. Figure 2 shows a node, Node 2 with one incoming link and three outgoing links. Refer to Example IV.1. 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 queueed 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 physical buffers are as those described in page 9. Note that here packets are replicated[10] (if at all) only when they are transmitted to the output. Upon arrival only a single copy of the packet is stored. Replication coincides with transmission to the outputs. This mode of replication is known as *replication-at-sending* (RAS)[4]. Let E_{Ti} be the set of outgoing links of tree T at node i, e.g., $E_{T_1 2} = \{L_2, L_3\}$, $E_{T_1 1} = \{L_1\}$, $E_{T_2 2} = \{L_2, L_3, L_4\}$ in Example IV.1, Figure 2. Let physical buffer u store packets of tree T of session n at node i. $X_u(t) = \max_{e \in E_{Ti}} B_{Tne}(t)$.

Example VI.1: Refer to Example IV.1, Figure 4. It shows the physical buffer storing tree T_2,

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

session 2 packets at node 2. It 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.2), where $j > l > m$. From Example IV.1 and $E_{T_2 2} = \{L_2, L_3, L_4\}$, $X_u(t) = \max(B_3(t), B_6(t), B_7(t)) = j+1$.

Packets travelling along different trees must occupy different memory locations. Thus whether packets of the different trees passing through the same node are stored in the same or different physical buffers make no essential difference in our case. Thus we could assume the existence of a physical buffer for each tree at each node without any loss in generality.

- If the packets are output queued, then every outgoing link has a physical buffer to store all packets transmitted to it from its origin node. 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 per 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[22]. $X_w(t) = \sum_{T\in\mathcal{V}} \max_{e\in E_{Tw}} B_{Tn_Te}(t)$, where $\mathcal{V} = \cup_{n=1}^{N} \mathcal{T}_n$, E_{Tw} is the set of outgoing links of tree T at node w, n_T is the session corresponding to tree T.

It follows from the relation between physical buffer and logical buffer queue lengths 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 $\{\vec{B}(td+\theta)\}_{t=0}^{\infty}$, $\theta = 0, \ldots, Q-1$, such that $\vec{B}(td+\theta)$ converges weakly to a random vector $\hat{B}^{(i,j+\theta\%Q)}$, $i \in \{0, \ldots, P-1\}$, $j \in \{0, \ldots, Q-1\}$, 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 in 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 $\mathcal{T}_n$.

For simplicity assume that each packet has a deterministic service time equal to 1 slot. 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 a_n can not be split in portions a_n^m such that the a_n^ms 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 1 unit. Note that the dependences between $A_{n_1}(t)$ and $A_{n_2}(t)$ have not been ruled out for any $n_1 \neq n_2$.

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

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

Note that a large class of g_is 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.

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

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

R represents the routing matrix.

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

Initially assume that the routing policy satisfies either (R1) or (R2) and the scheduling policy satisfies either (S1) or either (S2). We would discuss the case for other routing and scheduling policies towards the end of this section.

$$\Xi_e(t) = \Omega^e_{\iota+1} - t, \text{ where } \Omega^e_\iota \leq t < \Omega^e_{\iota+1}$$

$\Xi_e(t)$ is the residual time for the next scheduling decision at link e.

$$\xi_n(t) = \omega^n_{\iota+1} - t, \text{ where } \omega^n_\iota \leq t < \omega^n_{\iota+1}$$

$\xi_n(t)$ is the residual time for the next routing decision for the nth session. 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 below.

Theorem 1: There exists random vectors $\hat{B}^{(k,l)}$, $k = 0, 1, \ldots, P-1$, $l = 0, 1, \ldots, d-1$, $E\hat{B}^{(k,l)} < \infty$ such that given $(\vec{\Xi}(0), \vec{\xi}(0))$, $\{\vec{B}(td+i)\}_{t=0}^{\infty}$, $i = 0, 1, \ldots, d-1$, converges weakly to $\hat{B}^{(k,l+i\%d)}$, k, l uniquely known given $(\vec{\Xi}(0), \vec{\xi}(0))$, $k = 0, 1, \ldots, P-1$, $l = 0, 1, \ldots, d-1$.

We prove the above theorem towards the end of this section using Proposition 1 and Lemmas 1 and 2 stated below.

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

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

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 have been stated in a manner appropriate to our context. It follows from a theorem in [15] which we describe in Appendix B. Basically it states that if a "negative drift condition" (stated in Lemma 2) holds, then the system is stable.

Lemma 1: Given $(\vec{\Xi}(0), \vec{\xi}(0))$, $\{\vec{Y}(td+i)\}_{t=0}^{\infty}$, $i = 0, \ldots, d-1$, is a discrete time countable state aperiodic Markov chain with state space $\mathcal{X}_{(I(0), J(0)+i\%d)}$. $(I(0), J(0))$ is uniquely known given $(\vec{\Xi}(0), \vec{\xi}(0))$.

$\mathcal{X}_{(I(0),J(0)+i\%d)}$ has a single closed communication class for all $(I(0),J(0)),i$. This class is accessible from all states in $\mathcal{X}_{(I(0),J(0)+i\%d)}$.

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

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

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 A.

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 $\mathcal{C}_i$s are closed communication classes of periodicity d. 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 amongst 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 into periodicity classes $\mathcal{C}_{(i,0)}, \ldots, \mathcal{C}_{(i,d-1)}$, such that if $(\vec{\Xi}(t), \vec{\xi}(t)) \in \mathcal{C}_{(i,j)}$, $(\vec{\Xi}(t+1), \vec{\xi}(t+1)) \in \mathcal{C}_{(i,j+1\%d)}$, 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[11] from every state and has periodicity d for all i. The periodicity classes of $\mathcal{X}_i$ are $\mathcal{X}_{(i,0)}, \ldots, \mathcal{X}_{(i,d-1)}$, where $\vec{Y}(t) \in \mathcal{X}_{(i,j)}$ iff $(\vec{\Xi}(t), \vec{\xi}(t)) \in \mathcal{C}_{(i,j)}$. Let

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

Hence proved. $\square$

Proof of Theorem 1: It follows from Lemmas 1 and 2 and Proposition 1 that there exists random vectors $\{\hat{Y}^{(k,l)}\}$, $E\hat{Y}^{(k,l)} < \infty$, $k \in \{0, \ldots, P-1\}$, $l \in \{0, \ldots, d-1\}$ such that given $(\vec{\Xi}(0), \vec{\xi}(0))$, $\{\vec{Y}(td+i)\}_{t=0}^{\infty}$ converges in distribution to a random vector $\hat{Y}^{(I(0),J(0)+i\%d)}$, $E\phi_1(\hat{Y}^{(I(0),J(0)+i\%d)}) < \infty$. Since $\vec{Y}(t) = (\vec{B}(t), \vec{E}(t), \vec{\Gamma}(t), \vec{\Xi}(t), \vec{\xi}(t))$, given $(\vec{\Xi}(0), \vec{\xi}(0))$, $\{\vec{B}(td+i)\}_{t=0}^{\infty}$ converges in distribution to a random vector $\hat{B}^{(I(0),J(0)+i\%d)}$, $I(0) \in \{0, \ldots, P-1\}$, $J(0) \in \{0, \ldots, d-1\}$. We used function $\phi_1(\vec{y}) = \max(1, \frac{2\lambda' \sqrt{\sum_{i=1}^{M} c_i b_i^2}}{\kappa})$, $\kappa > 1$, $\lambda' > 0$ are constants. c_is are strictly positive constants. Thus $E\phi_1(\hat{Y}^{(I(0),J(0)+i\%d)}) < \infty$ implies that $E\hat{B}^{(I(0),J(0)+i\%d)} < \infty$. Hence the result follows. $\square$

[11] 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.

