Architectural and Protocol Frameworks for Multicast Data Transport in Multi-service Networks **

K. Ravindran
Department of Computing and Information Sciences
Kansas State University
Manhattan, KS 66506, USA.
E-mail: *ravi@cis.ksu.edu*

25th January 1996

Abstract

The paper presents architectural and protocol techniques for multicast transport of multi-service data (such as video, audio and graphics). A tree-structured channel in the network is employed as building block for constructing multicast transport services. Data from different sources are multiplexed over a shared tree channel for reaching destinations through intermediate network nodes and links. The choice of multicast architecture is based on the extent of sharing of common links in a tree across multi-source data flows to allow finer degree of link bandwidth allocation control and reduce the fixed cost of links for data transport. The canonical network substrate so constructed exemplifies a 'programmable network' that may be 'plugged-in' with QOS and flow parameters to instantiate the network behavior for matching the needs of each application. This philosophy is in alignment with the evolving 'Internet Service Layer' functionalities. The paper describes the functional and protocol elements of multicast architectures to support the 'programmable network' view. It also describes the salient features of an implementation embedding these architectures.

Key words: Distribution trees, shared path segments, network-wide transport cost, flow and QOS support functions, functional view of routing algorithms, routing control protocol, architecture embeddings.

** Part of this work is supported by **US Airforce Rome Laboratory**, under contract: F30602-94-C-0241.

1 Introduction

The currently evolving multi-service applications have diverse transport requirements such as multi-source broadcasting of data to a common set of destinations, large volumes of bidirectional and unidirectional data transfers among users, and widely varying transfer rates of data from sources (e.g., video conferencing, broadcast audio, graphics image distribution). Also, there may be a large number of receivers, with somewhat different bandwidth capabilities, connected to an on-going data distribution through the network (such as audiocast and videocast over the 'Internet' [2, 1]).

The diversity in transport requirements necessitates a fine granular control over the allocation of resources in network channels to support a given data flow. For example, a transfer rate of 10 *megabits* (mb)/sec for video data requires 1 mb buffers in a network node for a 0.1 sec switching interval, while a transfer rate of 1 mb/sec for image data requires 100 *kilobits* (kb) buffers for this interval. Such a finer control manifests in the form of a tighter linkage between the user specifications of data flow and the network resource allocations.

From the above perspective, the paper describes the main elements of multicast transport architectures. An architecture typically employs *tree-structured channels* for data distribution, whereby the switching system in network replicates the data available at root of a tree and forwards the data over pre-established multipoint paths to destinations at leaves of the tree [3, 4]. We believe that a transport architecture should be *network-centric* in that the user-network interface allows casting the needs of applications onto network internal mechanisms. For example, a path leading to audio-only terminals may be set up over a 100 kb/sec link as against a path leading to video+audio terminals over a 3 mb/sec link. In such an architecture, the user-level flow specifications are transcribed to the network for the latter to determine a cost-optimal data distribution path. See Figure 1.

In determining an appropriate architecture, we employ 'programmable network' as the underlying theme that is becoming the basis for building large networked systems, such as the Internet. There are somewhat different schools of thought on the notion of 'programmable network':

Service-oriented view **[5]:** A network model that can offer a variety of service classes for data delivery such as delay jitter-free and jitter-allowed data transport for high-fidelity audio and text based conferencing respectively;

Resource allocation-oriented view **[6]:** A network model that allows users to define "data flows" and their requirements which can be passed on to network subsystems for resource allocations such as bandwidth, buffers and state in the routers;

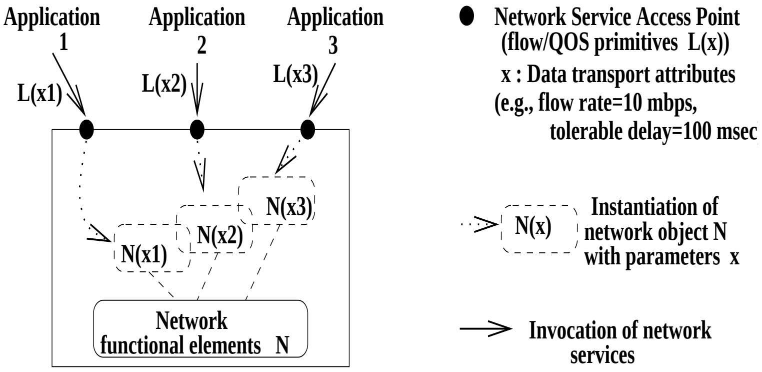

Figure 1: Network service interface viewed as 'programmable network'

Network-independent bearer service view [7]: An abstract organization of protocols and service interfaces not tied to any existing protocol suite, which is capable of cross-connecting applications to backbone networks.

We believe that the above views are intricately inter-woven with one another, and hence need to be addressed in the design of any large network system. Accordingly, our treatment of multicast transport architectures is guided by these views.

A concretization of the "programmable network" view in transport architectures manifests in the following aspects:

- Fine granular control on network resource allocations through application level QOS/flow specifications;
- Extensibility of allowing QOS/flow based 'network parameterization' for diverse applications.

Incorporating these aspects, we describe canonical models/architectures for multicasting and then map them to operational backbone networks (such as ATM networks and interconnected LANs). See Figure 2.

We have analyzed the functional requirements of multicast transport architectures through studies, simulations and prototype implementations on operational networks. However, these requirements are themselves independent of target network specifics. The paper also evaluates the effectiveness of various alternative transport architectures in supporting multicast-based applications (such as multimedia conferencing).

The paper is organized as follows: Section 2 identifies the basic ingredients required of multicast transport architectures. Section 3 compares currently available architectures in supporting multi-service applications. Section 4 identifies the canonical functional elements required of the network for multicast transport. Section 5 describes how multicast routing algorithms can be designed in such architectures. Section 6 describes a case study of architecture implementation. Section 7 concludes the paper.

