8

Communication Channels for Data Multicasting in Multi-service Networks*

K. Ravindran,
Department of Computing & Information Sciences,
Kansas State University Manhattan, KS 66506 (USA).

Abstract

The paper generalizes the abstraction of tree structured communication channels used in wide-area data multicasting by introducing the notions of: i) *acyclic graph* structured channel to provide connectivity among user entities in an application through network nodes and links, and ii) user specifiable attributes of data flow through a channel, namely, *data directionality* and *data transfer rates.* The abstraction is useful for multi-service networks where applications have diverse transport requirements such as multi-source broadcasting to a common set of destinations (e.g., video conferencing) and bidirectional/unidirectional data flow (e.g., broadcast audio). The directionality is used by the network to create unidirectional or bidirectional edges in the graph over the links between nodes through which data may flow in only one or both directions, as the case may be. The data rate may be mapped to resource demands on the network, viz., link bandwidth and node buffers, to transport data over the edges. The data directionality and data rate information are part of user level *functional binding* to graph structured channels in the network, and allow the network to effectively 'micro-manage' the resources. This becomes desirable, particularly with a large number of users and diverse transport requirements of applications, as in multi-service networks. Using the multicast graph abstraction, the end-systems in the network can directly support many-to-many communication structures for applications such as video conferencing and digital TV.

Key words

Multicast routing, flow specification, backbone network topology, resource reservation protocol, network-wide resource control.

*Network modeling and system development & implementation components of this work are supported by **US AirForce Rome Laboratory** under contract: F30602-94-C-0241.

S. V. Raghavan et al. (eds.), *Computer Networks, Architecture and Applications*

1 INTRODUCTION

Recent research on data multicasting in wide-area networks has proposed *tree structured channels* whereby the switching system in the network replicates data from the source at root of a tree and forwards the data over pre-established multipoint paths to the destinations at leaves of the tree [1, 2]. See Figure 1. Given a tree in the network rooted at a source, multicasting requires setting up a path segment from a destination d to the tree through a set of intervening switches and links so that d can receive the data sent over the tree. The multicasting function in a switch then has two components:

- Selecting a set of links through which a data received on a link is to be sent out on its way to destinations (*routing*);
- Allocation of switch buffers and link bandwidths to receive data over the incoming link and send data over the outgoing link(s) attached to the switch (*resource allocation*).

Accordingly, a tree structured channel may be viewed as indicating: i) logical connectivity between a source and its destinations through a network-wide data path consisting of switches and links, and ii) network-wide resource allocations at the switches and links in this path, to support data flow from the source to destinations.

Tree structured channels have been successfully used in the network to support multicasting for many computer applications such as mail distribution and decentralized 'object location' in a network [1, 3]. In this paper, we study the abstract properties desired of these network channels to meet the diverse transport requirements of an application in multi-service networks such as multi-source broadcasting of data to a common set of destinations (e.g., client access to a replicated database), bidirectional and unidirectional data transfers among entities (e.g., video conferencing, audio distribution), and widely varying transfer rates of data from sources (e.g., audio, video and graphics).

The canonical communication structure of a multi-service application at the multicast data transport level may be viewed as a flat grouping of source and destination entities, with the network establishing logical connectivity among these entities and allocating resources in the underlying path, to enable flow of data between them. With use of a tree structured channel as the basic network level object to embody the data multicasting functions, the network needs to create multiple trees to support the application, where each tree is rooted at a distinct source and a destination connects to each of these trees separately to receive data from the various sources ('tree 1' and 'tree 2' rooted at sources s_1 and s_2 respectively in Figure 1). The network however cannot relate these trees as supporting a single application, without maintaining additional state information about the transport level grouping of source and destination entities connected by different trees. Accordingly, the data multicasting overhead in the network (viz., link cost incurred to

support multiple path segments, channel identifier space consumed and extent of routing control actions required) for the application increases linearly with the number of sources even when these trees have overlapping path segments. Secondly, the application level communication structure needs to compose multiple network channels for exercising the required data transport from various sources to a common set of destinations across the user-network interface (UNI), thereby requiring separate local bindings to each of the channels across the UNI (such as 'call reference numbers' in Broadband ISDN Signalling Protocols). This is inflexible in comparison to the case where a single instance of ***functional binding*** to channels, i.e., binding based on transport attributes supported on channels, across the UNI suffices for exercising the data transport and allows uniformity in the communication structure. For example, a dynamic conferencing group that allows varying number of sources/destinations can be uniformly structured with a functional interface to the network level multicast transport. Thus it is desirable to provide a flexible multicast service model that allows transmission of the information on transport attributes of an application from the users to the network to parameterize the channel characteristics and reduce the multicasting overhead.

Our model is based on a network level abstraction of *acyclic multicast graph* that embodies logical connectivity across different sources and destinations and resource allocations in the underlying paths. A graph has vertices in the switches and edges over the links, with each user residing in a vertex and data flowing from sources to destinations in a given application through various edges. The direction of data flow projects a graph into multiple trees with root at various sources and leaves at the destinations. In other words, a graph is an unrooted tree. A user is allowed to specify *data directionality*, i.e., whether a user can only send or only receive or both send and receive data, through a graph. The directionality is used by the network to create unidirectional or bidirectional edges in the graph, as may be necessary. A user may also specify *data rate*, i.e., the bit rate at which the user can send and/or receive data, through the graph. The data rate may be mapped to resource demands on the network, viz., link bandwidth and switch buffers, to transport data from and/or to the user. Effectively, the transport attributes of data flow (viz., data directionality and data rate) across all user entities in the application allow the network to 'micro-manage' the resources. Thus a graph structured channel is a network level object that can be accessed by user entities through a functional interface.

With the multicast graph abstraction, there is potential for an underlying routing algorithm to reduce the network overhead for data transport by combining the data from various sources to flow over common links towards destinations thereby amortizing any fixed link costs and routing control actions across these sources. This becomes desirable, particularly with a large number of users and diverse transport requirements of applications, as in multi-service networks. Using multicast graphs, the end systems in the network can build a variety of high level many-to-many communication structures for

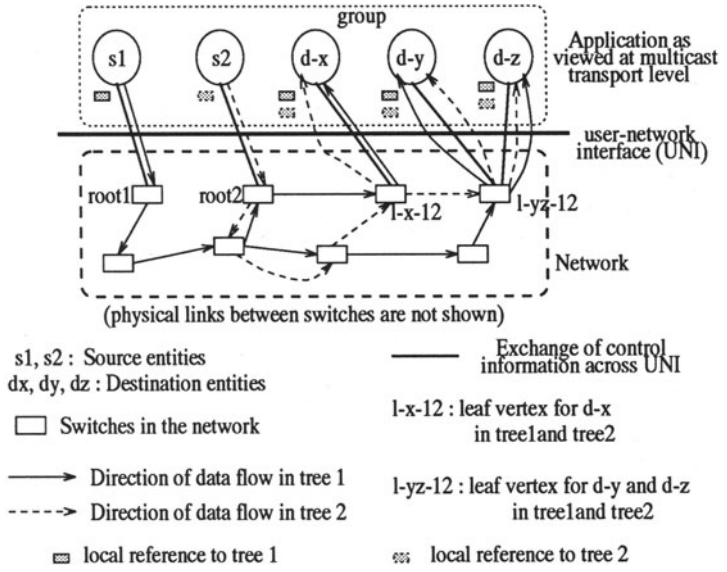

Figure 1: Tree structured communication channels.

applications such as conferencing, video distribution and distributed computing. One-to-one communication (or unicasting) for applications such as video-telephony is elegantly subsumed into this approach. The paper also deals with resource allocation control in the network to support our graph abstraction.

