Framework for Multicast in Hierarchical Networks

Reuven Cohen Eyal Felstaine
Dept. of Computer Science
Technion, Haifa 32000, Israel
{rcohen,eyalfe}@cs.technion.ac.il

Roy Emek
IBM Haifa Research Laboratory
MATAM, Haifa 31905, Israel
emek@il.ibm.com

Abstract—**We propose a framework for the creation and maintenance of multicast trees in hierarchical ATM networks. This framework aims at coping with an inherent difficulty of topology aggregating in such networks. The main idea of the proposed framework is to distribute the tree topology information among a set of hierarchical Multicast Group Servers (MGSs) nominated for each multicast tree, while keeping regions that do not have a member in the multicast group unaware of the tree. The framework can be employed with every multicast routing algorithm designed for non-hierarchical networks.**

Keywords: multicast, hierarchical networks, PNNI, ATM networks, NIMROD, scalable routing.

I. Introduction

This paper deals with multicast trees in hierarchical networks. Such a network consists of a hierarchy of subnetworks called domains [1]. To allow scalability, domains do not reveal details of their internal structure to nodes in external domains. Instead, every domain advertises only a summary, or aggregated, view of its internal structure. This is the concept adopted by the ATM Forum as the PNNI (Private Network to Network Interface) standard for hierarchical ATM networks [2].

In PNNI, every ATM switch is considered as a lower level node. Such a node maintains detailed topology information about the lowest cluster to which it belongs, namely about its Peer Group (PG). Such a PG usually contains a small number of switches and links. Hence, maintaining the detailed information for the PG does not lay an excessive communication, storage and processing burden on the PG nodes. In contrast, every node maintains only compressed topology information for its parent PG, because the latter contains a larger number of switches and links. Similarly, the node stores more aggressively compressed topology information about its "grandparent PG", and so forth. A detailed description of the PNNI model is presented in Section II.

In a communication network multicast is treated as the problem of creating, maintaining and updating efficient forwarding trees, rooted at the multicast source nodes and spanning the groups of destination nodes. A multicast tree may be dynamically created, as network nodes join and leave the destination group. It is maintained by some of the network nodes, mainly those sitting on its data path [3], [4]. It needs to be updated not only following changes in the multicast group but also following changes in the network connectivity and load. In a virtual circuit switched networks like ATM, the multicast tree can be represented by a single tree VC (virtual channel).

The computation of an efficient multicast tree required to maximize usage of network resources is often modeled by the Minimal Steiner Tree (MST) problem in graphs [5], [6]: Given a weighted graph with a non-negative weight for each edge and a subset of nodes called terminals, find a minimum cost connected subgraph that covers all terminals. An even more general problem in the context of network multicast is the *dynamic* MST problem [7], where the tree needs to efficiently adapt to dynamic changes in the multicast group. The basic MST problem is known to be NP-Complete, and heuristics for the problem have been proposed in the past [8], [9]. However, all the heuristics assume that the exact topology and location of the multicast group members are known, which is not the case in hierarchical networks.

In this paper we propose a framework for the adaptation of multicast routing protocols to hierarchical networks. In the context of this framework we address the various issues related to multicast that arise in hierarchical networks, like the distribution of information about the status of the tree, the distribution of information about layout of the multicast group members, and so forth. The proposed framework, refered to as HMF (Hierarchical Multicast Framework), deals with dynamic groups, whose members may join and leave at any time. The paper focuses on the ATM PNNI model which gains great interest due to its acceptance as a standard by the ATM forum. Nevertheless, HMF can be adopted to other hierarchical, packet switched (IP) or circuit switched, network models.

A complete *scalable* mechanism that constructs *efficient* multicast trees using *reasonably compact data structure* as well as *acceptable setup time and communication complexity*, was never presented for hierarchical networks. Each of the existing schemes (which are discussed in Section III) has at least one severe disadvantage with respect to either Memory storage requirements, Tree size efficiency or Protocol execution load. The proposed framework addresses all these disadvantages and does not impose "new" ones.