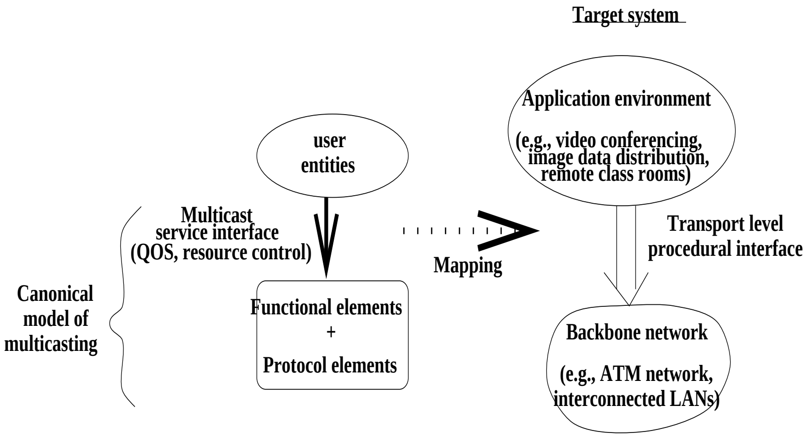

Figure 2: A bird's view of our approach to developing multicast service model

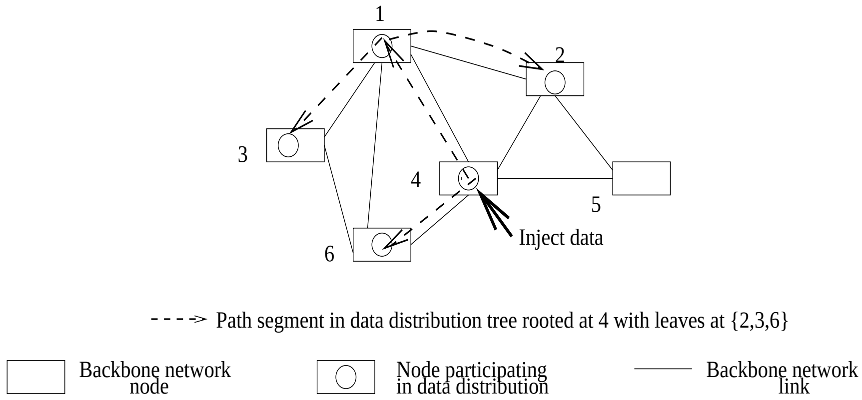

Figure 3: Distribution tree as building block for multicast architectures

2 Identifying architectural elements

In this section, we describe the basic elements of multicast architectures for supporting flow and QOS specifications.

2.1 Distribution tree

A multicast tree represents a logical topology superimposed on backbone network nodes and links such that a data can be injected at root node of the tree and get replicated along intermediate branches of the tree for reaching destinations at leaf nodes of the tree [3, 4]. See Figure 3. Since a tree topology does not have cycles, the forwarding of a data unit received over the input link of a node is only a matter of determining the output links to send the data unit. The data from various sources are made available at the root node of a tree through one or more point-to-point paths for forwarding towards destinations.

Each node in a multicast path implements: i) a routing function that determines the outgoing links for

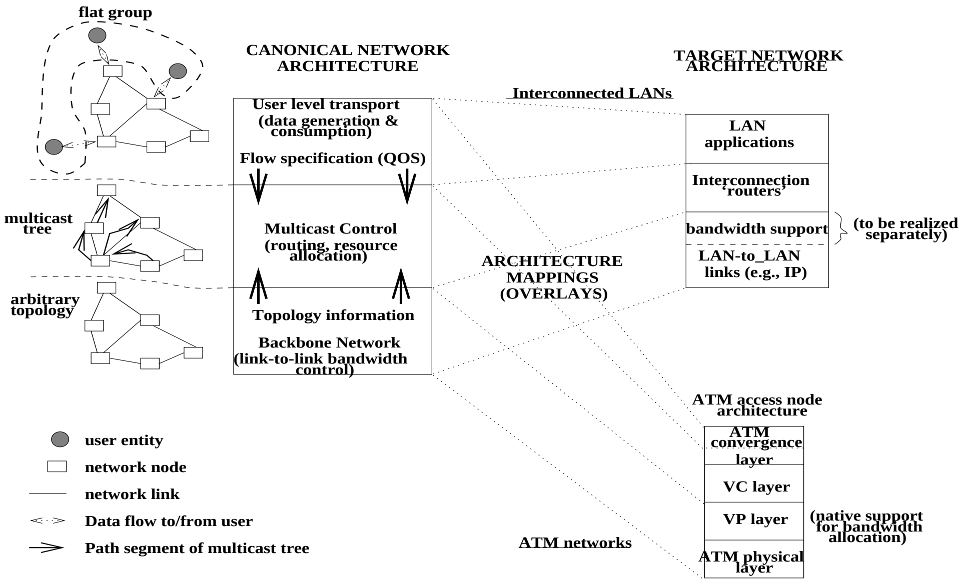

Figure 4: Layers of functions in a multicast transport architecture

an incoming data, and ii) a resource allocation function that reserves nodal buffers and link bandwidths to sustain the required data flow. The setting up of tree-structured path and point-to-point paths spanning the sources and destinations in an application is carried out by a multicast routing algorithm. The path determination may be based on the total amount of resource consumptions and routing overhead in getting the data flows from sources to destinations through the backbone network (transport cost).

2.2 Layers of transport functions

The source and destination entities involved in a multicast data transport may be viewed as organized in a flat group (UTL). These entities exercise multicast control functions (MCL) to set up a tree-based distribution path over the physical topology of backbone network nodes and links that can carry the required flows from sources to destinations. The backbone network (BNL) is assumed to provide node-to-node link bandwidth reservation upon demand.

The UTL, MCL and BNL functions are mappable onto appropriate layers in a target network architecture. The mapping may involve the canonical functions in these layers to be bound to the local functions in the target network. See Figure 4.In interconnected LANs for instance, the UTL functions reside in LAN applications, MCL functions in interconnection routers, and BNL functions in LAN link-to-link connectivity layer augmented with bandwidth allocation control. In ATM networks for instance, the BNL functions map to the 'virtual path' (VP) layer with built-in bandwidth allocation support, and the MTL functions are placed in the 'virtual circuit' (VC) layer and partly in the convergence layer above, exercising the VP level multicast

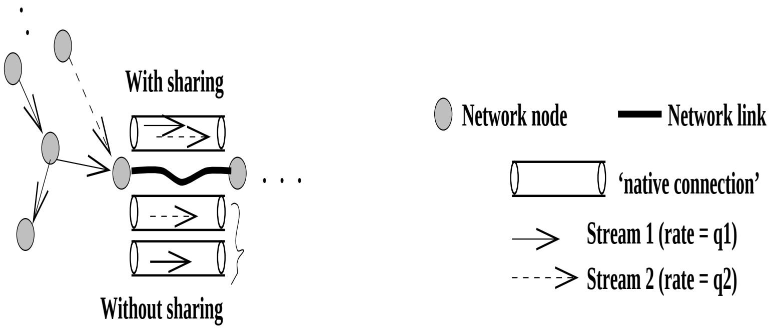

Figure 5: Sharing of multicast path segments across streams

provided in ATM networks (see section 6). Appropriate convergence sublayers may be employed in target networks to support these mappings.

2.3 Sharing of path segments

The data stream from each source flows over a tree towards a common set of destinations. The sharing of a tree segment, i.e., backbone network 'connection', across different data streams ('stream multiplexing' over a path segment) allows:

- Amortization of the 'connection' fixed costs (e.g., routing table entries and 'connection' descriptors) across the various streams;
- Better use of 'connection' bandwidth by exercising 'statistical interleaving' of streams along this segment and its downstream path segments, particularly when data flows are bursty (such as compressed video and text);

in comparison to sending each stream over a separate 'connection' set up over backbone network link. See Figure 5 that shows flows 1 and 2 in an application with rates q_1 and q_2 respectively. Empirically, the cost of these flows when multiplexed over a single 'connection' set up over a link may be given by:

$$\begin{aligned} C_{mux} &= fc \ + \ h.(q_1 + q_2) \quad \text{when } q_1, q_2 \text{ are average rates} \\ &= fc \ + \ h.(q_1 + q_2)^k \quad \text{when } q_1, q_2 \text{ are peak rates (for bursty flows),} \end{aligned} \tag{1}$$

where fc is the fixed cost of using the 'connection', $0 \ll k \leq 1.0$ such that data loss does not exceed an application-specified limit Δ, and h is a constant. The equation (1) can be generalized to any number of flows. Here, for a given Δ, k decreases as the number of streams multiplexed, data burstiness and/or flow rate is increased. And for a given set of flow parameters, k decreases as Δ is increased. Our experiments on ATM network traffic using 3 bursty streams of 15 $mbps$ each with 'burst factor' of 0.5 show that $k = [0.9, 1.0]$

for $\Delta = 2\%$ (e.g., voice data can tolerate up to 2% loss without noticeable degradation in quality). When the network cannot relate the flows as belonging to a single application, they may be sent over separate 'connections' set up over the link. Here, the cost of flows 1 and 2 may be given by

$$\begin{aligned} C_{no\_mux} &= 2.fc \; + \; h.(q_1 + q_2) \quad \text{when } q_1, q_2 \text{ are average rates} \\ &= 2.fc \; + \; h.(q_1 + q_2)^{k'} \quad \text{when } q_1, q_2 \text{ are peak rates (for bursty flows)}, \end{aligned} \tag{2}$$

where $k \leq k' \leq 1.0$. The parameter k' takes into account any 'statistical interleaving' of data units over the link, as supported internally by the backbone network (e.g., the multiplexing of 'cells' of different 'virtual circuits' over 'virtual path' links in ATM networks, based on network-assigned 'bandwidth classes').

As can be seen, $C_{no\_mux} > C_{mux}$. The less savings in bandwidth allocation arise because the network does not have information that will enable it to determine the criteria for 'statistical interleaving' of streams with respect to Δ.

Thus, sharing of path segments across data streams reduces the per-stream fixed costs, and offers the potential for reducing the flow costs[1]. The capability to share tree channels across data streams is itself specific to a network architecture for multicasting.

2.4 Architectural support for shared trees

Here, 'sharing' means that when a stream gets combined with another stream, both the streams are routed through common links all the way downstream towards destinations.

A key feature of multicast architectures is the set of nodes in physical topology of backbone network that are allowed to be chosen as *stream multiplexing points* (SMP). The data from various sources flow to a SMP node, and then the combined data flows along a tree rooted at this SMP node towards destinations. See Figure 6. A routing algorithm determines the data flow paths so as to optimize the network-wide cost of data transport. Thus, for a source in node s and destinations in nodes $\{d\}$, the transport cost may be given as:

$$\begin{aligned} net\_cost(s, \{d\}) = \;& flow\_cost(s, [s, \mathrm{SMP}(s)]) + mux\_flow\_cost(s, [tr(\mathrm{SMP}(s), \{d\})]) + \\ & fixed\_cost([s, \mathrm{SMP}(s)]) + fixed\_cost([tr(\mathrm{SMP}(s), \{d\})]), \end{aligned} \tag{3}$$

[1] As an example of analogy, consider the bursty arrival of cars along multiple inlet roads at a toll junction. The number of open toll booths corresponds to 'resources allocated' and how often the wait-time of car drivers at the toll junction exceeds a tolerance level corresponds to 'Δ'. All the inlet roads served by a single set of booths versus each inlet road served by a separate smaller set of booths correspond to sharing versus no-sharing. In a case of 2 inlet roads, the peak and average arrival rates as 10 and 3 cars/minute respectively on each road, the maximum burst width as 20 cars, and the toll service rate as 4 cars/minute/booth, an average worst-case wait time of, say, 0.5 minute, requires opening 3.35 booths in shared service mode. This is lower in comparison to the non-shared service mode where 2 booths need to be open on each road (i.e., 4 booths total) for the same level of wait. Thus a given level of 'driver happiness' can be attained with a less number of toll booths in the shared service mode when compared to the non-shared service mode.

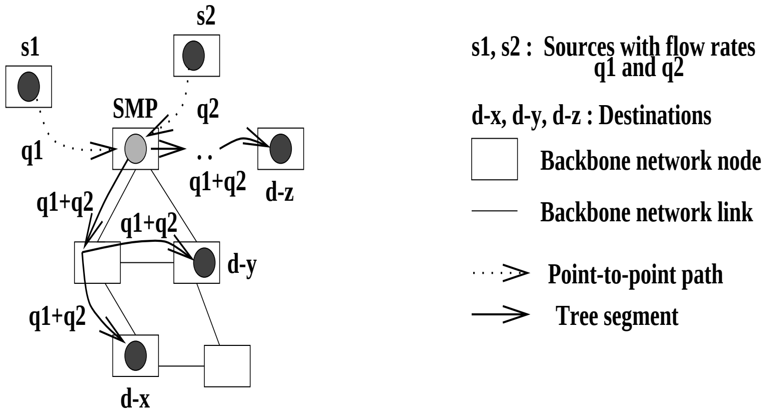

Figure 6: Illustration of architectural support for path sharing

where $[s, \mathrm{SMP}(s)]$ refers to a point-to-point path from s to $\mathrm{SMP}(s)$ and $[tr(\mathrm{SMP}(s), \{d\})]$ refers to path segments in the tree with root at $\mathrm{SMP}(s)$ and leaves at $\{d\}$. The mux_flow_cost depends on the number of hops in the tree, the bandwidth needs of flow from s, the number of streams multiplexed with s, and the burstiness of various streams; the $flow\_cost$ depends on the number of hops in the path and the bandwidth needs of s; the $fixed\_cost$ depends on the number of hops in the path, fc, and the number of streams multiplexed. In general, more the number of streams multiplexed along a path, lower the net_cost.

We now analyze the extent of support for shared distribution paths in current multicast architectures.

3 Comparative analysis of multicast architectures

Sharing of distribution paths in the network across data streams flowing from various source entities to destination entities in an application is a desirable feature from the perspective of optimizing the network-wide resource allocations. Support for this feature in various architectures has been mainly driven by implementation considerations on current network environments and how network mechanisms for multicast have been developed in the past. We shed a new light on this feature from a perspective of the evolving characteristics of multi-service applications.

3.1 Cost of data transport

Let V be the set of nodes in physical topology of the backbone network, and $\{s_i\}_{i=1,2,\ldots,N}, \{d_j\}_{j=1,2,\ldots,M} \subseteq V$ be the nodes containing source and destination entities respectively. For a given placement of $\{s_i\}$ and $\{d_j\}$ in the topology, we examine the cost of multicast paths in various architectures.

'source-rooted tree' (SSRT) architecture

The stream from each source flows over a separate tree that serves all destinations. There is no sharing of the common links across various streams. So the transport cost is given by

$$net\_cost(\{s_i\},\{d_j\}) = \sum_{\forall i} fixed\_cost([tr(s_i,\{d_j\})]) + flow\_cost(s_i,[tr(s_i,\{d_j\})]).$$

The 'Internet Stream Protocol' (ST-II) developed at BBN [8] falls under this architecture.