If the routing policy satisfies either (R1) or (R2) and the scheduling policy satisfies (S3), then the Markov chain $\vec{Y}(t) = (\vec{B}(t), \vec{E}(t), \vec{\Gamma}(t), \vec{\xi}(t))$ represents the system. If the routing policy satisfies (R3) and the scheduling policy satisfies either (S1) or (S2), then the Markov chain $\vec{Y}(t) = (\vec{B}(t), \vec{E}(t), \vec{\Gamma}(t), \vec{\Xi}(t))$ represents the system. Finally if the routing policy satisfies (R3) and the scheduling policy satisfies (S3), the Markov chain $\vec{Y}(t) = (\vec{B}(t), \vec{E}(t), \vec{\Gamma}(t))$ 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 are 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., $d = P = 1$. Thus Lemma 1 holds in this case as well. As we shall discuss in the appendix A that Lemma 2 holds in all these cases. Thus Theorem 1 holds in all these cases. Hence the proof of stability for feasible arrival rate vector generalizes. Also, the proof of stability for feasible arrival rate 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[23]. There we assumed that the arrival process for session n, $\{A_n(t)\}_{t=1}^{\infty}$ is a markov modulated process. 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. [28] adopts a similar approach 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 a_n can not be split in portions a_n^m such that the a_n^ms satisfy the capacity condition stated in Section II. All symbols introduced in this section have been listed in symbol table of page 46. 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. $\{(A_1(t), \ldots, A_N(t)\}_{t=1}^{\infty}$ can be a stationary ergodic process or a probabilistic function of a finite state irreducible aperiodic discrete time Markov Process.[12] As mentioned in

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

Section VI, we assume that each packet has a deterministic service time equal to 1 unit.

We shall use the following propositions later. The first is the well known Birkhoffs ergodic theorem and the second follows from a trivial extension of a similar result for positive recurrent discrete time Markov chains[20].

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

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

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 future $S(t)$s and $A(t)$s, then

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

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

We first introduce certain terminologies we use throughout. $\tilde{f} = (f_1, \dots, f_M)$ will be denoted as a *buffer discharge* vector. A *valid* buffer discharge vector is one in which $f_i \geq 0$, $\sum_{i \in S_e} f_i \leq 1$. S_e as defined before is the set of buffers sharing the same outgoing link e. Let F be the set of valid buffer discharge vectors. $\hat{a} = (a_1^1, \dots, a_1^{M_1}, a_2^1, \dots, a_2^{M_2}, \dots, a_N^1, \dots, a_N^{M_N})$ 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. *tree arrival rate vector* is just a nomenclature. If $\tilde{a} = (a_1, a_2, \dots, a_N)$ is the arrival rate vector,

$$A(\tilde{a}) = \{\hat{a} : \sum_{m=1}^{M_n} a_n^m = a_n, a_n^m \geq 0, m = 1, \dots M_n, n = 1, \dots N\}$$

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 $A(\tilde{a})$.

$$C = \{\hat{a} : \sum_{n=1}^{N}\sum_{m=1}^{M_n} a_n^m T_n^m \leq (1, \dots, 1)\}$$

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 can not 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 $f_n^m = \min_{i:m(i)=m,n(i)=n} f_i$, i.e., f_n^m is the minimum buffer discharge component amongst those corresponding to logical buffers belonging to the mth multicast

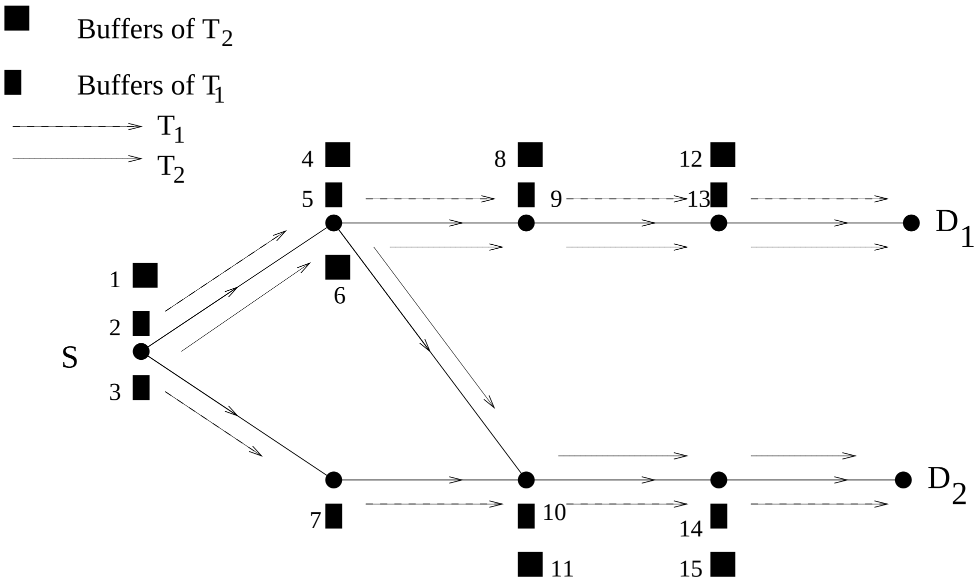

Fig. 5. A network with a single session, source S and destinations D_1, D_2.

tree of the nth session. Let $l_T = \arg\min_{i:m(i)=T} f_i$ (if there are more than one buffers attaining this minimum, the tie is broken arbitrarily in favor of any one of them). 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.

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

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 in page VII is the number of exogeneous packet arrivals at B_i at slot t.) We illustrate the concept of the buffer graph with the following example.

Example VIII.1: Figure 5 shows a multicast network with a single session, source S and destinations D_1, D_2. The session has two trees T_1 and T_2. Figure 5 shows the trees T_1 and T_2 and the corresponding logical buffers. 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. Figure 6 shows the corresponding buffer graph. Each buffer in Figure 5 corresponds to a vertex in the buffer graph shown in Figure 6. Buffers numbered 4 and 6 are destinations of buffer numbered 1 in the multicast network. Thus there are directed edges from vertex 1 to vertices 4, 6 in the buffer graph. The buffer graph consists of disconnected

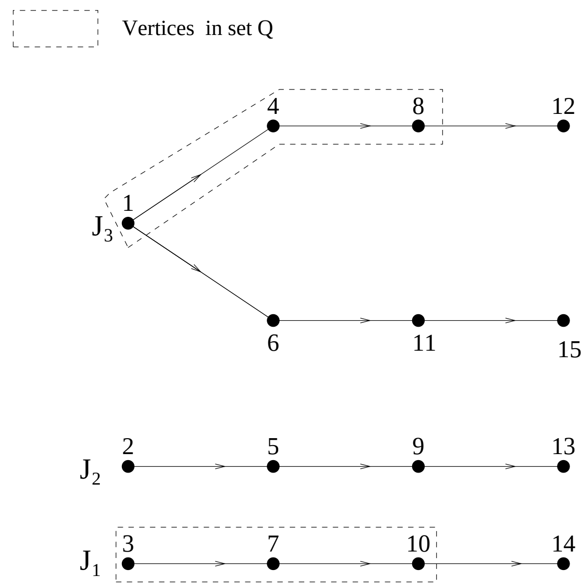

Fig. 6. Buffer graph of the multicast network shown in Figure 5. Trees J_1, J_2 correspond to tree T_1 and tree J_3 corresponds to tree T_2 of the multicast network. The set Q for a particular $\tilde{f}$ given in Example 8.1 has been shown.

trees. Trees J_1, J_2 correspond to tree T_1 and tree J_3 corresponds to tree T_2 of the multicast network. Let $f_8 = .3$, $f_{10} = .1$. $f_i = .4$, $i \notin \{8, 10\}$. Thus f_{10} is the minimum amongst the buffer discharge components of tree T_1 of the multicast network and f_8 is the minimum amongst the buffer discharge components of tree T_2. Vertex 3 is the root node of tree J_1 and vertex 10 is on tree J_1 corresponding to tree T_1 of the multicast network. Thus the vertices on the path from vertex 3 to vertex 10, i.e., vertices $3, 7, 10$ belong to Q. Similarly vertices $1, 4, 8$ on the path from vertex 1, the root node of tree J_3 containing vertex 8, to vertex 8 are in the set Q. Any exogeneous arrival in the session in the multicast network 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 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 every valid buffer discharge vector, $\tilde{f}$, satisfies the following property.