As already indicated, one can use HMF in order to implement multicast routing algorithm that have been originally designed for flat networks [5], [10], [11] in a hierarchical environment. Other properties of HMF are as follows:

1. It scales to very large networks. This is mainly because the multicast-related activities are performed distributedly, and only in those areas that have switches sitting on the tree data path. Moreover, HMF allows to evenly distribute the multicast-related overhead among many nodes in the network.
2. HMF generates an efficient multicast tree, that is comparable to the trees generated by the same multicast algorithms in flat networks where topology information is not hidden.
3. The setup time required for joining and leaving a multicast tree is comparable to the setup time of a regular VC.
4. HMF supports the construction of multicast trees according to the rules dictated by the multicast algorithm. For instance, the tree can be generated based on QoS constraints, given a QoS-based multicast routing algorithm for a flat network. To un-

derstand the importance of this property, consider a scheme for hierarchical multicast routing, according to which a node that needs to join to a tree is informed of the identity of a switch to which it needs to create a virtual circuit. In such a case, the route from the joining node to the switch on the tree will be determined by the underlying unicast routing algorithm. In contrast, our scheme specifies not only the switch to which the joining node needs to connect, but also a description of an hierarchical route over which the circuit is to be established as determined by the multicast algorithm. This feature allows the multicast algorithm to create a tree which cannot be created if our scheme had relied upon the unicast QoS-based routing algorithm.

The rest of this paper is organized as follows. Section II outlines the main concepts of PNNI network organization and unicast routing. Section III describes the main issues in hierarchical multicast routing, and discusses previous related works. In Section IV we present the HMF. Section V presents a complexity analysis of the HMF properties and demonstrates the efficiency of the constructed trees. Section VI concludes the paper.

II. An Overview of ATM PNNI Routing

In computer networks, routing is a collection of algorithms that determine the routes that data packets will traverse until reaching their destination nodes. In order to make routing decisions, the network nodes should obtain topology information and maintain routing tables. There are two well-known approaches to perform this task distributedly. In the first approach, called *distance-vector routing* [12], each node sends its *neighboring nodes* its *entire* routing table. The receiving nodes use the received information in order to update their own routing tables, which they then send to their own neighbors. In the second approach, called *link-state routing* [12], each node broadcasts information to *all network nodes* regarding the status of its *local links only.* The network nodes use the received information in order to create and maintain an up-to-date network map, from which they deduce their routing tables.

The main drawback of the distance-vector algorithm is that it takes a long time to reconverge to alternate paths when a failure occurs in the network [12]. During that time, the routing tables may define loops that may cause congestion in the network. The link-state protocol responds much faster to topology changes. However, in large networks it lays an excessive communication, storage and processing burden on the nodes. The concept of hierarchical routing introduced by [13] is usually employed in order to overcome this limitation of link state routing. The idea of using hierarchical routing in order to avoid the excessive complexity in topology advertisement is discussed in [14].

According to PNNI, the network nodes and links are organized hierarchically. At the lowest level of the hierarchy, each node represents an ATM switch and each link represents a physical link or an ATM virtual path (VP). The nodes and links of each level can be recursively aggregated into higher levels, such that a high-level node represents a collection of one or more lower level nodes, and a high-level link represents a collection of one or more lower level links. The OSPF protocol [12], used for autonomous system routing in the Internet, has two levels of hierarchy. The ATM PNNI (Private Network-to-Network Interface)[2] is a hierarchical, dynamic link-state routing protocol,