'core-based tree' (CBT) architecture [9]

The streams from all sources are first collected at a fixed central node, called the 'core'; the combined streams are then sent over a distribution tree rooted at the 'core' and serving all destinations. Here, any node in the tree that is directly reachable from a source can serve as its SMP. The path segments from the 'core' to destinations are shared across streams, but the point-to-point paths from sources to the 'core' are not. Accordingly, the transport cost is given by

$$\begin{aligned} net\_cost(\{s_i\},\{d_j\}) = & \sum_{\forall i}(flow\_cost(s_i,[s_i,\mathrm{core}]) + fixed\_cost([s_i,\mathrm{core}])) + \\ & fixed\_cost([tr(\mathrm{core},\{d_j\})]) + \sum_{\forall i} mux\_flow\_cost(s_i,[tr(\mathrm{core},\{d_j\})]). \end{aligned}$$

'rendezvous-point rooted tree' (PIM) architecture [10]

The streams from all sources are first collected at a fixed set of nodes, called the 'rendezvous points' (RP); the combined streams are then sent over the distribution trees rooted at these RPs and serving each a disjoint subset of destinations. Here, any node in a RP-tree that is directly reachable from a source can serve as its SMP. The path segments from a RP to its subset of destinations are shared across streams, but the point-to-point paths from sources to the RP are not. Accordingly, the transport cost is given by

$$\begin{aligned} net\_cost(\{s_i\},\{d_j\}) = & \Big(\sum_{\forall x\in \mathrm{RP}}\sum_{\forall i}(flow\_cost(s_i,[s_i,x]) + fixed\_cost([s_i,x])) + \\ & fixed\_cost([tr(x,\{d_j(x)\})]) + \sum_{\forall i} mux\_flow\_cost(s_i,[tr(x,\{d_j(x)\})])\ \Big), \end{aligned}$$

where $d_j(x)$ is a destination served by the tree rooted at RP x.

'unrooted tree' (URT) architecture [11, 12]

The streams from sources are collectable at any set of nodes; the combined streams are then sent over the distribution trees rooted at these nodes and serving all destinations. Here, any node in the channel that is directly reachable from a source can serve as its SMP. Both the point-to-point paths from a source to its SMP node and the multipoint paths from this SMP node to destinations are sharable across streams. Accordingly, the transport cost is given by

$$net\_cost(\{s_i\},\{d_j\}) = fixed\_cost([\mathcal{G}]) + \sum_{\forall i}(flow\_cost(s_i,[s_i,\mathrm{SMP}(s_i)]) + mux\_flow\_cost(s_i,[tr(\mathrm{SMP}(s_i),\{d_j\})])),$$

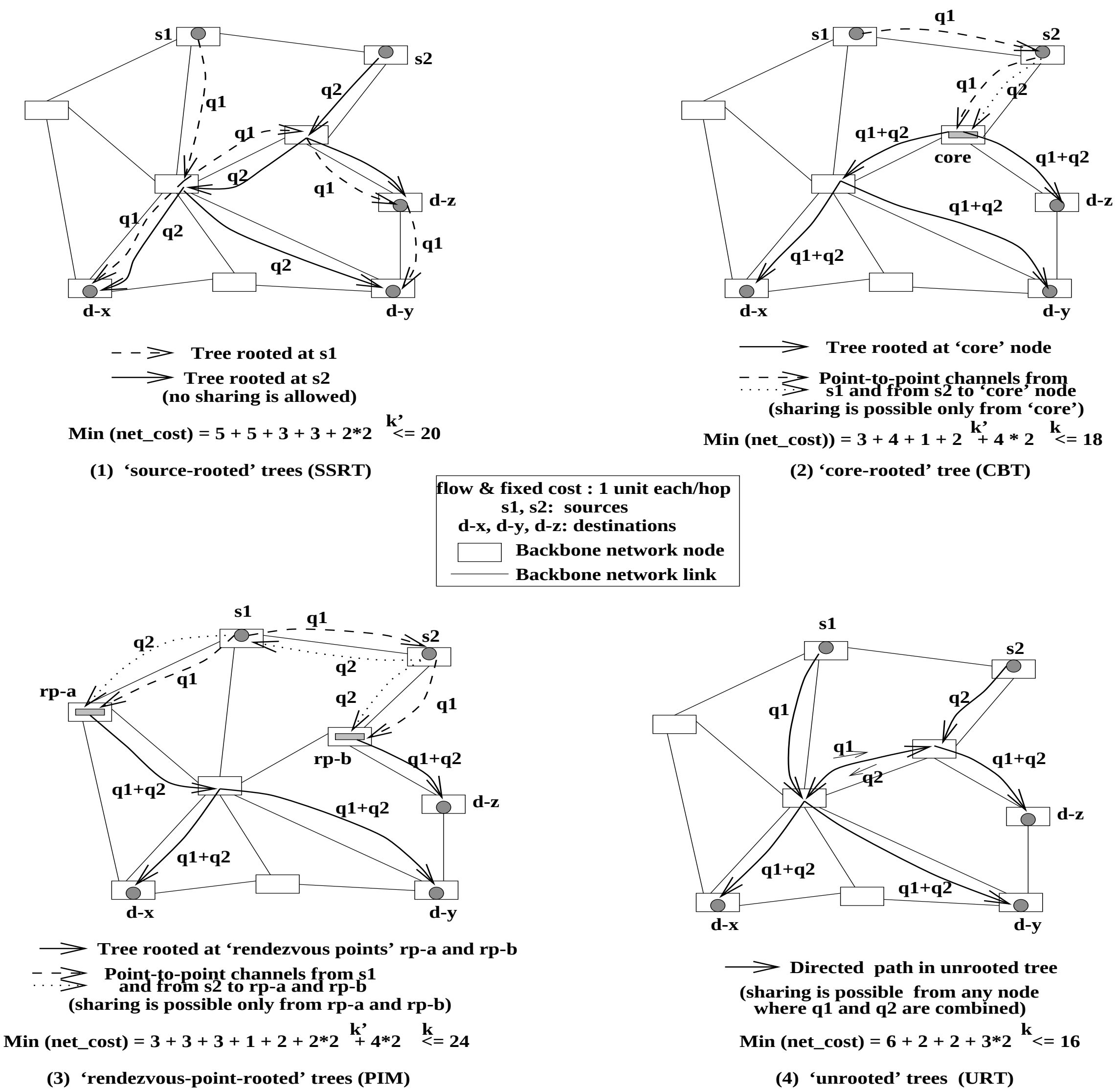

Figure 7: Sample multicast channel configurations in SSRT, CBT, PIM and URT architectures

where $\mathcal{G}$ is an acyclic graph such that $tr(s_i, \{d_j\}) \subseteq \mathcal{G}|_{\forall i}$ and $\mathcal{G} = \bigcup_{\forall i} tr(s_i, \{d_j\})$.

Figure 7 illustrates the multicast paths for a sample source-destination configuration under the SSRT, CBT, PIM and URT architectures each.

3.2 Empirical studies on path costs

We have studied a variety of source-destination configurations in physical topologies to determine the network-wide cost savings due to the sharing of path segments across various data streams. With 'statistical multiplexing' of bursty streams, bandwidth savings of 18-22% are possible across 5 streams with a burst factor of 0.25 each and $\Delta = 2\%$, for a configuration consisting of 8 destinations on a 25-node topology. The savings are higher at about 32% with a burst factor of 0.5 and $\Delta = 5\%$. The savings in fixed cost alone can be as high as 60% in many large sized configurations (say, 5 sources and 15 destinations on a 50-node topology) which, however, may be perceivable only when the flow costs are relatively small.