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

Z_l is the set of destinations of buffer B_l. $n(\tilde{f})$ is a particular session associated with $\tilde{f}$.

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

$$\begin{aligned} U_{\tilde{a}\hat{a}\tilde{f}} &= \max_{m,n}(a_n^m - f_n^m) \qquad \hat{a} \in A(\tilde{a}) \\ V_{\tilde{a}\tilde{f}} &= \min_{\hat{a} \in A(\tilde{a})} U_{\tilde{a}\hat{a}\tilde{f}} \end{aligned}$$

$V_{\tilde{a}\tilde{f}}$ is well defined as $A(\tilde{a})$ is a closed and bounded set 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 $U_{\tilde{a}\hat{a}\tilde{f}} \leq 0$, i.e., $a_n^m \leq f_n^m$, $\forall m, n$.

$$\begin{aligned} \sum_{n=1}^{N} \sum_{m=1}^{M_n} a_n^m T_n^m &\leq \sum_{n=1}^{N} \sum_{m=1}^{M_n} f_n^m T_n^m \\ &\leq (\ldots, \sum_{i \in S_e} f_i, \ldots) \qquad (9) \\ \text{Thus } \sum_{n=1}^{N} \sum_{m=1}^{M_n} a_n^m T_n^m &\leq (1, \ldots, 1) \end{aligned}$$

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^ms 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 $f_n^m \leq f_i$ 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 $\sum_{m=1}^{M_q} a_q^m - \sum_{m=1}^{M_q} f_q^m \leq 0$, $\forall q, \hat{a} \in A(\tilde{a})$. 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 $\max_n(a_n - \sum_{m=1}^{M_n} f_n^m) > 0$ for all $\tilde{f} \in F$. F is a closed and bounded set. $\max_n(a_n - \sum_{m=1}^{M_n} f_n^m)$ is a continuous function of $\tilde{f}$. Hence $\min_{\tilde{f} \in F} \max_n(a_n - \sum_{m=1}^{M_n} f_n^m)$ exists. Let $\epsilon = \min_{\tilde{f} \in F} \max_n(a_n - \sum_{m=1}^{M_n} f_n^m)$. $\epsilon > 0$ since $\max_n(a_n - \sum_{m=1}^{M_n} f_n^m) > 0$ for all $\tilde{f} \in F$. Thus $\max_n(a_n - \sum_{m=1}^{M_n} f_n^m) \geq \epsilon > 0$ for all $\tilde{f} \in F$. If $n(\tilde{f})$ 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 equation (7). □

Theorem 2: If 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 $W \subseteq \{1, \ldots, M\}$.

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

$$\begin{aligned} \sum_{i \in W} (R\vec{E}(t))_i &= \sum_{i \in W} ((|Z_i \cap W| - 1) E_i(t)) \\ &\geq - \sum_{i \in W, Z_i \cap W = \phi} E_i(t) \end{aligned} \tag{11}$$

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

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.)

$$\begin{aligned} \text{Let } \vec{\lambda}(t) &= \frac{1}{t} \sum_{\theta=1}^{t} \vec{E}(\theta) \\ \sum_{i \in S_e} \lambda_i(t) &= \frac{1}{t} \sum_{\theta=1}^{t} \sum_{i \in S_e} E_i(\theta) \\ &\leq 1 \text{ since } \sum_{i \in S_e} E_i(\theta) \leq 1 \ \forall \theta \\ \lambda_i(t) &\geq 0 \end{aligned}$$

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

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

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 $|\frac{1}{t}(\sum_{\theta=1}^{t} A_n(\theta)) - a_n| < \delta$ 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. $\square$

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 $\{1, \ldots, M\}$, $W_i \neq \phi$, $\forall i$. Thus

$$\begin{aligned} \sum_u X_u(t) &\geq \sum_j \frac{1}{|W_j|} (\sum_{i \in W_j} B_i(t)) \\ &\geq \frac{1}{\max_j |W_j|} \sum_{i=1}^{M} B_i(t) \end{aligned}$$

Thus it follows from Theorem 2 that if the arrival rate vector is unstable, $\sum_u X_u(t) \rightarrow_{t \rightarrow \infty} \infty$ a.s. Thus the system can not be stable if the arrival rate vector is unstable.

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. Significant amount of research have been directed towards the construction and the nature of the tree, e.g., whether the tree should be a shortest path tree or a core based tree 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 focussed on the selection in a dynamic manner of the appropriate tree for an incoming packet. This addresses the problem of load balancing. The scheduling in current multicast network is still best effort service. We have proposed a scheduling based on local information. The maximum throughputs attained by existing multicast routing protocols like DVMRP[5], CBT[1], PIM[6], MIP[18] 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. [16] proposes a throughput optimal algorithm for broadcasts in a mesh network. This happens to be a special case of ours. [30] 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.

Throughput optimal routing and scheduling policies are known for unicast networks[27], [26]. It proposes a routing and scheduling policy, called the parametric back pressure policy (PBP). Our scheduling policy is somewhat similar to the scheduling used there. But the intricacies of *traffic multiplication* in the multicast scenario can not be captured by the system model introduced in [27] 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 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 [26] nor [27] uses these scale factors and bias terms.

Appendix

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

We first introduce some terminologies. All terminologies introduced in appendix A have been listed in symbol table of page 47. H is the set of possible activation vectors, $\vec{E}$s. Note that $H = \{\vec{\gamma} : \gamma_i \in \{0,1\}, \sum_{i \in S_e} \gamma_i \leq 1\}.$, where $S_e = \{i : e(i) = e, 1 \leq i \leq M\}$ (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, 5 and 6. We state them below but prove them later.

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

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

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

Informally speaking Lemma 4 states that the cross 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 cross 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 $V(t)$. We will prove this lemma precisely later but the proof follows from the fact that, scheduling decisions are taken not too infrequently and when a link is scheduled, trees with large difference of source destination buffer backlogs is preferred over others (bias terms are also taken into account while making a decision but they are small compared to $V(t)$, for large $V(t)$).

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

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

Informally speaking 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. Intuitively this lemma holds because routing decisions are taken not too infrequently and when routing decision is taken for a session the trees with small queue lengths at source buffers are preferred over others (again scale factors and constant bias terms are taken into account).

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

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

Lemma 6 states that the relative difference between $V(t_1)$ and $V(t_2)$ becomes negligible as $V(t_1)$ increases if length of the interval $|t_2 - t_1|$ is bounded. We prove this lemma more precisely later, but 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.

Proof of Lemma 2: Let $\phi_2(\vec{y}) = \sum_{i=1}^{M} c_i B_i^2$, $\phi_2(\vec{Y}(t)) = V(t)$, where $V(t) = \sum_{i=1}^{M} c_i B_i^2(t)$

$$\text{Clearly } \phi_2(\vec{y}) < \infty, \forall \vec{y} \in \mathcal{X}, \tag{14}$$

where $\mathcal{X}$ is the state space of $\vec{Y}(t)$.

$$\begin{aligned}
V(t) &= (K\vec{B}(t))^T(K\vec{B}(t)) \\
K &= \text{diag}(\sqrt{c_1}, \ldots, \sqrt{c_N}) \\
V(t+1) - V(t) &= (K(\vec{B}(t+1) - \vec{B}(t)))^T(K(\vec{B}(t+1) + \vec{B}(t))) \\
&= (K(R\vec{E}(t+1) + \tilde{A}(t+1)))^T(K(2\vec{B}(t) + \\
&\quad R\vec{E}(t+1) + \tilde{A}(t+1))) \\
&= (R\vec{E}(t+1) + \tilde{A}(t+1))^T K^T K(R\vec{E}(t+1) + \\
&\quad \tilde{A}(t+1)) + 2(K^T K\vec{B}(t))^T(R\vec{E}(t+1) + \tilde{A}(t+1)) \\
E(V(t+1) - V(t)/\vec{Y}(t)) &= E((R\vec{E}(t+1) + \tilde{A}(t+1))^T K^T K(R\vec{E}(t+1) + \\
&\quad \tilde{A}(t+1))/\vec{Y}(t)) + 2E((K^T K\vec{B}(t))^T(R\vec{E}(t+1) + \\
&\quad \tilde{A}(t+1))/\vec{Y}(t))
\end{aligned} \tag{15}$$

$$E((R\vec{E}(t+1)+\tilde{A}(t+1))^T K^T K(R\vec{E}(t+1)+\tilde{A}(t+1))/\vec{Y}(t)) \leq \alpha\ \forall \vec{Y}(t) \tag{16}$$