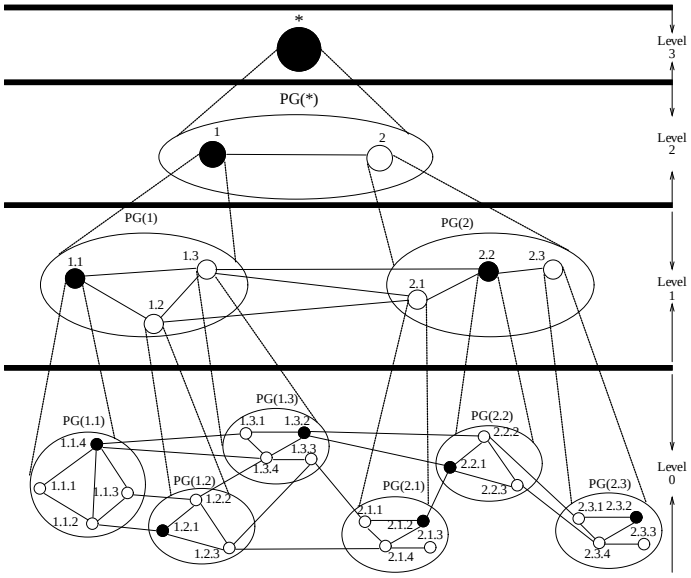

Fig. 1. An example for ATM PNNI hierarchy.

designed to scale to the largest possible ATM networks, encompassing thousands of switches. It may support therefore a maximum of 105 hierarchy levels.

ATM PNNI uses hierarchical link-state routing. To support this hierarchy, PNNI defines a uniform network model at each level, with a set of logical nodes connected by logical links. Each lowest level node represents a physical ATM switch with a unique ATM address. Nodes within a given level are grouped into sets of peer groups. A peer group (PG) is a collection of logical nodes that exchange link-state messages, called PTSPs (PNNI Topology State Packets), with other members of the group, such that all members maintain an identical view of the group. Each PG is assigned a unique identifier and is represented in its parent PG as a single node, called *logical group node (LGN).*

Figure 1 shows an example of ATM PNNI hierarchy. In this figure, a logical node whose identity is of the form $x.y.z$ represents a physical node (ATM switch). Nodes $1.1.1 - 1.1.4$ are grouped into a PG called PG(1.1), and, in the same way additional 5 PGs are created. Hence the network has 22 logical nodes in level 0 and 6 logical nodes in level 1. In level 2 there exist only two logical nodes: PG(1), which consists of LGNs (1.1)-(1.3), and PG(2), which consists of LGNs (2.1)-(2.2). The upper level always has only one logical node (LGN $*$ in Figure 1). The links between the physical (level 1) nodes, like (1.1.1,1.1.4) or (1.3.2,2.2.1), represent physical links or virtual path (VP) connections. By contrast, the links between logical nodes in level 2 or above, like (1.1,1.3) or (1,2), represent a *set* of physical links and VP connections.

As already indicated, each PG is represented in the next hierarchical level by a single LGN. The functions associated with an LGN are actually performed by the *peer group leader (PGL).* The PGL is determined by means of a distributed election process, executed by all the nodes in the PG (See [2] for more details). In Figure 1, the PGLs are those LGNs represented by

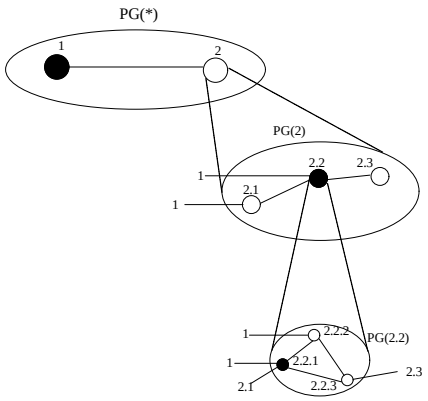

Fig. 2. Local view of the network from nodes in PG(2.2).

black circles. Note that physical node 1.1.4 functions as a PGL of PG(1.1). However, since LGN 1.1 is the PGL of PG(1) and LGN 1 is the PGL of PG$(*)$, the functions that LGNs 1.1 and 1 are supposed to perform as PGLs are actually implemented by physical node 1.1.4.