Since the degree of path sharing is influenced by the multicast architecture employed, a cost comparison of multicast transport under different architectures can reveal the extent of savings in resources possible with respect to the base case where no sharing is allowed, as in SSRT (here, the fixed cost overhead is $\mathcal{O}(N)$). In many cases, the degree of sharing and the total number of hops in a multicast path counter-balance each other. For instance, an attempt to achieve the highest degree of sharing often leads the streams to a single SMP at the 'geographic center' of the configuration for onward flow, which however increases the number of hops the streams need to traverse. Here, the savings in per-hop bandwidth and/or fixed overheads accrued from the increased sharing of path segments can be out-weighed by the increased 'path distance'. Thus determining a cost-optimal multicast path is a computationally complex task.

3.3 Implications on routing algorithms

The sharing of distribution paths across multiple streams casts itself upon routing algorithms in the form of: i) constraints on the choice of nodes in a distribution path, ii) the extent of routing control actions required during path setup, and iii) adaptivity to dynamic changes in source-destination configuration and flow parameters. The actual resource consumptions and cost incurred for multicast paths depend on the flow specifications and the source-destination configuration in physical topology.

In SSRT model, each source-rooted tree can be individually minimized, but these non-sharable trees together do not always yield minimality for the entire source-destination configuration. In CBT model, cost minimization of the 'core-rooted' tree and the non-sharable point-to-point paths from various sources to 'core' node together need not be the best. This is because of the need to send all data to the 'core', which limits the 'spread-out' of channel topology towards paths of lower cost. In Figure 7 for instance, an optimal placement of 'core' node for the given sources and destinations may not yield cost minimality any more when, say, one of the destinations leaves or a new source/destination joins the configuration. A similar argument holds for PIM model with respect to 'rendezvous-point' nodes. With URT model, the trees projected at various sources can be analyzed together for shared path segments, and can be 'spread out' with less constraints towards lower cost paths. Thus the cost minimality achievable in various models can be different[2].

[2] Another example of analogy is in place here. Consider passengers flying from San Fransisco to San Diego. Assume that there is a flight from Chicago to San Diego enroute Denver and Los Angeles, and that any flight to San Diego needs to stop in Los Angeles. With CBT model, the rules stipulate that passengers can change flights only at a single hub, say Denver (for discussion purpose, we treat PIM model to be the same as CBT model). With URT model, the rules allow passengers to change

The message space required for SSRT model is the least, since the same message types may be used for setting up each tree and a destination will connect to a tree only if it needs to receive the stream(s) flowing over this tree. In comparison, the URT model requires a larger message space, due to the need to install stream multiplexors at various nodes in shared path segments. The CBT and PIM models require a still larger message space due to the need to set up the point-to-point paths from sources to designated root nodes, in addition to setting up shared trees from root nodes with stream multiplexors.

See [12, 13] for detailed comparison of various architectures from transport cost and routing algorithm perspectives. Our treatise on network support for shared distribution paths allows one to determine how far the application-level flow/QOS specifications can be taken down into the network to influence its behavior (i.e., 'network programmability') and the tradeoffs involved therein.

The multi-source broadcasting of data are cast into network level functional elements for use by routing algorithms, as described in the next section.

4 Network functional elements

In a canonical form, multicast support in the network requires flow/QOS based functional elements that can be appropriately exercised for resource allocation and routing control purposes.

4.1 Switch level resource allocator

A switch S may employ a functional relation $\mathcal{F}$ that maps a flow rate to resources, viz., link bandwidth needed to support the flow rate and switch buffers needed for sending and/or receiving data through a port of at this rate. S may realize $\mathcal{F}$ in the form of pre-computed tables that relate the number of streams multiplexed, their average flow rates and burstiness, and the application-wide parameter Δ to the bandwidth/buffer needs. See Figure 8 for an illustration.

A specific implementation of these resources, such as the processing cycles consumed in S to support a flow, is itself a switch internal issue that does not concern a routing control protocol. However, the quantification of resource needs for a given flow has a global scope so that the network-wide cost of a path through S can be estimated. In other words, S should support a quantifiable abstraction of resources in order to interwork with

flights at any hub, whereby passenger routing may assign the city of Los Angeles, the nearest city to San Fransisco, for flight change and continuation to San Diego. With SSRT model, the rules require passengers to use a separate flight all the way to San Diego via Los Angeles. Here, both URT and CBT models allow the seating capacity of Chicago-SanDiego flight to be shared with passengers from San Fransisco; however, CBT model incurs increased 'flight distance' by forcing a travel to Denver. SSRT model, though incurs the same 'flight distance' as URT model, does not allow sharing the seating capacity of Chicago-SanDiego flight.

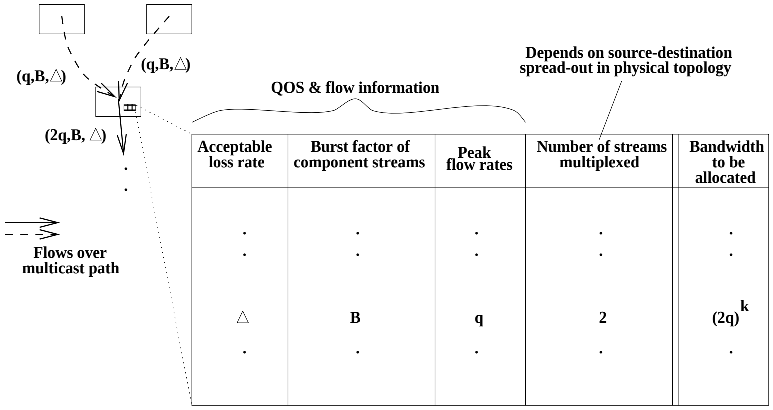

Figure 8: Illustration of flow-to-resource mapping tables in nodes

a routing algorithm. The de-lineation of estimating the resource needs for a flow and a local implementation of resources in switches should be guaranteed by the BNL.

For this purpose, a backbone network may implement a convergence sublayer to export a global view of resources, hiding their local implementation. The global view is then integratable into a MCL function.

4.2 Switch level QOS controller

With a 'multicast tree' sharable by more than one data stream (c.f. section 2.4), the ability to distinguish the various data streams may still be needed in the network to handle the QOS related transport attributes (say, to map the 'loss sensitivity' and 'acceptable delay jitter' of various data to scheduling priorities on data). For this purpose, the various flow related parameters specified by users are keyed on stream ids.

An important QOS parameter is the acceptable end-to-end delay of data units, i.e., the maximum allowable delay between when a data unit is generated by a source and when it is consumed by a destination. Since each hop in a path segment can induce a certain amount of data transfer delay, the end-to-end delay constraint may influence the setting up of tree connecting various destinations. It is also likely that destinations will be able to determine the level of acceptable end-to-end delay on a data stream. In a multimedia conferencing application involving audio and video streams for example, an interactive participant may specify 50 *msec* as the end-to-end delay, while a passive participant may specify 250 *msec*. This receiver-specific delay control allows more flexibility in path selections by a routing algorithm.