(α is a finite positive constant)

$$\begin{aligned} E((K^T K\vec{B}(t))^T(R\vec{E}(t+1)+\tilde{A}(t+1))/\vec{Y}(t)) &= (K^T K\vec{B}(t))^T(R\vec{E}(t+1)+ \\ &\quad E(\tilde{A}(t+1)/\vec{Y}(t))) \end{aligned} \tag{17}$$

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

$$E(\tilde{A}_i(t+1)/\vec{Y}(t)) = \begin{matrix} a_n & \text{if } B_i \in O_{T_n(t+1)n(i)} \\ 0 & \text{otherwise.} \end{matrix} \tag{18}$$

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

$$\begin{aligned} (K^T K\vec{B}(t))^T R &= -\vec{D}(t+1) \\ -\vec{D}(t+1)^T\vec{E}(t+1) &\leq -\max_{\vec{\gamma}\in H}\vec{D}(t+1)^T\vec{\gamma}+\psi(t) \text{ from Lemma 4} \\ (K^T K\vec{B}(t))^T R\vec{E}(t+1) &\leq -\max_{\vec{\gamma}\in H} D(t+1)^T\vec{\gamma}+\psi(t) \end{aligned} \tag{19}$$

$$R\vec{f} = -\hat{a} \tag{20}$$

$$\text{where } f_i = a_{n(i)}^{m(i)} \tag{21}$$

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

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

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, $\sum_{i\in S_e} f_i = \sum_{n=1}^{N}\sum_{m=1}^{M_n} a_n^m(T_n^m(e)) < 1$, (equation (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 $\sum_{i\in S_e}\gamma_i \leq 1$ and $\gamma_i \in \{0,1\}$. This means that $\vec{f} = \sum_{\vec{\gamma}\in H}\lambda_{\vec{\gamma}}\vec{\gamma}$, $\sum_{\vec{\gamma}\in H}\lambda_{\vec{\gamma}} < 1$, $\lambda_{\vec{\gamma}} \geq 0$, if the arrival rate vector is feasible.

$$\hat{a} = -\sum_{\vec{\gamma}\in H}\lambda_{\vec{\gamma}} R\vec{\gamma} \tag{22}$$

$$(K^T K\vec{B}(t))^T E(\tilde{A}(t+1)/\vec{Y}(t)) = \sum_{n=1}^{N} a_n \sum_{B_i\in O_{T_n(t+1)n}} c_i B_i(t)$$

$$
\begin{aligned}
&\leq \sum_{n=1}^{N} \sum_{B_i \in O_n} f_i c_i B_i(t) + \varepsilon, \\
&\quad O_n = \cup_{1 \leq l \leq M_n} O_{ln}, \varepsilon = \sum_{i=1}^{n} a_n \varepsilon_n \qquad (23) \\
&\quad \text{(Using Lemma 5 and (21))}
\end{aligned}
$$

$$
\begin{aligned}
\sum_{n=1}^{N} \sum_{B_i \in O_n} f_i c_i B_i(t) &= (K^T K \vec{B}(t))^T \hat{a} \\
&= \sum_{\vec{\gamma} \in H} \lambda_{\vec{\gamma}} (-(K^T K \vec{B}(t))^T R) \vec{\gamma} \text{ (from (22))} \\
&= \sum_{\vec{\gamma} \in H} \lambda_{\vec{\gamma}} \vec{D}^T(t+1) \vec{\gamma} \\
&\leq \lambda \max_{\vec{\gamma} \in H} \vec{D}^T(t+1) \vec{\gamma} \qquad (24) \\
\lambda &= \sum_{\vec{\gamma} \in H} \lambda_{\vec{\gamma}} \\
\lambda &< 1
\end{aligned}
$$

From equation (17), inequalities (19), (23) and (24), we have

$$
E((K^T K \vec{B}(t))^T (R\vec{E}(t+1) + \tilde{A}(t+1))/\vec{Y}(t)) \leq -(1-\lambda) \max_{\vec{\gamma} \in H} \vec{D}^T(t+1)\vec{\gamma} + \psi(t) + \varepsilon \qquad (25)
$$

$$
\text{Let } m = \arg \max_{1 \leq i \leq M} \sqrt{c_i} B_i(t)
$$

$$
\begin{aligned}
\sqrt{c_m} B_m(t) &\geq \sqrt{\frac{V(t)}{M}} \\
c_m B_m(t) &\geq \sqrt{c_m} \sqrt{\frac{V(t)}{M}} \\
&\geq (\min_{1 \leq i \leq M} \sqrt{c_i}) \sqrt{\frac{V(t)}{M}} \qquad (26)
\end{aligned}
$$

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. B_m is the first element. 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_js in the set $s(B_m)$ and so on. This sequence is finite and would end in the B_js for which $s(B_j) = Z_j = \phi$. Let $Q_i(t) = c_i B_i(t) - \sum_{B_j \in Z_j} c_j B_j(t)$. Observe that $c_m B_m(t) = \sum_{i:B_i \in \text{ sequence}} Q_i(t)$. Let the sequence have X terms. $X \leq M$. Thus

$$
\begin{aligned}
c_m B_m(t) &\leq X \max_{1 \leq i \leq X} Q_i(t) \\
&\leq M \max_{1 \leq i \leq M} \left(c_i B_i(t) - \sum_{B_k \in Z_i} c_k B_k(t) \right) \\
&\leq M \max_{\vec{\gamma} \in H} \vec{D}^T(t+1) \vec{\gamma} \qquad (27)
\end{aligned}
$$

$$\max_{\vec{\gamma}\in H} \vec{D}^T(t+1)\vec{\gamma} \geq (\min_{1\leq i\leq M} \sqrt{c_i})\sqrt{\frac{V(t)}{M^3}} \text{ from equations (26) and (27)} \tag{28}$$

From Lemma 4

$$\left.\begin{array}{ll} \psi(t) \leq \lambda'\sqrt{V(t)} & \text{if } V(t) \geq L(\lambda'), \\ \lambda' = \frac{1}{2}\frac{(1-\lambda)\min_{1\leq i\leq M}\sqrt{c_i}}{M^{3/2}}, & L(\lambda') \text{ is a constant depending upon } \lambda'. \end{array}\right\} \tag{29}$$

Since $\lambda < 1$, (29) follows from Lemma 4. From inequalities (25), (28) and (29)

$$E((K^TK\vec{B}(t))^T(R\vec{E}(t+1)+\tilde{A}(t+1))/\vec{Y}(t)) \leq -\lambda'\sqrt{V(t)}+\varepsilon \text{ if } V(t) \geq L(\lambda'). \tag{30}$$

Using equations (15), (16) and (30)

$$E(V(t+1)-V(t)/\vec{Y}(t)) \leq \alpha + 2\varepsilon - 2\lambda'\sqrt{V(t)} \text{ for all sufficiently large } V(t) \tag{31}$$

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

where λ' is the same as that in equation (29).

$$\phi_1(\vec{Y}(t)) \geq 1,\ \forall \vec{Y}(t) \in \mathcal{X} \tag{32}$$

$$\begin{aligned} t_k &= (t+1)d+i-k-1 \\ (t+1)d+i &= t_0+1 \\ td+i &= t_{d-1} \\ E(V(t_0+1)-V(t_{d-1})/\vec{Y}(t_{d-1})) &= E(V(t_{d-1}+1)-V(t_{d-1})/\vec{Y}(t_{d-1})) \\ &\quad + \sum_{k=0}^{d-2} E(V(t_k+1)-V(t_k)/\vec{Y}(t_{d-1})) \\ E(V(t_k+1)-V(t_k)/\vec{Y}(t_{d-1})) &= \sum_{\tilde{y}\in\mathcal{X}} E(V(t_k+1)-V(t_k)/\vec{Y}(t_k)=\tilde{y}) \\ &\qquad Pr(\vec{Y}(t_k)=\tilde{y}/\vec{Y}(t_{d-1})) \qquad (33) \\ &\text{since } \vec{Y}(t) \text{ is Markovian and } t_{d-1} \leq t_k \end{aligned}$$