The paper is organized as follows: Section 2 discusses related works. Section 3 describes our model of multicasting using acyclic connectivity graphs. Section 4 gives a view of multicast graphs oriented towards resource allocations in the network. Section 5 describes the functional elements needed in the network to support the model. Section 6 shows a realization of sample applications using the model. Section 7 concludes the paper.

2 RELATED WORKS

Consider the multicast models for low-to-moderate speed wide-area networks such as ARPANET and interconnected LAN's [1]. These models support primarily applications with less diverse transport requirements such as distributed databases and mail distribution, and hence are inadequate for the evolving applications such as video conferencing, digital TV and multi-user graphic visualization. For instance, these models do not allow user level flow specification of the data to the network. The recent work on multicasting

of real-time audio over wide-area internets [4] is primarily along the lines of providing additional support elements in the network (e.g., buffer reservation at various switches in the multicast tree) rather than providing a generic higher level service model that can systematically augment the network with the support elements.

The Internet Stream Protocol ST-II [5] and the resource reservation protocol RSVP [6] incorporate new features expressly designed to support the evolving applications, namely, allowing a flow specification (primarily, data rates) to be systematically transmitted from users to the network switches. The ST-II allows the flow specification from a source to be propagated to agents located at the vertices of a tree rooted at the source so that these agents may make resource allocations in the switches to support the required flow. But the protocol by itself does not provide a systematic mechanism by which a given switch can handle the flow specifications from multiple sources that send data through the switch (though an implementation of the protocol can install local procedures to handle the multi-source flows). The RSVP allows multiple sources in an application to specify their flows to the network; however, it does not deal with how the multi-source flow specification can be incorporated into a network abstraction that allows a systematic use of this flow information by the switches. In this sense, our work provides elements of a canonical network model that can be instantiated into appropriate flow specification protocols (such as ST-II and RSVP) executing at the UNI.

3 OUR MODEL OF MULTICASTING

The network consists of a set of nodes (or switches) $\mathcal{V}$, interconnected with one another through high speed communication links. Users, viz., service providers and subscriber terminals, may be attached to various network nodes. See Figure 3. A user is capable of exchanging data with one or more users, such as in TV broadcasting, video conferencing and voice phone. The data switching functions of network nodes are provided by the backbone network such as ATM switches in MANs, routers in interconnected LANs and packet switches in high speed WANs.

The user entities implementing an application are the end-points of communication, with nodes $\mathcal{U} \subseteq \mathcal{V}$ containing the user entities forming a *multicast set.* A user may have both send and receive capability (UPDOWN), only send capability (UP) or only receive capability (DOWN). These capabilities depict end-to-end directionality of data flow through the network. Users in a multicast set indicate transport requirements on the network in the form of a *data generation rate* by UP or UPDOWN users (i.e., sources) and a *data consumption rate* by DOWN or UPDOWN users (i.e., destinations). In an example of conference, the source is the initiator of a conversation and the destinations are the other participants in the conference browsing the conversation. The data rates may be part of a user-specified *quality of service* (QOS) required of the network. The transport

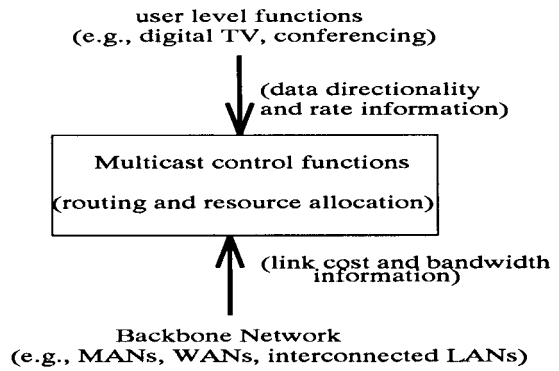

Figure 2: Layers of functions in network architecture.

attributes, viz., data rates and data directionality, are used in defining an appropriate network control structure to support multicasting.

The underlying communication architecture consists of: i) user-level functions to generate and consume data, ii) backbone network that manages physical connectivity among switches $\mathcal{V}$, and iii) *multicast control functions* that allow routing of data from sources to destinations over a path consisting of a set of switches and links in the backbone network. See Figure 2. A multicast data transport incurs a *cost* on the backbone network in the form of link bandwidth allocation and link management overhead. The multicast control functions define an abstraction, viz., *graph structured channel*, for data transport. The abstraction allows the routing functions to systematically utilize the data rate/directionality information from the users and the link cost/topology information from the backbone network so that a network-wide performance index can be optimized (e.g., finding a minimal cost path from the sources to destinations).

We discuss below the multicast graph abstraction (hereafter, the term 'network' includes the multicast control functions, unless stated otherwise).

3.1 Multicast connections

A *connection* supports multipoint communication among users through a set of network nodes $\hat{\mathcal{U}}$, where $\mathcal{U} \subseteq \hat{\mathcal{U}} \subseteq \mathcal{V}$. The network views the connection as an acyclic graph $\mathcal{G}(\hat{\mathcal{U}}, \hat{\mathcal{E}})$ providing logical connectivity among users, with vertices residing in $\hat{\mathcal{U}}$ and edges $\hat{\mathcal{E}}$ mapping to the intervening links. Refer to Figure 3. The flow of data from a node x to a neighboring node y over the link between these nodes, where $x, y \in \hat{\mathcal{U}}$, manifests as traversal of the edge $e(x, y) \in \hat{\mathcal{E}}$ connecting the adjacent vertices in x and y. A route between vertices, say, $\hat{u}, \hat{u}' \in \hat{\mathcal{U}}$ consists of a set of adjacent edges $[e(\hat{u}, t), e(t, \_), \cdots, e(\_, q), e(q, \hat{u}')]$. Figure 3 shows a graph that connects user entities U_a, U_b and U_d through a backbone net-