Topology information flows horizontally through a peer group and downward in to and through child peer groups. A PGL within each peer group has the responsibility of creating PTSPs that represent the status of the links connecting its PG with other PGs, and broadcasting these PTSPs to the other LGNs in its parent PG. In Figure 1, LGN 1.1.4 is the PGL of PG(1.1), LGN 1.2.1 is the PGL of PG(1.2), and LGN 1.3.2 is the PGL of PG(1.3). Hence, these three LGNs exchange PTSPs in PG(1). The PTSPs created by 1.1.4 indicates the status of the links connecting PG(1.1) with other PGs: (1.1.4,1.3.1), (1.1.4,1.3.4), (1.1.3, 1.2.2) and (1.1.2,1.2.1). This PTSP exchange may be thought of as the PGL feeding information up the hierarchy. It is necessary for creating the hierarchy and for distributing routing information about child peer groups. Conversely, feeding information down the hierarchy is necessary to allow nodes in the lower level PGs to obtain knowledge about the full network hierarchy in order to select routes to destinations. When information is fed down from one level to a lower level, it is aggregated (summarized) [15], [16], [17], [18]. Hence, at the lowest level each node has full information about its peer group, aggregated information about its parent group, more aggregated information about its parent's parent group and so on.

By exchanging topology information among nodes, each node obtains the information needed to create its "view of the world". Figure 2 depicts the local view obtained by the nodes in PG(2.2) of Figure 1. The solid lines at PG(2) and PG(*) represent aggregated logical links, like (2.1,2.2), or physical links, like (2.2.1,2.2.2), inside peer groups.

The ATM PNNI is supposed to provide sophisticated QoS routing [19] while still allowing flexibility in the choice of route computation. Hence, each implementation is free to use its own algorithm. This gives rise to source routing, which does not require different switches to agree on the same computation. The switch of the source host creates a hierarchical route consisting of a detailed path within the source node's PG, a less detailed path within the source node's parent PG and so on until reaching the lowest level PG which is an ancestor PG to both the source and destination nodes. When the LGN that contains the destination node in the lowest level common ancestor is reached, a new "source" route is computed to descend to the final destination. A "source" route is also computed when necessary by a node which is the first of its PG along a certain path (a border node). Such a node determines the best way to cross its PG.

A path is encoded as a set of Destination Transit Lists (DTLs), which is explicitly included in a stack within the PNNI signaling call setup request. Each DTL contains the description of a path for one level in the hierarchy. It explicitly specifies every LGN, and optionally every link, used to cross the PG. Each DTL is associated with a pointer that indicates the next element in the list to be processed.

As an example, assume a VC connection is to be set up between a host of 2.2.3 and a host of 1.1.2 in Figure 1. Node 2.2.3 examines its local view of the topology (Figure 2), and finds three possible paths to reach node 1, which is an LGN in the lowest level ancestor PG it shares with 1.1.2: (2.2.3,2.2.2,1), (2.2.3,2.2.1,2.1,1) and (2.2.3, 2.2.1,1). Suppose that on the basis of metrics and policy, node 2.2.3 chooses the last option: (2.2.3, 2.2.1,1). The DTL stack representing this path is:

$[\underline{2.2.3}, 2.2.1]$, $[\underline{2.2}]$, $[\underline{2}, 1]$

where the underline indicates the location of the pointer in each DTL. Before forwarding the call setup message to 2.2.1, node 2.2.3 advances the pointer of the first DTL which becomes: $[2.2.3, \underline{2.2.1}]$. When node 2.2.1 receives the message, it notices that the top DTL points to its own ID. Since the first and the second DTLs are exhausted, node 2.2.1 looks at the third DTL and finds that the message should be routed to node 1. Hence, it advances the pointer in the third DTL and forwards the call setup message to the border node 1.3.2 over link (2.2.1,1.3.2) with the following DTL stack: $[2, \underline{1}]$.

Node 1.3.2 receives the message and realizes that the current DTL destination has been reached. It therefore needs to build a new "source" route to descend to the final destination based on its local view of the network. It can select a path that goes through 1.2 to 1.1, or a path that goes directly from 1.3 to 1.1. Suppose it chooses the second option. It also needs to determine the exact routing inside 1.3, e.g. (1.3.2,1.3.4). It then sends the call setup message with a new DTL stack which looks as follows:

$[1.3.2, \underline{1.3.4}]$, $[\underline{1.3}, 1.1]$, $[2, \underline{1}]$.

Node 1.3.4 removes the top DTL, advances the pointer of the new top DTL, and hands the message to node 1.1.4, which is its neighbor in 1.1, with the following DTL stack:

$[1.3, \underline{1.1}]$, $[2, \underline{1}]$.