$E(V(t_k+1)-V(t_k)/\vec{Y}(t_k)) < 0$, if $V(t_k) > \max(L(\lambda'), \frac{\alpha+2\varepsilon}{2\lambda'})$ (equation 31). Since $|t_k - t_{d-1}| < d$ (bounded), it follows from Lemma 6 that the above holds w.p. 1 if $V(t_{d-1})$ is sufficiently large. Thus every term in the summation in equation (33) is nonpositive if $V(t_{d-1})$ is sufficiently large. Now it follows from equation (31) that

$$\text{Thus } E(V(t_0+1)-V(t_{d-1})/\vec{Y}(t_{d-1})) \leq \alpha + 2\varepsilon - 2\lambda'\sqrt{V(t_{d-1})} \tag{34}$$

for all sufficiently large $V(t_{d-1})$

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

$$\begin{aligned}
E(\phi_2(\vec{Y}((t+1)d+i))/\vec{Y}(td+i)) &\leq \phi_2(\vec{Y}(td+i)) - \phi_1(\vec{Y}(td+i)), \forall \vec{Y}(td+i) \in A^c, \\
& A = \{\vec{y} : \phi_2(\vec{y}) \leq \beta, \vec{y} \in \mathcal{X}\}
\end{aligned}$$

A is a finite set because $\{\vec{B}(td+i) : V(td+i) \leq \mu\}$ is a finite set for all $\mu \in R$ and $\vec{E}(td+i)$, $\vec{\Gamma}(td+i)$, $\vec{\Xi}(td+i)$, $\vec{\xi}(td+i)$ take values in finite sets only.

$$\text{Thus } E(\phi_2(\vec{Y}((t+1)d+i))/\vec{Y}(td+i) = \vec{y}) \leq \phi_2(\vec{y}) - \phi_1(\vec{y}), \forall \vec{y} \in A^c, |A| < \infty \tag{35}$$

Equations (14), (32), (35) 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. □

We proved that Lemma 2 holds when the routing policy satisfies either (R1) or (R2) and the scheduling policy satisfies either (S1) or (S2). We shall prove later that Lemmas 4, 5 and 6 hold for other cases as well, with the Markov chain $\vec{Y}(t)$ suitably defined in each case as discussed before. Thus Lemma 2 holds in all these cases.

A. *Proof of Lemma 6*

Proof of Lemma 6:

$$|B_i(t+T) - B_i(t)| \leq \begin{cases} T & B_i \notin O_{n(i)} \\ K_{n(i)}T & B_i \in O_{n(i)} \end{cases} \quad (O_n \text{ defined in (23)}) \tag{36}$$

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 $K_{n(i)}$ arrival 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. It follows that

$$\forall i\ |B_i(t+T) - B_i(t)| \leq \sigma T \tag{37}$$

$$|D_i(t+T) - D_i(t)| \leq \Upsilon\sigma T, \tag{38}$$

$$\text{where } \Upsilon = \sum_{i=1}^{M} c_i, \sigma = \max_{1 \leq n \leq N} K_n \tag{39}$$

$$\begin{aligned}\frac{|V(t_1)-V(t_2)|}{V(t_1)} &\leq \frac{2\sum_{i=1}^{M} c_i B_i(t_1)\sigma|t_2-t_1| + M\Upsilon\sigma^2(t_2-t_1)^2}{\sum_{i=1}^{M} c_i B_i(t_1)^2} \\ &\leq \delta \ \forall t_1 \text{ s.t. } V(t_1) \geq L(\Lambda,\delta), \text{ since } |t_2-t_1| \leq \Lambda\end{aligned} \tag{40}$$

Thus for all $\delta > 0$, there exists $L(\Lambda, \delta)$ such that

$$(1-\delta)V(t_1) \leq V(t_2) \leq (1+\delta)V(t_1) \ \forall t \text{ s.t. } V(t_1) \geq L(\Lambda,\delta)$$

(40) follows from (37) □

The proof nowhere makes any assumption about which property the routing and scheduling intervals satisfy.

B. Proof of Lemma 4

Proof of Lemma 4: Initially we do not make any assumption about which property the routing and scheduling intervals satisfy. Let

$$\begin{aligned} W_1 &= \{e : E_i(t+1) = 0, \ \forall i \in S_e\} \\ W_2 &= \{e : D_i(t+1) \leq 0, \ \forall i \in S_e\} \\ W_3 &= \{e : j_e(t+1) \neq k_e(t+1)\} \\ \text{where } j_e(t+1) &= \arg\max_{i \in S_e} E_i(t+1) \\ \text{and } k_e(t+1) &= \arg\max_{i \in S_e} D_i(t+1) \end{aligned}$$

$$\begin{aligned}\vec{D}(t+1)^T\vec{E}(t+1) &= \max_{\vec{\gamma}\in H} \vec{D}(t+1)^T\vec{\gamma} - \sum_{e\in W_1\cap W_2^c} D_{k_e(t+1)}(t+1) + \\ &\sum_{e\in W_1^c\cap W_2^c\cap W_3} \left(D_{j_e(t+1)}(t+1) - D_{k_e(t+1)}(t+1)\right) + \\ &\sum_{e\in W_1^c\cap W_2} D_{j_e(t+1)}(t+1)\end{aligned} \tag{41}$$

Now let the scheduling intervals follow property (S1) or (S2). Let $\nu^e(t) = \arg\max_{\Omega_\iota^e \leq t} \Omega_\iota^e$. $\nu_1^e(t) = \nu^e(t+1) - 1$. Let $e \in W_1^c \cap W_2$.

$$\begin{aligned} \vec{E}_{j_e(t+1)}(\nu^e(t+1)) &= 1 \\ D_{j_e(t+1)}(\nu^e(t+1)) + l_{j_e(t+1)}(\nu^e(t+1)) &> 0 \\ D_{j_e(t+1)}(\nu^e(t+1)) &\geq -l_{j_e(t+1)}(\nu^e(t+1)) \\ l_{j_e(t+1)}(\nu^e(t+1)) &= g_{j_e(t+1)}(\vec{B}(\nu_1^e(t))) \end{aligned}$$

Thus from (38) and since $t+1-\nu^e(t+1) \leq T_s$ from (S1), (S2)

$$\begin{aligned}\forall e \in W_1^c \cap W_2, \ D_{j_e(t+1)}(t+1) &\geq -\Upsilon\sigma T_s - g_{j_e(t+1)}(\vec{B}(\nu_1^e(t))) \\ &\geq -\Upsilon\sigma T_s - \max_{\substack{1\leq i\leq M \\ e\in E}} |g_i(\vec{B}(\nu_1^e(t)))|\end{aligned} \tag{42}$$

Let $e \in W_1^c \cap W_2^c \cap W_3$.

$$\begin{aligned} D_{j_e(t+1)}(t+1) - D_{k_e(t+1)}(t+1) &\geq D_{j_e(t+1)}(\nu^e(t+1)) - D_{k_e(t+1)}(\nu^e(t+1)) - 2\Upsilon\sigma T_s \\ D_{j_e(t+1)}(\nu^e(t+1)) + l_{j_e(t+1)}(\nu^e(t+1)) &> 0 \end{aligned}$$

Let $k_e(t+1) \notin P_e(\nu^e(t+1))$, i.e., $B_{k_e(t+1)}(\nu^e(t+1)-1) = 0$

$$\begin{aligned} D_{k_e(t+1)}(\nu^e(t+1)) &= -\sum_{m \in Z_{k_e(t+1)}} c_m B_m(\nu^e(t+1)-1) \\ D_{j_e(t+1)}(\nu^e(t+1)) - D_{k_e(t+1)}(\nu^e(t+1)) &\geq D_{j_e(t+1)}(\nu^e(t+1)) \\ &> -g_{j_e(t+1)}(\vec{B}(\nu_1^e(t))) \end{aligned}$$