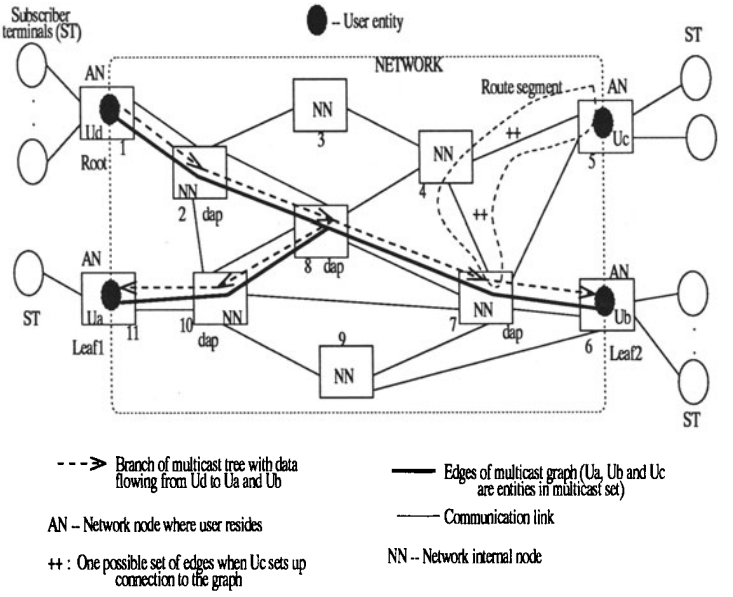

Figure 3: Graph structured communication channels.

work with $\mathcal{V} = \{1, 2, \cdots, 10, 11\}$. For this graph, $\mathcal{U} = \{1, 6, 11\}$, $\hat{\mathcal{U}} = \{1, 2, 8, 7, 6, 10, 11\}$ and $\hat{\mathcal{E}} = \{e(1,2), e(2,8), e(8,10), e(10,11), e(8,7), e(7,6)\}$.

The network may support one or more graphs, with a switch capable of supporting a vertex belonging to each of these graphs. Typically, each graph may provide connectivity among users in a single application. For example, all participants in a video conference may be connected by a single graph. For the purpose of this paper, we assume a single graph in the network without loss of generality.

3.2 Multicast routes

The vertices and edges in $\mathcal{G}$ participating in a data flow may be projected as a multicast tree $\mathcal{T}_s(\mathcal{G})$ with the root at node $s \in \mathcal{U}$ where the source resides and the leaves at nodes $\{d\} \subseteq (\mathcal{U} - s)$ where the destinations reside. In this tree, the flow of data from s to d can be viewed as a traversal of the edges $e(s,t), e(t,\_), \cdots, e(\_,q), e(q,d)$ through the vertices in $s, t, \cdots, q, d \in \hat{\mathcal{U}}$ in that sequence, with $e(s,t)$ originating from (i.e., directed from) s and $e(q,d)$ incident on (i.e., directed towards) d. The multicast route in $\mathcal{T}_s$ is a merge of the paths $[e(s,t), e(t,\_), \cdots, e(\_,q), e(q,d)]_{\forall d}$, i.e., the union of all these paths such that the common edges in the paths appear in $\mathcal{T}_s$ exactly once. $\mathcal{T}_s$ is thus a spanning tree for the nodes $\{s, t, \cdots, q, d\}_{\forall d}$, with no loops in the data flow along various paths. A vertex $x \in \mathcal{T}_s$

constitutes a ***branch point*** if there is more than one edge directed from x; the data arriving at the branch point x gets replicated along the edges $\{e(x,\_)\}$. Figure 3 illustrates a tree projected from a graph, with root at the node 1, leaves at the nodes 6 and 11, branch point at node 8 and the multicast route as $[e(1,2), e(2,8), [e(8,10), e(10,11)], [e(8,7), e(7,6)]]$.

There can be more than one $\mathcal{T}_s$ projected from $\mathcal{G}$ with root at various nodes $\{s\} \subseteq \mathcal{U}$ such that $\bigcup_{\forall s}\ s + \{d\} = \mathcal{U}$; the data from various sources flow along the edges of $\{\mathcal{T}_s\}$ to reach the destinations. A $\mathcal{T}_s$ may have one or more of its edges overlapping with that of other $\mathcal{T}_s$'s, implying that the multicast routes in the various trees may share one or more links. A vertex $y \in \mathcal{G}$ is a ***junction point*** if there is more than one edge directed towards y; the various data arriving at the junction point y along the edges $\{e(\_, y)\}$ get combined with one another for forwarding. In Figure 3, the node 8 is the junction point for the data arriving along the edges $e(2,8)$ and $e(10,8)$. The multicast route in $\mathcal{G}$ is then a merge of the multicast routes in each of $\{\mathcal{T}_s\}_{\forall s}$. Thus, $\mathcal{G}$ is an unrooted spanning tree for $\widehat{\mathcal{U}}$, representing[1] an end-to-end view of data flow between user entities through the network[2].

3.3 Data access points

Each node $u \in \widehat{\mathcal{U}}$ supports a certain level of data rate by allocating link resources (e.g., buffers) to handle the data arriving at and/or sent by u. This allows data to be 'injected' into and/or 'tapped' from $\mathcal{G}$ through u by user entities, so u can be considered as a ***data access point*** (DAP). When a user U requests connection to $\mathcal{G}$, a routing algorithm in the network locates a node u that can serve as DAP for U and sets up a path from U to u. The path setup involves:

- Creating a *route segment* $[u_1, u_2, \cdots, u_l]_{l \geq 1}$ that consists of:

  Vertices in nodes $u_1, u_2, \cdots, u_l$ adjacent to one another, with u_1 containing U and $u_2, \cdots, u_l$ providing physical connectivity between u_1 and u;

  Edges $e(u_1, u_2), e(u_2, \_), \cdots, e(\_, u_l)$ over the links intervening various nodes;

- Interconnecting the route segment $[u_1, u_2, \cdots, u_l]$ to u with an edge $e(u_l, u)$.

The new graph now is $\mathcal{G}((\widehat{\mathcal{U}} + \{u_1, u_2, \cdots, u_l\}), (\widehat{\mathcal{E}} + \{e(u_1, u_2), e(u_2, \_), \cdots, e(\_, u_l), e(u_l, u)\}))$.

[1] $\mathcal{G}$ need not span all the nodes $\mathcal{V}$ in the network but only sufficient number of nodes $\widehat{\mathcal{U}}$ to connect the users attached to nodes $\mathcal{U}$ (i.e., $\mathcal{U} \subseteq \widehat{\mathcal{U}} \subseteq \mathcal{V}$). In this sense, our notion of multicast graph is unlike the existing notions of implementing multicasting on top of a 'network-wide spanning tree' [7, 8].

[2] A connection refers to a control association between various entities to bind them to a logical unit for data transport, viz., multicast set. It does not refer to any data level association between entities that may be needed to provide a desired degree of reliability in data transport such as 'virtual circuit' and 'acknowledged data transfer'.

Thus the network has the perspective of adding new vertices and edges to $\mathcal{G}$, while the data sources and destinations involved in the connection have the perspective of embedding new trees on the physical topology.