Each node u of a tree channel maintains information on the cumulative delays incurred by the various streams $\{s\}$ flowing through u, denoted as $\{str\_del(s,u)\}$. For this purpose, u may obtain the hop delay information (from BNL) for its incoming path segments from various upstream neighbors (there can be more than one upstream neighbor in URT architecture and exactly one upstream neighbor in CBT/PIM/SSRT

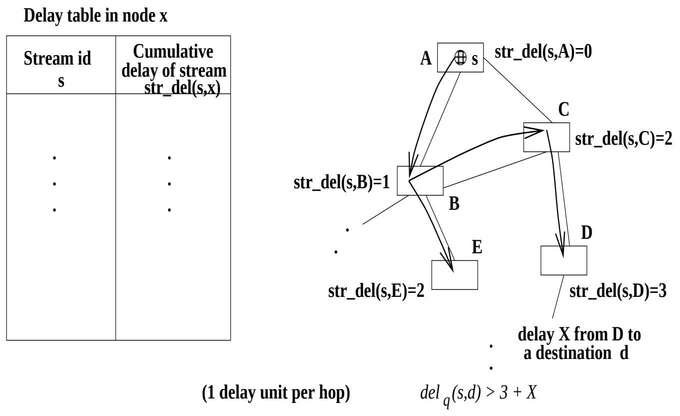

Figure 9: Illustration of end-to-end delay control information in switches

architectures). The routing algorithm may connect a destination d to a source s through node u in the tree channel such that the sum of delays from s to u and from u to d is less than the prescribed end-to-end delay limit $del_q(s, d)$ (the cost of path through u should also be a minimum in comparison to other paths)[3]. See Figure 9 for an illustration.

4.3 Stream filters

A source-selective receipt of data by destinations (e.g., an audio-only-capable terminal participating in a video+audio conference session) may be supported in the network by filtering of data streams at various nodes of a shared tree. The filter prevents a stream from flowing along path segments that lead only to destinations not requiring this stream, thereby saving resources along these path segments. This functional element is somewhat similar to the 'upward pruning of distribution trees' proposed in RSVP [14].

A destination d which *does not* wish to receive data from $s_1, s_2, \cdots, s_m|_{m \leq N}$, includes the list of stream (id,rate) pairs $\{(s_j, q_j)\}_{j=1,2,\ldots,m}$, referred to as 'source mask' $src\_msk(d)$, as part of the flow attributes it specifies (N is the number of sources in the application). The network splits the combined data stream flowing through the tree at an appropriate node u to isolate a stream not specified in src_msk for downward flow from u towards d. This is done with a filter installed along a path e of u through which d is reachable. The filter maintains a mask list $k\_lst(u, e) = \{(s_j, q_j)\}$, and forwards only the data units carrying a stream id not listed in $k\_lst(u, e)$ through e (when no filtering is required, $k\_lst(u, e) = \emptyset$).

Given that d does not wish to receive data from a source s_i, the network may install a filter at each node in the upstream path segment up to the node that roots a subtree with more than one branch and the data is

[3] We expect that destination entities will engage in an end-to-end protocol with source entities to determine reasonable lower/upper bounds on acceptable end-to-end delays for specification to the network in 'connect' requests.

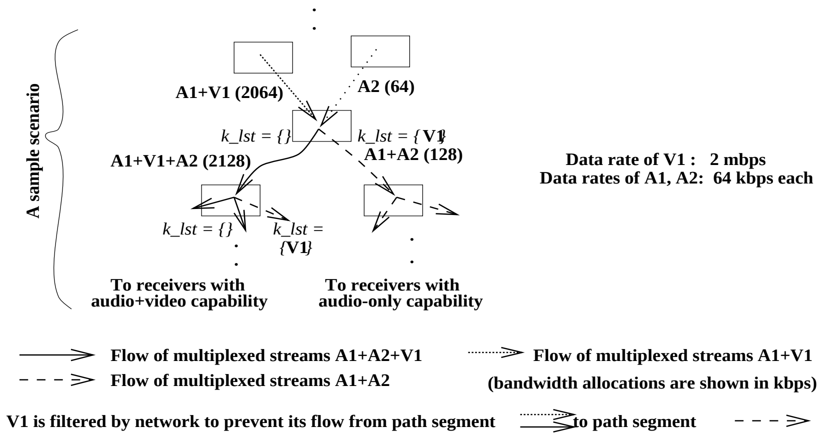

Figure 10: Illustration of network supported filters of data streams

needed along at least one of these other branches. The filter installation also involves de-allocating resources to the extent of $\mathcal{F}(q_i)$ at each node in the upstream path segment. Thus the flow from s_i is ON if at least one destination wishes to receive from s_i. When none of the destinations wish to receive from s_i, the filter mechanism automatically turns OFF s_i. See Figure 10 for illustration.

The functional elements described in this section are common across the SSRT, CBT, PIM and URT architectures (it is beyond the scope of this paper to dwell on architecture-specific extensions to these functions).

We now describe how routing algorithms exercise these functional elements when connecting/disconnecting user entities to/from a multicast channel.

5 Design of Multicast routing algorithms

The protocol support for multicast consists of decentralized information structures maintained at various nodes to support network functional elements and the control message space required to exchange this information across nodes. This section presents a high level view of routing algorithms in terms of this protocol support.

5.1 Functional decomposition of routing algorithms

A multicast routing algorithm may be viewed as consisting of the following components (see Figure 11):

- Mapping of user-level flow specifications $\mathcal{R}$ to resource demands at a node/link;
- Determining the network-wide resource needs of various interconnection paths for a given source-destination configuration $\mathcal{C}$;
- Optimizing a cost index Φ in selecting the data paths;

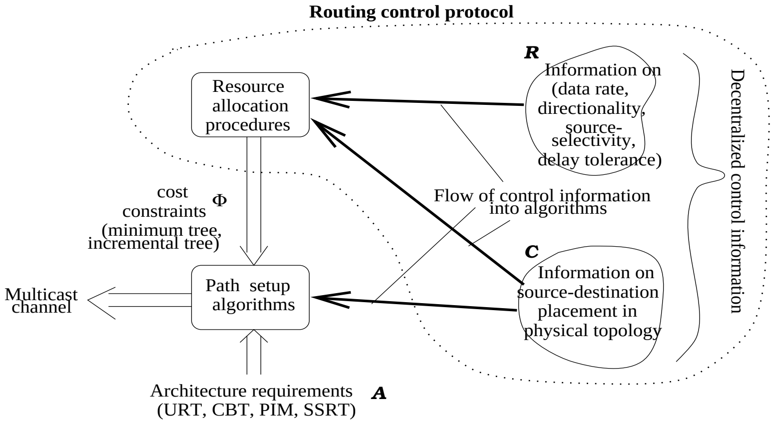

Figure 11: Functional components of multicast routing algorithms

- Enforcing the architectural constraints $\mathcal{A}$ on path selection.

Thus a routing algorithm is representable as $\Phi(\mathcal{F}(\mathcal{R}), \mathcal{C}, \mathcal{A})$. In fact, many existing routing algorithms (such as DVMRP [15]) can, in retrospect, be viewed in terms of such a functional decomposition.