If $k_e(t+1) \in P_e(\nu^e(t+1))$,

$$\begin{aligned} D_{j_e(t+1)}(\nu^e(t+1)) + l_{j_e(t+1)}(\nu^e(t+1)) &\geq D_{k_e(t+1)}(\nu^e(t+1)) + l_{k_e(t+1)}(\nu^e(t+1)) \\ D_{j_e(t+1)}(\nu^e(t+1) - D_{k_e(t+1)}(\nu^e(t+1)) &\geq g_{k_e(t+1)}(\vec{B}(\nu_1^e(t)) - g_{j_e(t+1)}(\vec{B}(\nu_1^e(t))) \end{aligned}$$

$$\begin{aligned} &\forall e \in W_1^c \cap W_2^c \cap W_3, \\ D_{j_e(t+1)}(t+1) - D_{k_e(t+1)}(t+1) &\geq -2\Upsilon\sigma T_s + \min\Big(-g_{j_e(t+1)}(\vec{B}(\nu_1^e(t))), \\ &\quad -g_{j_e(t+1)}(\vec{B}(\nu_1^e(t))) + g_{k_e(t+1)}(\vec{B}(\nu_1^e(t)))\Big) \\ &\geq -2\Upsilon\sigma T_s - 2\max_{\substack{1\leq i\leq M \\ e\in E}} |g_i(\vec{B}(\nu_1^e(t)))| \end{aligned} \tag{43}$$

Let $e \in W_1 \cap W_2^c$. Consider the case when $E_{j_e(\nu^e(t+1))}(\nu^e(t+1)) = 0$.

$$D_{k_e(t+1)}(\nu^e(t+1)) \leq \begin{cases} -g_{k_e(t+1)}(\vec{B}(\nu_1^e(t))) & k_e(t+1) \in P_e(\nu^e(t+1)) \\ 0 & \text{otherwise.} \end{cases}$$

Now consider the case when $E_{j_e(\nu^e(t+1))}(\nu^e(t+1)) = 1$. $B_{j_e(\nu^e(t+1))}(t') = 0$, for some $\nu^e(t+1) - 1 \leq t' \leq t$ Thus $B_{j_e(\nu^e(t+1))}(\nu^e(t+1)-1) \leq t' - \nu^e(t+1) + 1 \leq T_s$ (since at most 1 packet can be served from $B_{j_e(\nu^e(t+1))}$ in a slot). Thus $D_{j_e(\nu^e(t+1))}(\nu^e(t+1)) \leq c_{j_e(\nu^e(t+1))}T_s \leq \Upsilon T_s$.
Let $k_e(t+1) \in P_e(\nu^e(t+1))$.

$$\begin{aligned} D_{k_e(t+1)}(\nu^e(t+1)) + l_{k_e(t+1)}(\nu^e(t+1)) &\leq D_{j_e(\nu^e(t+1))}(\nu^e(t+1)) + l_{j_e(\nu^e(t+1))}(\nu^e(t+1)) \\ D_{k_e(t+1)}(\nu^e(t+1)) &\leq \Upsilon T_s + l_{j_e(\nu^e(t+1))}(\nu^e(t+1)) - l_{k_e(t+1)}(\nu^e(t+1)) \end{aligned}$$

If $k_e(t+1) \notin P_e(\nu^e(t+1))$, $D_{k_e(t+1)}(\nu^e(t+1)) \leq 0$. Thus $D_{k_e(t+1)}(\nu^e(t+1)) \leq \max(0, -g_{k_e(t+1)}(\vec{B}(\nu_1^e(t))), \Upsilon T_s + g_{j_e(\nu^e(t+1))}(\vec{B}(\nu_1^e(t))) - g_{k_e(t+1)}(\vec{B}(\nu_1^e(t)))$. From (38) and since $t+1-\nu^e(t+1) \leq T_s$ from (S1), (S2)

$$\forall e \in W_1 \cap W_2^c,\ D_{k_e(t+1)}(t+1) \leq \Upsilon\sigma T_s + \max\Big(0, -g_{k_e(t+1)}(\vec{B}(\nu_1^e(t))),$$

$$\begin{aligned}
&\Upsilon T_s + g_{j_e(\nu^e(t+1))}(\vec{B}(\nu_1^e(t))) - \\
&g_{k_e(t+1)}(\vec{B}(\nu_1^e(t)))\Big) \\
\leq\; &\Upsilon(\sigma+1)T_s + 2\max_{\substack{1\leq i\leq M\\ e\in E}} |g_i(\vec{B}(\nu_1^e(t)))| \qquad (44)
\end{aligned}$$

From (41), (42), (43), (44)

$$\left.\begin{aligned}
\vec{D}(t+1)^T\vec{E}(t+1) &\geq \max\nolimits_{\vec{\gamma}\in H}\vec{D}(t+1)^T\vec{\gamma} - \zeta_1 - \\
&\quad 5|E|\max\nolimits_{\substack{1\leq i\leq M\\ e\in E}} |g_i(\vec{B}(\nu_1^e(t)))| \\
\zeta_1 &= 4|E|\Upsilon(\sigma+\tfrac{1}{4})T_s
\end{aligned}\right\} \qquad (45)$$

$$\begin{aligned}
\psi(t) &= \zeta_1 + 5|E|\max_{\substack{1\leq i\leq M\\ e\in E}} |g_i(B(\vec{\nu_1^e}(t)))| \qquad (46)\\
(\hat{i}(t),\hat{e}(t)) &= \arg\max_{\substack{1\leq i\leq M\\ e\in E}} |g_i(B(\vec{\nu_1^e}(t)))|
\end{aligned}$$

From (5), for any $\delta_1 > 0$ there exists $L_1(\delta_1)$, such that

$$\frac{\psi(t)}{\sqrt{V(\nu_1^{\hat{e}(t)}(t))}} \leq \delta_1 \ \forall t \text{ s.t. } V(\nu_1^{\hat{e}(t)}(t)) \geq L_1(\delta_1) \qquad (47)$$

Let $\delta_2 > 0$. From Lemma 6 that there exists $L_2(T_s,\delta_2)$ such that

$$(1-\delta_2)V(t) \leq V(\nu_1^{\hat{e}(t)}(t)) \leq (1+\delta_2)V(t) \ \forall t \text{ s.t. } V(t) \geq L_2(T_s,\delta_2) \qquad (48)$$

since $|t-\nu_1^{\hat{e}(t)}(t)| \leq T_s$. Given any δ', choose δ_1,δ_2, such that $\delta_1\sqrt{1+\delta_2} \leq \delta'$. It follows from (47) and (48) that $\psi(t) \leq \delta'\sqrt{V(t)}$, for all t such that $V(t)$ is sufficiently large ($V(t) \geq \max(L_1(\delta_1)/(1-\delta_2), L_2(T_s,\delta_2))$, where δ_1,δ_2 have been chosen to satisfy $\delta_1\sqrt{1+\delta_2} \leq \delta'$) and $\vec{D}(t+1)^T\vec{E}(t+1) \geq \max_{\vec{\gamma}\in H}\vec{D}(t+1)^T\vec{\gamma} - \psi(t)$ from (45) and (46).