3.4 Network-wide directionality of data

The multicast graph model allows the various data sources $\{s_i\}_{i=1,2,\cdots,N}$ and the destinations to reside in vertices of a single connected acyclic graph $\mathcal{G}$. From a destination point of view, the data from various sources get combined over a common set of links and switches in their path towards the destination. From a source point of view, its data flowing through a switch gets replicated over one or more links attached to the switch in its path towards various destinations. The graph model unifies these two views within a single abstraction, with the various switches where data are combined/replicated and the data flow over the intervening links represented as vertices and edges respectively of a graph. Thus the graph model can be used to provide network level support for multi-source broadcasting of data to destinations.

The various trees $\{T_{s_i}(\mathcal{G})\}_{i=1,2,\cdots,N}$ may have one or more edges of $\mathcal{G}$ common between them. The data from two different sources that share a common edge $e(x,y)$ may flow either in the same direction, say from x to y, or in opposite directions between x and y. The edge $e(x,y)$ is *unidirectional*, directed from x towards y, in the former case and is *bidirectional*, directed from x towards y and vice-versa, in the latter case. Since the combining of data from sources and the replication of data towards destinations may occur at various vertices of $\mathcal{G}$, some edges of $\mathcal{G}$ may be unidirectional with the remaining edges being bidirectional. Consider the graph illustrated in Figure 3 as example. Suppose U_a is an UPDOWN entity, U_b is a DOWN entity and U_d is an UP entity. Along the tree T_{U_d}, the data from the source U_d attached to node 1 flows along the edges $[e(1,2), e(2,8)]$, gets replicated at the vertex in node 8, and then flows along the edges $[e(8,10), e(10,11)]$ and $[e(8,7), e(7,6)]$ to reach the destinations U_a and U_b attached to the nodes 11 and 6 respectively. Along the tree T_{U_a}, the data from U_a flows along the edges $[e(11,10), e(10,8), e(8,7), e(7,6)]$. So in the graph, the edges $e(10,11)$ and $e(8,10)$ are bidirectional, while the edges $e(1,2)$, $e(2,8)$, $e(8,7)$ and $e(7,6)$ are unidirectional; the data from U_a and U_d get combined at the vertex in node 8 to flow over the edges $[e(8,7), e(7,6)]$. Thus for a given set of data rates from s_i's, the logical topology of $\mathcal{G}$, viz., the nodes $\widehat{\mathcal{U}}$ that form vertices of $\mathcal{G}$ and how the various edges $\widehat{\mathcal{E}}$ spanning these nodes are directed, in general reflects the source-destination configuration in the physical topology (i.e., the placement of sources and destinations relative to one another in the physical topology) and the link cost constraints enforced by the multicast routing algorithm.

We now describe how a multicast graph can characterize resource allocations in the network.

4 NETWORK-WIDE RESOURCE ALLOCATIONS

A switch S that contains a vertex $u \in \mathcal{G}$ allocates resources for one or more links of S through which edges of $\mathcal{G}$ are incident on u and/or originate from u. The transport resource allocations for a link include reservation of buffers at a *port* in S through which the link is attached to S and[3] assignment of bandwidth by the backbone network for sending and/or receiving data over the link.

4.1 Data rates of multi-source data

A general scenario of many-to-many communication consists of one or more sources $\{s_i\}_{i=1,2,\cdots,N}$, with each s_i generating data at the rate of q_i, and one or more destinations consuming each the combined data at the rate given by $q_1 + q_2 + \cdots + q_N$, i.e., the transfer rate of a combined data is the sum of the rates specified in each q_i (this holds for peak and average data rates). Consider, for example, a multimedia conferencing with 5 participating window-based workstations. Each workstation may generate video and audio with data rate requirements of 2 mb/sec and 64 kb/sec respectively. The various workstations may form a multicast set with 10.32 mb/sec as the combined data rate requirement. A destination can connect to the s_i's only if it can consume data at the combined rate[4].

The data traffic sent from various s_i's over a graph get mixed at one or more vertices. See Figure 4. Consider a vertex in switch B with edges through ports p_a, p_b and p_c. The data arriving in p_a and p_b are sent out through p_b, p_c and p_c respectively, thereby appearing as a mixed data traffic over the link attached to p_c and the downstream paths from p_c. If q_1 and q_2 indicate the data rates for the data arriving in p_a and p_b respectively, the transfer rate of the mixed data sent out through p_c is $(q_1 + q_2)$.

The network may allocate resources in various vertices of a graph to meet the combined data rate requirement. So after mixing the data traffics from various s_i's, the network need not distinguish between the data from individual s_i's for the purpose of resource allocations[5].

4.2 Relationship between data rates and network resources

We assume the availability of a functional relation to provide the mapping between a data rate and the underlying resource needs at a switch to support data transfer at this rate,

[3]Typically, a port maps to the network interface to a link in a switch.

[4]The data belonging to different sources are separable from the mixed traffic by a higher level filter protocol at a destination.

[5]The distinction may however need to be maintained by the network to handle other transport attributes such as delay jitter and loss sensitivity of various data for determining packet scheduling priorities.

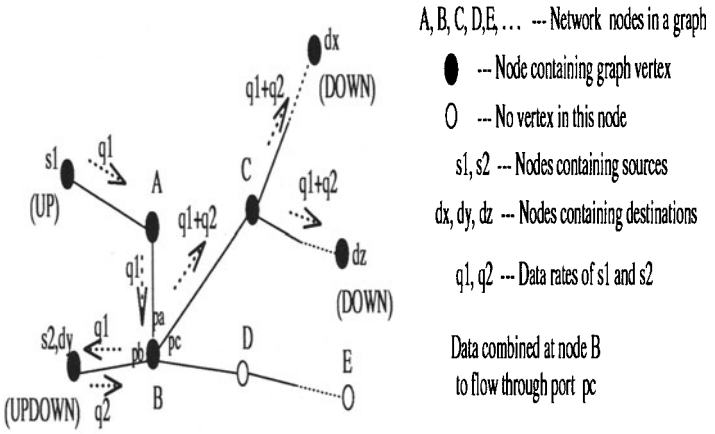

Figure 4: Multi-source data flowing over a graph.

given as below:

$$\mathcal{F} : \mathrm{q} \rightarrow \mathcal{R},$$

where $\mathcal{F}$ provides one-to-one mapping of a data rate to resource needs $\mathcal{R}$, viz., packet buffers and link bandwidth [9]. Consider the earlier example of multimedia conferencing. At the point of delivery of video and audio data to a workstation window, the backbone network should support a link bandwidth of at least 10.32 mb/sec. In general, q is specified as a pair (a, m) indicating the average data rate a and the peak data rate m.