A routing algorithm obtains global knowledge of: i) data flows network-wide from various sources to destinations (i.e., $\mathcal{R}$) based on user level specifications of data directionality, data rates and source-selectivity, and ii) the configuration information (i.e., $\mathcal{C}$) based on relative placement of source and destination entities in physical topology of backbone network. With the above information, the costs of various possible data paths between sources and destinations under the chosen architecture (i.e., $\mathcal{A}$) may be computed so that a path meeting a cost and delay constraint (i.e., Φ) is determined. In the presence of dynamic changes in the configuration, Φ may refer to, for instance, a 'network-wide minimal cost' path across every change and a 'incrementally minimal cost' path at each change.

See [4, 16, 17] for representative cost-based routing methods. The notion of 'geographic spread' based cost minimization proposed in [18] seems to be appropriate. However, it needs to be augmented with the data flow characteristics for effective use in multi-service networks. Consider, for instance, 2 sources s_1 and s_2 in a configuration. A case where high bandwidth video data flows from s_1 and low bandwidth audio data flows from s_2 is different from a case where only audio data flows from s_1 and s_2 each. In the former case, the paths from s_1 and s_2 to destinations are likely to be accentuated more towards connecting s_1 to destinations with less number of hops in the physical topology, while in the latter case, these paths are likely to be evenly spread out in the topology.

5.2 End-to-end delay constraints

The architecture specification $\mathcal{A}$ may also influence the setting up of paths with end-to-end delay guarantees. This is because an architecture restricts the selectable set of paths to only a subset of paths feasible in the physical topology.

Consider a destination d joining a channel configuration connecting to sources s_1 and s_2. In CBT architecture, the delay condition is:

$$str\_del(x, \text{core}) + str\_del(\text{core}, d) \leq del_q(x, d)|_{x=s_1,s_2}.$$

In PIM architecture, the 'core' is replaced by a RP node[4]. Suppose the above condition does not hold. With CBT architecture, d cannot connect to the channel. The PIM architecture however allows d to bypass the RP node and set up a source-rooted tree to s_1 and s_2 each that meets the delay condition (thus, the RP-tree and source-rooted trees coexist). Since the URT architecture does not pre-designate any particular node to serve as SMP, it is possible for d to change the choice of SMP node for s_1 and s_2 such that the delay condition can be met. This reduces the probability of connection failures.

In a modification to the CBT architecture [19], the 'core' can be dynamically split into many 'core' nodes if the path to one or more destinations from the current 'core' node does not meet the delay condition. The topological placement of such 'dynamic core' nodes needs to be carefully done because, in some extreme cases, the join of each destination can cause a re-assignment of all the 'core' nodes.

We now describe a protocol model that allows systematic acquisition and use of source-destination configuration information by a routing algorithm to optimally exercise network-wide resource allocation control.

5.3 Routing control protocol

The functional view of routing algorithms underscores a canonical protocol model that defines the decentralized flow-related information structures $\mathcal{R}$ maintained by 'agents' executing at various nodes in the connectivity configuration $\mathcal{C}$ and the routing control messages exchanged between these 'agents' to access the flow information. See Figure 12.

The messages propagate user-level flow specification and source-destination configuration information to the 'agents' in various nodes. The messages also carry information on resource allocations at various nodes/links and the overhead and delay incurred on the 'native connections' over these nodes/links. Representative types of messages include:

[4] Here, the 'core'/RP node acts as a source agent.

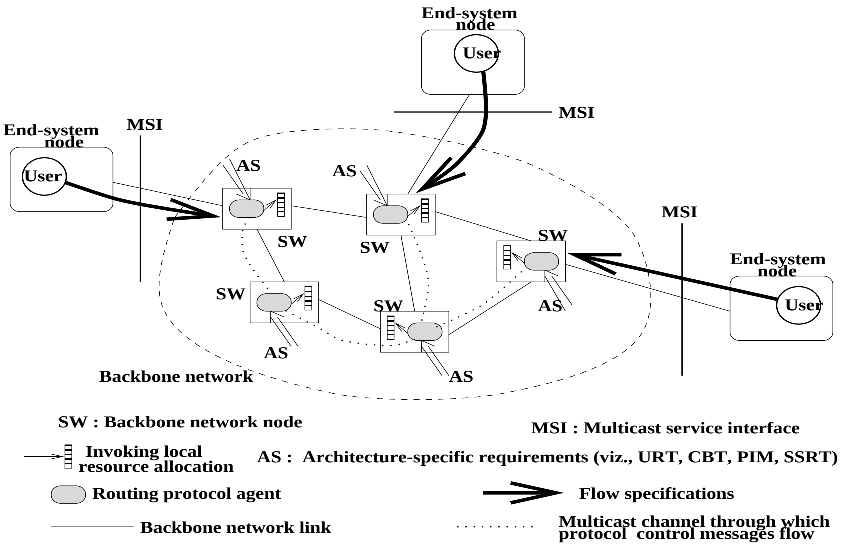

Figure 12: Protocol model for multicast path set up

- 'search' messages to locate SMP nodes that can support the required flow from/to user;
- 'resource check' messages to ascertain the availability of resources and the end-to-end delay guarantees along downward path segments to support a flow;
- 'path setup' messages to allocate resources in nodes along a path through a given node;
- 'stream tap' messages to locate a stream flowing along upward paths and bring it to a given node;
- 'path tear down' messages to de-allocate resources along chosen path segments;
- 'stream remove' messages to delete a stream flowing through a node and along its upward path segment up to a point where the stream needs to flow along other branches of the tree.

During execution, the protocol may root a tree at any chosen node to propagate these control messages to all other nodes in the path segments.

The protocol model is largely independent of the underlying network architecture employed. Architecture-specific extensions (viz., for SSRT, CBT, PIM and URT) can be incrementally added in the target implementation of a routing algorithm. Details of the protocol model in terms of canonical information structures maintained at 'agent' nodes and the control message flows across nodes to access these information may be found in [20]. The model is closely related to the 'Resource ReserVation Protocol' (RSVP) [14] and the 'Internet Stream Protocol' (ST-II) [8] in light of their scope towards multi-service applications.

A routing algorithm so constructed can explicitly factor in the diverse transport attributes of multi-service data (such as flow rates, burstiness and tolerable delays) and the source-destination configuration for determining the data paths.

6 An implementation case study

We have studied the functional model of multicast routing algorithms on an ATM network consisting of 5 nodes (Fore and GTE switches) and 3 SUN-Sparc workstations. These workstations implement the source and destination entities. Two distinct application configurations are studied: one with 4 streams (2 video and 2 audio) and 4 destinations spread out across 3 workstations and the other with 4 streams (2 text and 2 graphics) and 2 destinations spread out across 2 workstations.

Since the native ATM switches do not support our functional model, we placed the protocol agents in 'logically extended switches' (LES), implemented on (additional) workstations attached to native ATM switches. The VP/VC set up and VP level bandwidth allocation functions provided in the native ATM switches are available to the agents in LES through a programming interface. The physical topology of backbone network is basically a set of VP links interconnecting the various switches. Data flow paths and control message paths are realized through separate VCs multiplexed on common VP links (VP link is a unidirectional channel between a pair of ATM switches).