Now let the scheduling intervals follow property (S3).

$$\begin{aligned}
\forall e \in W_1\cap W_2^c,\ D_{k_e(t+1)}(t+1) \leq\; &\max\Big(-g_{k_e(t+1)}(\vec{B}(t)), \\
&-g_{k_e(t+1)}(\vec{B}(t)) + g_{j_e(t)}(\vec{B}(t)) + \\
&\varsigma_{e1}, -g_{k_e(t+1)}(\vec{B}(t)) + \varsigma_{e3}\Big) \qquad (49)\\
\leq\; &\max(\varsigma_{e1},\varsigma_{e3}) + 2\max_{1\leq i\leq M}|g_i(\vec{B}(t))| \qquad (50)
\end{aligned}$$

$$\begin{aligned}
&\forall e \in W_1^c\cap W_2^c\cap W_3, \\
D_{j_e(t+1)}(t+1) - D_{k_e(t+1)}(t+1) \geq\; &g_{k_e(t+1)}(\vec{B}(t)) - g_{j_e(t+1)}(\vec{B}(t)) \\
&-\varsigma_{e1} \qquad (51)\\
\geq\; &-2\max_{1\leq i\leq M}|g_i(\vec{B}(t))| - \varsigma_{e1} \qquad (52)
\end{aligned}$$

$$\begin{aligned}\forall e \in W_1^c \cap W_2,\ D_{j_e(t+1)}(t+1) &> -g_{j_e(t+1)}(\vec{B}(t)) - \varsigma_{e2} && (53)\\ &\geq -\max_{1\leq i\leq M}|g_i(\vec{B}(t))| - \varsigma_{e2} && (54)\end{aligned}$$

(49) can be shown as follows. $D_{k_e(t+1)}(t+1) \leq 0$ if $k_e(t+1) \notin P_e(t+1)$, but since $e \in W_2^c$, $D_{k_e(t+1)} > 0$. Hence $k_e(t+1) \in P_e(t+1)$. If $t+1 \in \{\Omega_\iota^e\}$, then $D_{k_e(t+1)}(t+1) + l_{k_e(t+1)}(t+1) \leq 0$, since the link idles. If $t+1 \notin \{\Omega_\iota^e\}$, then consider two cases: $E_{j_e(t)}(t) = 1$ and $E_{j_e(t)}(t) = 0$. Let $E_{j_e(t)}(t) = 1$. Now $E_{j_e(t+1)}(t+1) = 0$ can occur only because $B_{j_e(t)}(t) = 0$. Thus we have

$$\begin{aligned}D_{j_e(t)}(t+1) &\leq 0\\ D_i(t+1) + l_i(t+1) &< D_{j_e(t)}(t+1) + l_{j_e(t)}(t+1) + \varsigma_{e1},\\ &\quad \forall i \in P_e(t+1) \text{ from (S3a).}\\ D_{k_e(t+1)}(t+1) + l_{k_e(t+1)}(t+1) &< l_{j_e(t)}(t+1) + \varsigma_{e1}\end{aligned}$$

Now let $E_{j_e(t)}(t) = 0$. From (S3c) $D_i(t+1) + l_i(t+1) < \varsigma_{e3}, \forall i \in P_e(t+1)$. Thus

$$D_{k_e(t+1)}(t+1) + l_{k_e(t+1)}(t+1) \leq \varsigma_{e3}$$

$$\begin{aligned}\text{Thus } D_{k_e(t+1)}(t+1) &\leq \max(-l_{k_e(t+1)}(t+1), -l_{k_e(t+1)}(t+1) +\\ &\quad l_{j_e(t)}(t+1) + \varsigma_{e1}, -l_{k_e(t+1)}(t+1) + \varsigma_{e3})\end{aligned}$$

Hence (49) follows.

(51) can be justified as follows. If $k_e(t+1) \notin P_e(t+1)$, $D_{k_e(t+1)}(t+1) \leq 0$, but since $e \in W_2^c$, $D_{k_e(t+1)}(t+1) > 0$. Thus $k_e(t+1) \in P_e(t+1)$. If $t+1 \in \{\Omega_\iota^e\}$, from the scheduling mechanism, $D_{j_e(t+1)}(t+1) - D_{k_e(t+1)}(t+1) \geq l_{k_e(t+1)}(t+1) - l_{j_e(t+1)}(t+1)$. If $t+1 \notin \{\Omega_\iota^e\}$, since $E_{j_e(t+1)}(t+1) = 1$, $j_e(t+1) = j_e(t)$. Using this it follows from (S3a) that $D_{j_e(t+1)}(t+1) - D_{k_e(t+1)}(t+1) \geq l_{k_e(t+1)}(t+1) - l_{j_e(t+1)}(t+1) - \varsigma_{e1}$.

Let $t+1 \in \{\Omega_\iota^e\}$. $E_{j_e(t+1)}(t+1) = 1$ implies that $D_{j_e(t+1)}(t+1) > -l_{j_e(t+1)}(t+1)$. Now let $t+1 \notin \{\Omega_\iota^e\}$. Again since $E_{j_e(t+1)}(t+1) = 1$, $j_e(t+1) = j_e(t)$. From (S3b) and the fact that $j_e(t+1) = j_e(t)$, $D_{j_e(t+1)}(t+1) > -l_{j_e(t+1)}(t+1) - \varsigma_{e2}$. Hence (53) follows.

From (41), (50), (52), (54)

$$\left.\begin{aligned}\vec{D}(t+1)^T\vec{E}(t+1) &\geq \max\nolimits_{\vec{\gamma}\in H} \vec{D}(t+1)^T\vec{\gamma} - \zeta_2 - 5|E|\max\nolimits_{1\leq i\leq M}|g_i(\vec{B}(t))|\\ \text{where } \zeta_2 &= \textstyle\sum_{e\in E}\max(\varsigma_{e1}, \varsigma_{e2}, \varsigma_{e3})\end{aligned}\right\} \quad (55)$$

$$\begin{aligned}\text{With } \psi(t) &= \zeta_2 + 5|E|\max_{1\leq i\leq M}|g_i(\vec{B}(t))|\\ \vec{D}(t+1)^T\vec{E}(t+1) &\geq \max_{\vec{\gamma}\in H}\vec{D}(t+1)^T\vec{\gamma} - \psi(t)\end{aligned}$$

It follows from (5) that given any $\delta' > 0$, there exists $L(\delta')$ such that $\psi(t) \leq \delta'\sqrt{V(t)}$ if $V(t) \geq L(\delta')$.

□

C. *Proof of Lemma 5*

Proof of Lemma 5: If $t+1 \in \{\omega_\iota^n\}$, from the routing policy,

$$\begin{aligned}\sum_{B_i \in O_{T_n(t+1)n}} c_i B_i(t) &\leq (\sum_{B_i \in O_{mn}} c_i B_i(t)) + C_{mn} - C_{T_n(t+1)n} \ \forall m \in \{1, \ldots M_n\} \\ &\leq (\sum_{B_i \in O_{mn}} c_i B_i(t)) + \upsilon_n \ \forall m, 1 \leq m \leq M_n \qquad (56)\\ &\text{where } \upsilon_n = \max_{m_1, m_2 \in \{1, \ldots M_n\}} (C_{m_1 n} - C_{m_2 n}) \qquad (57)\end{aligned}$$

Let $t+1 \notin \{\omega_\iota^n\}$.

First let the routing decision intervals satisfy property (R1) or (R2).

$$\begin{aligned}\sum_{B_i \in O_{T_n(t+1)n}} c_i B_i(t) &\leq \sum_{B_i \in O_{T_n(t+1)n}} c_i B_i(\nu_1(t)) + \Upsilon K_n T_r \\ &\quad (\text{ from } (36), (39) \text{ and } (R1), (R2)) \\ &\leq \sum_{B_i \in O_{\Gamma_m n}} c_i B_i(\nu_1(t)) + \upsilon_n + \Upsilon K_n T_r \\ &\quad (O_{T_n(t+1)n} = O_{T_n(\nu_1(t)+1)}, \nu_1(t)+1 \in \{\omega_\iota^n\} \text{and } (56)). \\ &\leq \sum_{B_i \in O_{\Gamma_m n}} c_i B_i(t) + \upsilon_n + 2\Upsilon K_n T_r \qquad (58)\\ &\quad (\text{ from } (36), (39) \text{ and } (R1), (R2))\end{aligned}$$

Thus if the routing decision intervals satisfy property (R1) or (R2) the result follows from (56) and (58) with $\varepsilon_n = \upsilon_n + 2\Upsilon K_n T_r$.