The resource allocations by a switch to meet the requirements of data from various sources $\{s_i\}_{i=1,2,\cdots,N}$ satisfy the *linearity* property, indicated as: $\mathcal{F}(q_1 + q_2 + \cdots + q_N) = \mathcal{F}(q_1) + \mathcal{F}(q_2) + \cdots + \mathcal{F}(q_N)$. The linearity property allows incremental resource allocations and de-allocations when sources connect and disconnect respectively from a graph.

The cost of data transport over a link consists of two components:

(I): Network resources allocated for the data, namely, the link bandwidth assigned and buffers reserved at the port to transport the data;

(II): Overhead in the backbone network for maintaining a connection over the link.

The cost component **(I)** depends on the data rate q over the link, indicated as $\mathcal{F}(q)$, while the component **(II)** depicts a fixed cost that is solely based on the internals of the backbone network (e.g., a lease rental for the 'connect time' over the link). Thus we have the net cost as

$$h.\mathcal{F}(q) \; + \; \text{overhead cost}, \tag{1}$$

where h is a normalization constant that maps the resource needs to a cost[6]. Using equation (1) at various vertices of a graph, the network-wide cost of data transport across various sources and destinations connected by the graph can be computed.

4.3 Resource allocation control

Since only UP/UPDOWN entities can generate data, the vertices in the path from an UP/UPDOWN entity need to support the data rate of only this entity up to the vertex at which data traffic from another source is combined. Similarly, the vertices in the path towards a UPDOWN/DOWN entity need to support a data rate for the combined traffic from all sources. The creation and deletion of edges $\{e\}$ when users connect to and disconnect from $\mathcal{G}$ manifest in allocation and de-allocation of resources respectively in various vertices. Thus $\mathcal{G}$ may be viewed as an indicator of resource allocations at various vertices $\widehat{\mathcal{U}}$, with each edge $e \in \widehat{\mathcal{E}}$ providing an appropriate transport capacity. A reservation protocol handles the resource allocation and release along the data distribution path in $\mathcal{G}$ to ensure that the end-to-end QOS requirements are met.

When a user is to be connected to $\mathcal{G}$, choosing a DAP in $\mathcal{G}$ and a path to the DAP is a resource control policy decision. For instance, the routing algorithm for a user with UP directionality may choose a DAP such that the sum of the 'distance' from the DAP to all destinations and the 'distance' from the switch containing the user to the DAP meets a cost constraint. Typically, the routing algorithm may evaluate alternate paths in the physical topology and then set up a route segment over a path of minimal 'distance'. Thus a dynamic change in the graph size may be viewed as a network-wide resource control activity. For a given source-destination configuration in the physical topology, specific aspects of resource control such as how much resources are allocated and how the resource allocations are distributed across various vertices primarily depend on the data rates of traffic flowing network-wide from various sources to destinations.

We now discuss various elements of the multicast graph model from a resource allocation point of view.

5 FUNCTIONAL ELEMENTS OF MULTICAST GRAPH MODEL

A user specifies a pair $(u\_dir, \{s\_rate, r\_rate\})$ in its request for connection setup to a graph $\mathcal{G}$, where s_rate and r_rate indicate the sending and receiving data rates respectively and u_dir indicates the data directionality. The specification of these transport attributes is part of the functional interface to the network, and constitutes the user's reference to $\mathcal{G}$ across the UNI. We describe in this section how the directionality of data flow interacts with resource allocations in the network.

[6]The costs may be payable by users as a 'tariff' to the network provider.

5.1 Switch level resource needs

Consider a switch S containing a vertex u of $\mathcal{G}$ with edges through ports $\{p\} \subseteq$ **Set_of_Ports**(S), where **Set_of_Ports**(S) refer to the set of ports in S. For a port p linked to a neighboring switch S^p, u may allocate two sets of resources: $\mathcal{R}_s(p,u)$ to send data to and $\mathcal{R}_r(p,u)$ to receive data from the adjacent vertex u' in S^p. An edge through p is bidirectional carrying data both towards and away from p if $\mathcal{R}_s(p,u) > 0 \wedge \mathcal{R}_r(p,u) > 0$, unidirectional carrying data towards p if $\mathcal{R}_s(p,u) = 0 \wedge \mathcal{R}_r(p,u) > 0$, and unidirectional carrying data away from p if $\mathcal{R}_s(p,u) > 0 \wedge \mathcal{R}_r(p,u) = 0$. The following conditions should then hold for a stable data transport over the edge $e(u,u')$:

$$\begin{aligned} \mathcal{R}_r(p',u') &\geq \mathcal{F}\Big(\sum_{\forall x \in (in\_edge(u)-p)} in\_rate_{x,u}\Big) \\ \mathcal{R}_s(p,u) &\geq \mathcal{F}\Big(\sum_{\forall x \in (in\_edge(u)-p)} in\_rate_{x,u}\Big) \\ & \mathcal{R}_r(p,u) \geq \mathcal{F}(in\_rate_{p,u}), \end{aligned}$$

where $in\_rate_{x,u}$ is the data rate for the traffic flowing into u through port x in S, p' is the port in S^p through which S is linked, and $in\_edge(u)$ is the set of ports in S through which an edge is directed towards u. These conditions should hold in the presence of addition/deletion of edges to u and/or changes in the resources allocated for various edges (this is possible with the linearity property of resource allocations). See Figure 5 for an illustration.

5.2 Resource reservation protocols

The aggregation of resource allocations at various switches in the data distribution path, as described above, needs to be embodied into a reservation protocol. Such a protocol may be activated by user invocations for network multicast service across the UNI. The execution model of the protocol may consist of *agent entities* installed in the switches that interact with one another over a control channel (e.g., B-ISDN signaling channels) to coordinate the resource allocations in a decentralized manner. Accordingly, the protocol is specifiable in terms of:

- *Control messages* exchangeable across the agent entities to carry the QOS information pertaining to the flow of data from various sources and the resource availability information at switches;
- *Tables* maintained by the agent entities to describe the mapping function $\mathcal{F}$ in the form of QOS values (viz., average and peak data rates) and the corresponding resource needs.