The algorithm employs 'incremental path setup' as the cost constraint, and incorporates the architectural requirements of SSRT, CBT, PIM or URT. The algorithm generates paths in each of these architectures, specified in terms of bandwidth needs on each VP link and the filters placed along each VP link. Stream delay information is also generated by the algorithm at each LES in a path[5].

The logical topology generated by the algorithm is then embedded on to the ATM network VC/VPs as follows. The estimated bandwidth needs for a VP link are transcribed into the native switch bandwidth allocation for VP links. A distinct VC is assigned to each stream in the application, and multiple VCs share one or more common VP links, as allowed by the architecture[6]. A stream filter along an outgoing VP link is realized simply by not including the corresponding VCI-VPI mapping entry in the switch tables for this VP link. A bidirectional path segment is realized with 2 unidirectional VC/VPs since the native ATM switches support only unidirectional channels.

The total number of distinct VP links required network-wide to support both the application configurations

[5] Stream delay information is of less consequence in this small testbed under study, except for validating the algorithm functionality of enforcing delay control.

[6] Bandwidth savings due to link sharing across bursty flows could not be determined from this testbed, since the current version of ATM switches allocate the entire bandwidth for streams in the absence of contention and we could not generate enough traffic to saturate the switch capacity of 625 *mbps*. However, a separate study using ATM 'cell traffic analyzer' equipments confirm that bandwidth savings are possible due to 'statistical interleaving' of bursty streams.

(fixed cost overhead) under the URT, SSRT, CBT and PIM architectures are found to be 20, 36, 28, and 30 respectively. As a qualitative note, our functional model of routing algorithms allowed an easier realization of the above architectures and cost constraint. In fact, that it would have been impossible to implement multicast routing for multi-service networks with a traditional 'monolithic' view of routing algorithms is not an over-statement.

7 Conclusions

The paper identified the canonical elements of multicast transport architectures for multi-service applications. These requirements arise because of the diversity in transport characteristics of applications such as multi-source broadcasting of data to a common set of destinations, large volumes of bidirectional and unidirectional data transfers among users, and widely varying transfer rates of data from sources (e.g., video conferencing, broadcast audio, graphics image distribution).

The diversity in transport requirements of applications necessitates a fine granular control over the allocation of resources in network channels to support a given data flow. The finer control manifests in the form of a tighter linkage between the user specifications of data flow and the network resource allocations. This in turn necessitates a 'programmable network' as the underlying theme, wherein applications may specify their transport requirements at a finer granularity so that the network can determine its behavior to suit the needs of applications. Thus, multiple instances of the network may co-exist at the same time, with each instance parameterized to suit the needs of an individual application.

From the above perspective, the paper described the main elements of multicast transport architectures. An architecture typically employs tree-structured channels for data distribution. We believe that a transport architecture should be *network-centric* in that the user-network interface allows casting the needs of applications onto network internal mechanisms. In such a model, the user-level flow specifications are transcribed to the network for the latter to determine a cost-optimal data distribution path. The architectural elements align well with the evolving functionalities in the 'Internet Service Layer' model.

Incorporating flow/QOS-based control of network resource allocations and path set ups, the paper described canonical models/architectures for multicasting. In terms of these models, a multicast routing algorithm can be functionally decomposed into components dealing with the decentralized manipulation of flow/QOS information, path set ups to interconnect sources and destinations placed in physical topology of the backbone network, and architectural constraints on path selections. The functional view allows simplifying the validation and implementation of routing algorithms in light of the complex requirements of multi-service applications.

The paper also described how the models can be mapped to operational backbone networks (such as ATM networks and interconnected LANs). However, the model elements are themselves independent of target network specifics. The paper also evaluated the effectiveness of currently available transport architectures in supporting multicast-based applications (such as multimedia conferencing).

Overall, we agree with the current thinking in the Internet research community that an open-ended service model is the way to supporting the evolving multicast applications. We believe that the specific architectural elements described in the paper provide a concretization of this view point in terms of network protocol and service support functions.

Acknowledgement

The author acknowledges **Mr. Ting-Jian Gong** of Kansas State University (KSU) for valuable discussions on network architecture issues and cost analysis of architectures, and **Mr. Kenneth Gould** of KSU for the support work in setting up ATM network testbed and in conducting ATM network traffic studies.

References

[1] M. Macedonia and D. Brutzman. **MBone Provides Audio and Video Across the Internet**. In *IEEE Computer*, vol.27, no.4, pp.30-36, April 1994.

[2] S. Casner and S. Deering. **First IETF Internet Audiocast**. In *Computer Communication Review*, ACM SIGCOMM, vol.22, no.23, July 1992.

[3] 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.

[4] 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.

[5] S. Shenker. **Fundamental Design Issues for the Future Internet**. In *IEEE Journal on Selected Areas in Communications*, Special Issue on *Global Internets*, vol.13, no.7, pp.1176-1188, Sept. 1995.

[6] J. N. Chiappa. **Evolutionary Possibilities for the Internetwork Layer**. In Part IV, *IPng: Internet Protocol Next Generation*, Addison-Wesley Publ. Co., 1996.

[7] D. D. Clark. Foreward. *IPng: Internet Protocol Next Generation*, Addison-Wesley Publ. Co., 1996.

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

[9] T. Ballardie, P. Francis and J. Crowcraft. **Core Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing**. In Proc. *Comm. Architectures, Protocols and Applications*, ACM SIGCOMM, pp.85-95, Sept. 1993.

[10] S. Deering, D. Estrin, D. Farinacci and V. Jacobson. **An Architecture for Wide-Area Multicast Routing**. In Proc. *Comm. Architectures, Protocols and Applications*, ACM SIGCOMM, Sept. 1994.

[11] K. Ravindran. **A Flexible Network Architecture for Data Multicasting in High Speed 'Multi-service Networks'**. In *IEEE Journal on Selected Areas in Communications*, Special Issue on *Global Internets*, vol.13, no.8, pp.1426-1444, Oct. 1995.

[12] K. Ravindran. **Network Abstractions and Resource Allocation Models for Data Multicasting in 'Multi-service Networks'**. In *Tech. Report 93-7*, Dept. of Computing and Information Sciences, Kansas State University, Aug. 1995.

[13] T. J. Gong and K. Ravindran. **Cost Analysis of Data Transport Architectures for Multicasting in 'Multi-service Networks'**. In *Tech. Report*, Dept. of Computing and Information Sciences, Kansas State University, Mar. 1995.

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

[15] D. Waitzman, et al. **Distance Vector Multicast Routing Protocol** In RFC 1075, 1988.

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

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

[18] J. Kadirire. **Minimizing Packet Copies in Multicasting by Exploiting Geographic Spread**. In *Computer Communication Review*, ACM SIGCOMM, vol.24, no.3, pp.47-62, July 1994.

[19] Y. C. Chang, Z. Y. Shae, and M. H. W. LeMair. **Multiparty Video Conferencing using IP Multicast**. In *IS & E / SPIE Symp. on Electronic Imaging: Science and Technology*, 1996.

[20] K. Ravindran and T. J. Gong. **Flow and QOS based Routing Control Protocols for Data Multicasting in 'Multi-service Networks'**. In *Tech. Report*, Dept. of Computing and Information Sciences, Kansas State University, Jan. 1996.