Now let the routing decision intervals satisfy property (R3). For $t+1 \notin \{\omega_\iota^n\}$, $T_n(t) = T_n(t+1)$ and hence from (R3)

$$\sum_{B_i \in O_{T_n(t+1)n}} c_i B_i(t) \leq \sum_{B_i \in O_{mn}} c_i B_i(t) + \varrho_n + C_{mn} - C_{T_n(t+1)n} \qquad (59)$$

The result follows from (57), (56) and (59) with $\varepsilon_n = \varrho_n + \upsilon_n$. □

II. Justiification of Proposition 1

Here we show that Proposition 1 follows from the f-Norm Ergodic Theorem of Meyn and Tweedie [15] that is stated next.

Theorem 3 (pp $330 - 331$ [15]) Let a discrete time markov chain $\vec{Y}(t)$ (state space $\mathcal{X}$) be ψ-irreducible and aperiodic, and let $\phi_1 \geq 1$ 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 non-negative function ϕ_2 satisfying $\phi_2(\vec{x}_0) < \infty$, for some $\vec{x}_0 \in \mathcal{X}$, and

$$\triangledown \phi_2(\vec{x}) \leq -\phi_1(\vec{x}) + b 1_C(\vec{x}), \vec{x} \in \mathcal{X}$$

where $\bigtriangledown\phi_2(\vec{x}) = \int P(\vec{x}, d\vec{y})\phi_2(\vec{y}) - \phi_2(\vec{x}), \vec{x} \in \mathcal{X}$

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

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

where $P^t(\vec{x}, A) = Pr(\vec{Y}(t) \in A/\vec{Y}(0) = \vec{x})$, $A \in \mathcal{B}(\mathcal{X})$ and $||\nu||_f = sup_{g:|g|\leq f}|\nu(g)|$, $\nu(g) = \int gd(\nu)$ 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 amongst the φs for which the chain is φ-irreducible.
2. A set $A \in \mathcal{B}(\mathcal{X})$ is called ν_a-petite set if $K_a(\vec{x}, B) \geq \nu_a(B)$, for all $\vec{x} \in A$, $B \in \mathcal{B}(\mathcal{X})$, where ν_a is a non-trivial 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 $(a(1) = 1, a(n) = 0, n \neq 1$, $\gamma_a(A) = P(x, A)$, $\forall A \in \mathcal{B}(\mathcal{X})$.). Petiteness of any finite set follows from the fact that the union of two petite sets is petite (Proposition 5.5.5 (ii), page 122, [15]).
3. A chain is recurrent if it is ψ-irreducible and 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 2 of the theorem above, with the petite set being the finite set A of Proposition 1 and b be a real number satisfying

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

(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 w.r.t probability measure π.

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

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 $\pi_t(A) = Pr(\vec{Y}(t) \in A/\vec{Y}(0) = \vec{x})$, $A \in \mathcal{B}(\mathcal{X})$ converges to π where 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, *Proceedings of ACM SIGCOMM,* Ithaca, NY, Sept. 1993, pp. 85-95

[2] J. Bolot and A. Vega Garcia. Control mechanisms for packet audio in the Internet *Proceedings of IEEE Infocom* '96, San Francisco, April 96, pp. 232-239.

[3] D. Cheriton and S. Deering. Host groups: A multicast extension for datagram internetworks. *Proceedings of the 9th Data Communications Symposium* (Sept. 1985), ACM/IEEE New York, 1985, 172-179

[4] F. Chiussi, Y. Xia and V. Kumar. Performance of Shared-Memory Switches under Multicast Bursty Traffic *IEEE Journalon Selected Areas In Communications*, Vol 15, No. 3, 1997.

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

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

[7] C. Diot, W. Dabbous, J. Crowcroft. Multipoint Communication: A Survey of Protocols, Functions, and Mechanisms *IEEE Journal on Selected Areas In Communications,* Vol. 15 No. 3 April 1997.

[8] H. Eriksson. Mbone: The multicast backbone, *Comm. ACM,*vol 37, no. 8, pp. 54-60, Aug 1994.

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

[10] V. Hardeman, M-A. Sasse and I. Kouvelas. Successful multi-party audio communication over the Internet, *Commun. ACM,* 1997.

[11] C. Huitema. *Routing in the Internet.* Englewood Cliffs, NJ: Prentice Hall, 1995, ch. 11, pp. 235-26

[12] International Business Machines Corporation. *Technical Reference PC Network.* Doc. 6322916

[13] T. Kozaki, N. Endo, Y. Sakurai, O. Matsubara, M. Mizukami and K. Asano. $32x32$ shared buffer type ATM switch VLSI's for B-ISDN's, *IEEE Journal on Selected Areas In Communications,* Vol. 9, pp. $1239 - 1247$ Oct. 1991.

[14] S. McCanne and V. Jacobson. Vic: A flexible framework for packet video *Proceedings of ACM Multimedia,* Nov 1995.

[15] S. Meyn and R. Tweedie. *Markov Chains and Stochastic Stability.* Springer Verlag

[16] E. Modiano and A. Ephremides. Efficient Algorithms for Performing Packet Broadcasts in a Mesh Network *IEEE Transactions on Networking,* Vol. 4. No. 4. August' 96

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

[18] M. Parsa, J.J.Garcia-Luna-Aceves. A protocol for Scalable Loop-Free Multicast Routing, *IEEE Journal on Selected Areas In Communications*, Vol 15, No. 3, 1997.

[19] G. Rouskas and I. Baldine. Multicast Routing with End-to-End Delay and Delay Variation Constraints *IEEE Journal on Selected Areas In Communications,* Vol. 15 No. 3 April 1997.

[20] Sheldon Ross. Stochastic Processes

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

[22] H. Saito, H. Yamanaka, H. Yamada, M. Tuzuki, H. Koudoh and Y. Matsuda and K. Oshima. Multicasting function and its LSI implementation in a shared multibuffer ATM switch, *Proceedings of IEEE INFOCOM'94,*, Toronto, Canada, June 1994, pp. $315 - 322$.

[23] S. Sarkar and L. Tassiulas. A Framework for Routing and Congestion Control in Multicast Networks, *Technical Report* $\mathrm{TR}98 - 64$, Institute for Systems Research and University of Maryland, 1998.

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

[25] A. Shaikh and K. Shin. Destination-Driven Routing for Low-Cost Multicast *IEEE Journal on Selected Areas In Communications,* Vol. 15 No. 3 April 1997.

[26] L. Tassiulas and A. Ephremides. Stability properties of controlled queueing systems and scheduling policies for maximum throughput in multihop radio networks, *IEEE Transactions on Automatic Control,* $37(12) : 1936 - 1946$, December 1992.

[27] L. Tassiulas. Adaptive Back Pressure Congestion Control based on Local Information *IEEE Transactions on Automatic Control,* Vol. 40 No. 2 February 1995.

[28] L. Tassiulas. Scheduling and performance limits of networks with constantly changing topology. *IEEE Transactions on Information Theory,* Vol. 43 No. 3 pp. $1067 - 1073$, May 1997

[29] H. Y. Tzeng and K. Y. Siu. On Max-min Fair Congestion Control for Multicast ABR Service in ATM. *IEEE Journal on Selected Areas in Communications,* Vol. 15 No. 3 April 1997.

[30] E. Varvarigos and A. Banarjee. Routing Schemes for Multiple Random Broadcasts in Arbitrary Network Topologies *IEEE Transactions on Parallel and Distributed Systems* Vol 7, No. 8, Aug' 96.

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

[32] B. Vickers, M. Lee and T. Suda. Feedback Control Mechanisms for Real-Time Multipoint Video Services *IEEE Journal on Selected Areas In Communications,* Vol. 15 No. 3 April 1997.

[33] L. Wei and D. Estrin. The tradeoffs of multicast trees and algorithms *Proc. Int Conf. Comp. Comm. Networks,* San Francisco, CA, Sept. 1994

[34] White board software available through ftp://ftp.ee.lbl.gov/conferencing/wb/.

Symbols used throughout the paper

Symbols used in Section VIII

Symbols used in Appendix A