See Figure 6. Examples of control messages are the 'connect' and 'accept'/'refuse' messages used in ST-II [5]. The explicit representation of $\mathcal{F}$ in the form of tables is needed

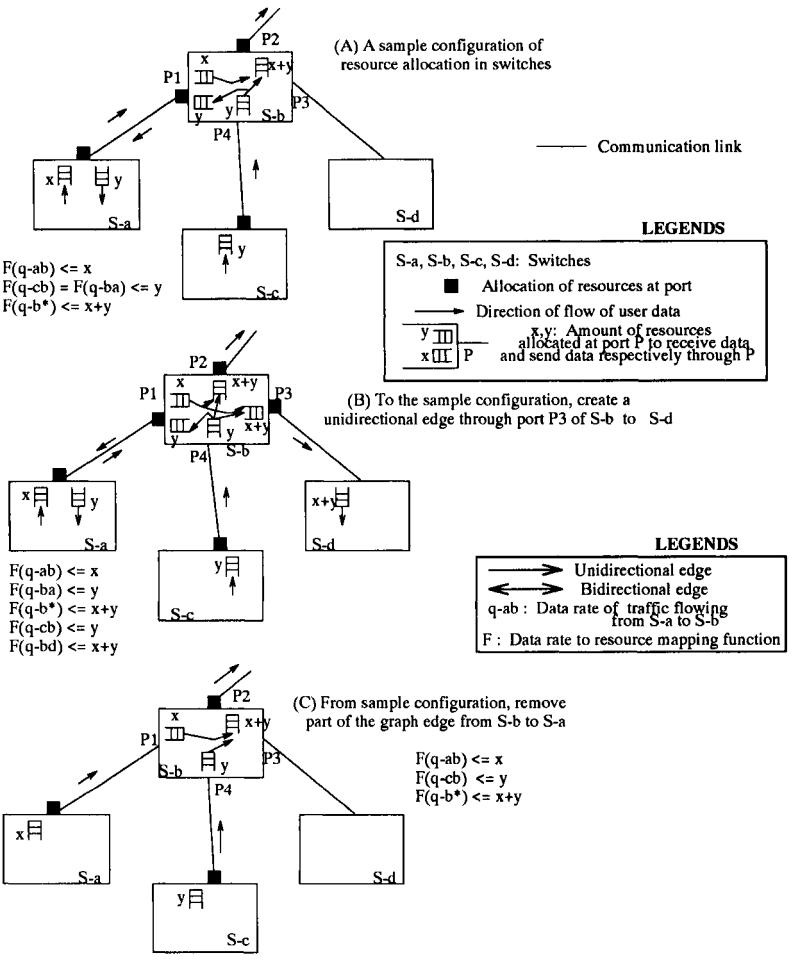

Figure 5: Resource allocation in switches and its representation as directional edges in the network.

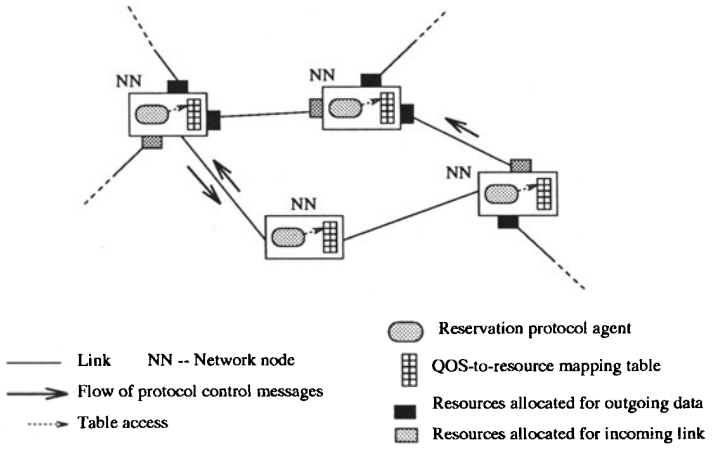

Figure 6: Execution model of resource reservation protocol.

in our graph model to allow dynamic changes in the traffic levels flowing through various switches due to addition/removal of users in the multicast configuration.

Suppose (a, m) is the QOS value currently supported by the reservation agent in a switch, with resources allocated in the amount given by $\mathcal{F}((a, m))$. When the level of traffic flowing into the switch has to be increased by $(\delta a, \delta m)$, the agent computes the resource needs $\mathcal{F}((a+\delta a, m+\delta m))$ from the table and determines the admissibility of the additional traffic by checking if enough resources are available both locally and at other downstream agents to satisfy the increased needs. If the check is successful, the additional resources are committed at appropriate switches in the path. When the traffic level has to be decreased by $(\delta a, \delta m)$, the agent computes the resource needs $\mathcal{F}((a-\delta a, m-\delta m))$ from the table and deallocates the excess resources both locally and at other downstream agents. When resource allocations satisfy the linearity property, the incremental amount of resources may be simply estimated as $\mathcal{F}((\delta a, \delta m))$.

For resource allocation purposes, each agent maintains a local state variable *c_rate* indicating the traffic level flowing through its switch. The inter-agent coordination to allocate/de-allocate resources and to update *c_rate* is achieved by exchanging control messages.

5.3 Routing control information

From our foregoing discussion, we see that a routing algorithm requires, to varying degrees, a global knowledge of the following resource control factors:

- Source-destination configuration in the physical topology;

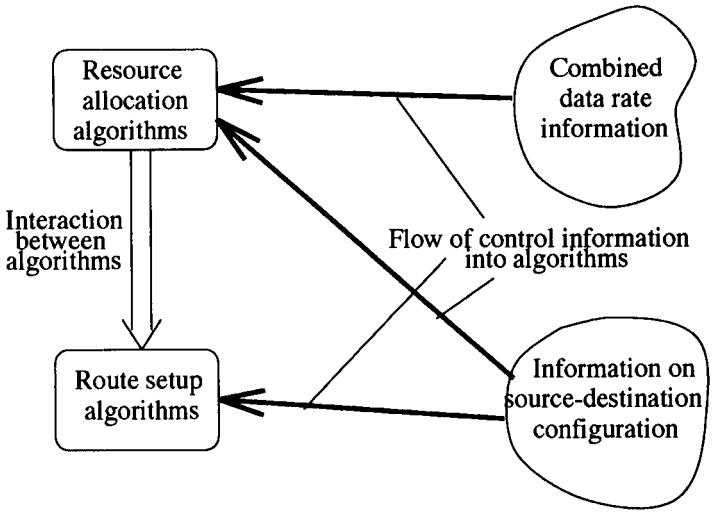

Figure 7: Network level view of resource control algorithms.

- Data traffic flowing network-wide from various sources to destinations.

The information on these factors may be obtained from the data directionality and data rates of user entities and the physical topology of backbone network. So the incorporation of data directionality and data rates as part of the multicast semantics in our graph model allows the required information to be made available to the routing algorithm (often acquired and processed in a distributed manner)[7]. See Figure 7.

A systematic availability of the information on data directionality, data rates and physical topology to the resource control algorithms allows the latter to optimize the cost of multicast data transport. Computation methods studied elsewhere [2, 10, 11] to enforce cost constraints on multicast data transport through tree structured channels may be appropriately incorporated into a chosen routing algorithm executed by the network in an implementation of our multicast model.

It is beyond the scope of this paper to deal with the design of multicast routing algorithms as such (see [1, 2] for routing algorithms studied elsewhere in different network environments). Rather, the paper focuses on providing a network model of multicasting

[7]It should be noted in this context that the network-wide configuration and data flow related information are not directly available to the resource control algorithms when the network employs tree structured connectivity.

that allows systematic acquisition and use of resource control information by a chosen routing algorithm (say, for a network-wide path search [12]).

We now give some sample scenarios to illustrate the dynamic formation of graphs in our model.

6 RESOURCE ALLOCATION TRACE FOR SAMPLE APPLICATIONS

For ease of description, we denote: i) the creation of an out-going edge from a vertex u in a switch through a port p as invoking a network level function **Assign_Edge**(p) by u, and ii) the removal of an out-going edge through p as invoking a function **Deassign_Edge**(p) by u. Note that the creation of the edge requires allocation of resources $\mathcal{R}_s(p,u)$ at p satisfying the data rate requirements and the removal of the edge causes de-allocation of these resources. We also assume a function **to**(S^p) provided by the backbone network that is invocable by a switch S to obtain the port p through which S is linked to the neighbor S^p (the function may also return a link cost value that may be used by a routing algorithm).

Let $\{S_0, S_1, \cdots, S_n\}_{n\geq 0}$ be the switches, in that order, in the path to a DAP, with the user U requesting connection attached to S_0 and the DAP residing in S_n. When $n = 0$, U can simply start sending and/or receiving packets through S_0, as the case may be.

6.1 Digital TV

A TV station initially creates a single-vertex graph with $c\_rate = 0$. A TV broadcaster is an UP entity connecting to a vertex u through unidirectional edges from the TV broadcaster vertex to u, created by invoking **Assign_Edge**(**to**(S_{l+1})) in each switch S_l and setting c_rate to the traffic level indicated by u, where $l = 0, 1, \cdots, n-1$. A TV receiver is a DOWN entity connecting to a vertex u' through unidirectional edges from u' to the TV receiver vertex by invoking **Assign_Edge**(**to**(S_l)) in each switch S_{l+1} and setting $c\_rate = in\_rate_{\textbf{to}(S_{l+1}),\_}$ at S_l. See Figure 8.

6.2 Conference

The conference manager initially creates a single-vertex graph with $c\_rate = 0$. An active participant who can send and receive data is an UPDOWN entity connecting to a vertex u through bidirectional edges created between u and the participant vertex: i) by invoking **Assign_Edge**(**to**(S_{n-1})) in S_n and setting its $c\_rate = c\_rate + in\_rate_{\textbf{to}(S_{n-1}),\_}$, ii) by invoking **Assign_Edge**(**to**(S_1)) in S_0 and setting its $c\_rate = s\_rate + in\_rate_{\textbf{to}(S_1),\_}$, and iii) when $n \geq 2$, by invoking **Assign_Edge**(**to**(S_{l+1})) and **Assign_Edge**(**to**(S_{l-1})) and setting its $c\_rate = in\_rate_{\textbf{to}(S_{l-1}),\_} + in\_rate_{\textbf{to}(S_{l+1}),\_}$ in each S_l, where $l = 1, 2, \cdots, n-1$.

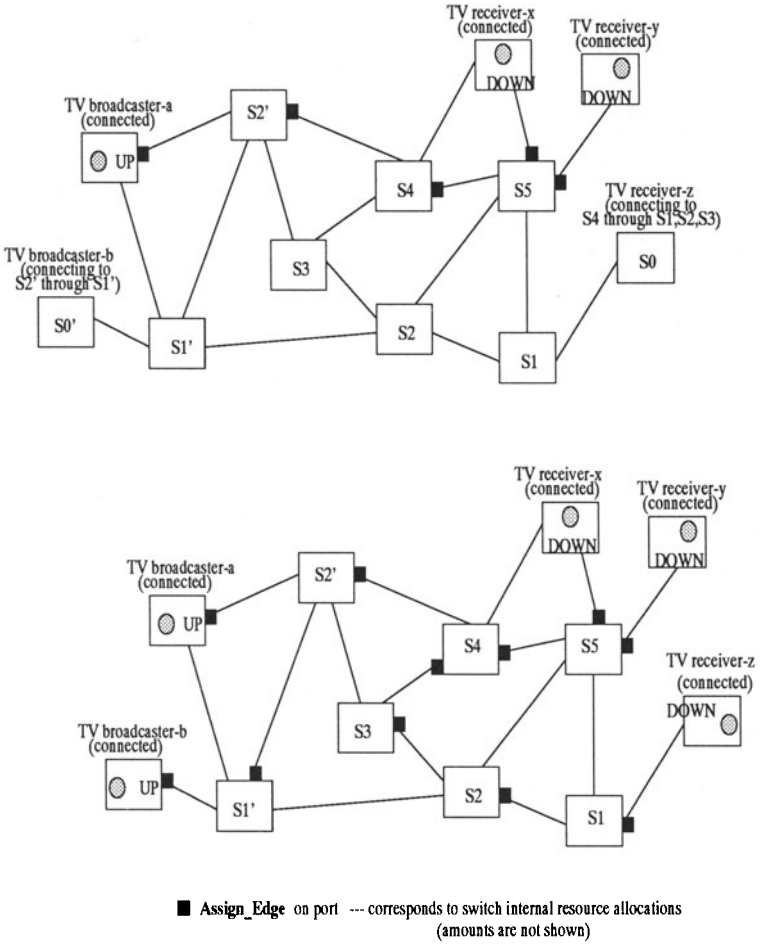

Figure 8: Scenarios before and after multicast connections in TV.

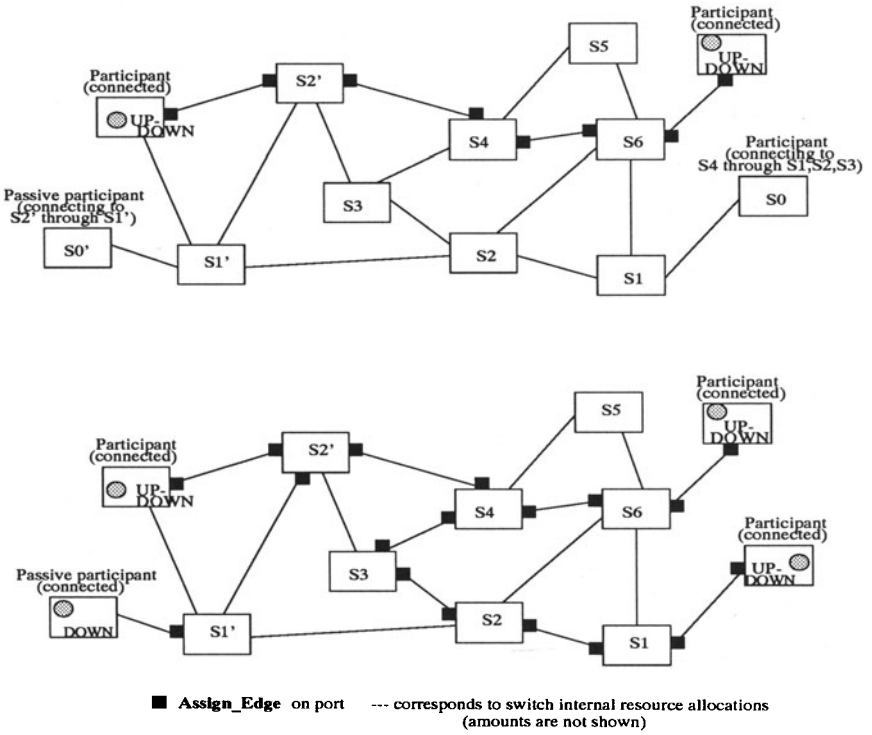

Figure 9: Scenarios before and after multicast connections in conference.

A passive participant who only receive data (e.g., observer) executes similar to the TV receiver discussed earlier. See Figure 9.

The creation of edges connecting the broadcaster in the TV example and the active participant in the conference example to u is preceded by an incrementing of c_rate in all vertices of the graph, including u, by an amount s_rate.

As can be seen, the resource demands on the network largely depend on the physical topology of the network, the location of the connected user entities and the directionality of the user entities.

7 CONCLUSIONS

The paper described a canonical model for a network to offer multicast services to applications. The model incorporates a functional form of the UNI to flexibly allow the network meet the diverse transport requirements imposed by the evolving multi-service applications. These requirements include support for widely varying data rates in com-

munication channels, delivery of data to a large number of destinations, unidirectional and bidirectional data transport, and multi-source broadcasting of data to a common set of destinations.

The paper generalized the abstraction of tree structured channels currently used for wide-area packet multicasting by introducing the notions of: (i) acyclic graph structured channel that connects various users, viz., source and destination entities, in an application through a set of vertices and edges realized over switches and links respectively, and (ii) user-specifiable data directionality and data rates that manifest in allocation of resources, viz., switch buffers and link bandwidths, at various vertices and edges respectively, in the graph. A graph structured channel can be accessed by a user at any of its vertices for sending and/or receiving data over the channel. The user specified directionality projects a graph into multiple trees, where each tree is rooted at a source and has leaves at its destination(s), with appropriate resources allocated in each of the underlying paths to support data flow from the concerned source. Thus a graph structured channel specifies the connectivity among multiple sources and destinations through switches and links, and also the resources allocated for the combined data flow from various sources to destinations.

In contrast, multicast models using tree structured channels [2, 5] often require the network to set up multiple channels for supporting bidirectional data flow and multi-source broadcasting of data in an application. This increases the network overhead for multicasting of data because of the need to manage each channel separately. Multicast models that employ graph structured channels but do not use the directionality information [1] often require the network to support bidirectional edges (i.e., a switch port containing an edge of the graph allocates resources to both send and receive data) by default and require the end-systems to enforce directionality at the point of admitting data traffic into and from the network. In this context, the use of data directionality in the network saves resources in a switch because a switch need not allocate resources in both directions of a link for unidirectional flow of data over this link. Accordingly, our graph abstraction is quite useful for multi-service networks since many applications have unidirectional data transfer requirement (e.g., broadcast video, passive participants in a conference).

Our model of a multicast graph incorporating the directionality/rates of data flow through the graph is usable as an elegant building block for the many-to-many communication among a set of users required in many applications. The model allows the network to map the transport requirements of applications to resource demands at various switches and links. It also optimizes the cost of multicast data transport across the entire application by reducing the data multicasting overhead in the network due to sharing of links by multiple paths.

We believe the model presented in the paper will lend insight into system architectures

that need to support data multicasting, at the network level, in an environment of diverse multi-service applications.

8 CURRENT STATUS OF WORK

The functional model described in the paper is basically a beginning of a comprehensive design and implementation of network architecture for data multicasting. There are many facets to the future activity: formulation of user level primitives at the network interface, development of network internal routing algorithms, embedding of network architecture on operational backbone networks (e.g., ATM-based testbeds), and demonstration of prototype applications (e.g., video conferencing) using the network architecture. This further activity is supported and funded by the **US Airforce Rome Laboratory** through a **R & D** contract.

9 REFERENCES

[1] S. E. Deering and D. R. Cheriton. **Multicast Routing in Datagram Internetworks and Extended LANs.** In *ACM Transactions on Computer Systems*, Vol.8, No.2, pp.85-110, May 1990.

[2] B. M. Waxman. **Routing of Multipoint Connections,** In *IEEE Journal on Selected Areas in Communications*, Vol.SAC-6, No.9, pp.1617-1622, Dec. 1988.

[3] D. R. Cheriton and S. E. Deering. **Host Groups: A Multicast Extension for Datagram Internetworks.** In *9th Data Communications Symposium*, ACM SIGCOMM, pp.172-179, Sept. 1985.

[4] S. Casner and S. E. Deering. **First IETF Internet Audiocast.** In *Computer Communications Review*, Vol.22, No.3, pp.92-97, July 1992.

[5] C. Topolcic. **Experimental Internet Stream Protocol, version 2 (ST-II).** In *RFC 1190*, CIP Working Group, Oct. 1990.

[6] L. Zhang, S. E. Deering, D. Estrin. S. Shenker and D. Zappala. **RSVP: A New Resource ReSerVation Protocol** In *Network*, IEEE Communications Society, pp.8-18, Sept. 1993.

[7] M. D. Schroeder and et al. **Autonet: A High Speed Self-configuring Local Area Network Using Point-to-point Links.** In *Tech. Report*, Digital Equipment Corporation (System Research Center), Palo Alto (CA), April 1990.

[8] D. Bertsekas and R. Gallager. **Routing in Data Networks.** Chapter 5 in *Data Networks*, Prentice Hall Publ. Co., 1992.

[9] A. A. Lazar, A. Temple, and R. Gidron. **An Architecture for Integrated Networks that Guarantees Quality of Service.** *Tech. Report*, Center for Telecomm. Research, Columbia University, 1990.

[10] D. C. Verma and P. M. Gopal. **Routing Reserved Bandwidth Multi-point Connections.** In Proc. *Communication Architectures, Protocols and Applications*, ACM SIGCOMM, pp.96-105, Sept. 1993.

[11] V. P. Kompella, J. C. Pasquale and G. C.Polyzos **Multicasting for Multimedia Applications.** In proc. *Computer Communications*, INFOCOM'92, Italy, 1992.

[12] K. Ravindran and M. Sankhla. **Multicast Models and Routing Algorithsm for High Speed Multi-service Networks.** In proc. 12^{th} *Intl. Conf. on Distributed Computing Systems*, Yokohama (Japan), June 1992.

10 BIOGRAPHY

Dr. K. Ravindran is currently a faculty member at the dept of Computing & Information Sciences, Kansas State University. He received his Ph.D from university of British Columbia, Canada and M.E./B.E. degrees from IISc Bangalore. He has held positions at ISRO (Bangalore), Bell-Northern Research (Canada) and IBM European Networking Center(Germany). His research interests are in Distributed systems modeling and design, high speed network architectures and protocols.