Node 1.1.4 notices that it is the current target, and therefore needs to find a route to 1.1.2 through PG(1.1). It chooses the route (1.1.4,1.1.1,1.1.2) and sends the call setup message to 1.1.1 with the following DTL stack:

$[1.1.4, \underline{1.1.1}, 1.1.2]$.

Node 1.1.1 advances the pointer of the top DTL, and forwards the message to the final destination — node 1.1.2.

III. Related Work

Hierarchical and dynamic link-state routing algorithms are designed to scale to the largest possible networks, encompassing

thousands of nodes. To support the demand for multicast application, multicast schemes, and in particular PNNI, need to support the construction of multicast trees. The cost of such trees with respect to their link metrics should be close to what can be achieved in a flat network, when a complete view of the network topology is available at every node. This is a great challenge, as the main concept behind hierarchical networks is that no single node knows the complete topology of the network. In what follows we describe several schemes for building multicast trees in flat networks. We also describe some of the ideas proposed in recent years with regard to multicast routing in PNNI.

Algorithms for the construction of efficient and dynamic multicast trees in non-hierarchical networks are extensively covered in the literature. The Waxman dynamic algorithm for computing efficient multicast trees [8] is one of the most popular. According to this algorithm, a new node that joins an existing multicast tree is connected to the nearest node on the tree. A different approach was taken by the Core Based Trees (CBT) algorithm [3]. In CBT, a multicast tree has a single predefined node, known as the tree *core*, from which branches towards group members are established. These branches are created along the shortest path between a new joining member and the core node. The main advantage of the CBT over the Waxman algorithm is that the joining node needs only to know the address of the core whereas in Waxman algorithm it must know the exact structure of the whole tree. The main disadvantage is that the cost of the CBT tree is generally higher. Several other algorithms and heuristics were proposed over the years.

ATM provides two important building blocks for multicast routing. First, an ATM switching fabric is capable of multicasting cells coming from one input port to a pre-defined list of output ports. Second, the ATM standard specifies two distinct mechanism to join a multicast tree: Leaf Initiation Join (LIJ) and Root Initiated Join (RIJ). In RIJ the root triggers the process of joining new nodes to the tree, whereas in LIJ the process is triggered by the joining nodes. However, PNNI has not specified any mechanism for supporting these two requests. Several solutions have been proposed for filling this gap:

- In the context of IP over ATM, it was proposed to use a centralized multicast server that maintains the topology of the whole network and controls new connections [20]. This scheme suffers from lack of scalability, and does not take advantage of the benefits of a hierarchical network.
- *Multiple CBT* [21]: This scheme establishes a core based tree for each PG for every multicast group. When a node wishes to join a multicast group, it tries to attach itself to the core node within its PG. If the core node is not on the tree, it first joins the multicast tree by approaching the core node in the parent PG. This procedure is performed recursively, until an "active" core node is found. The main advantages of the scheme we propose in this paper (HMF) over Multiple CBT are:

1. Our scheme supports any multicast routing algorithm, whereas this scheme supports only a specific routing algorithm. This scheme uses CBT which has been designed for flat packet switched network (the Internet), and cannot be efficiently adapted to an hierarchical circuit switched network. For instance, in CBT a join or a leave message is sent from one router to another on the path towards the core, taking advantage of the underlying packet based routing.
2. In this scheme, if a single core is used in every PG for all the multicast trees, this core is likely to become a bottleneck. If a different core is defined for every group, the overhead for maintaining the core and spreading information regarding the cores will be excessive. Our scheme nominates a "core" (called MGS, "multicast group server") only when needed. Namely, a PG that has no members in a multicast group will not have an MGS for this group. This reduces the maintenance cost significantly.

- In [22], an improvement for Multiple CBT was proposed. Instead of core nodes, this scheme uses *Rep* nodes. A Rep node performs multicast routing decisions inside its PG, but does not have to be a part of the multicast tree. This eliminates the first disadvantage of the previous solution, but does not affect the second.
- Another proposed solution is the Distributed Multicast Routing Protocol (DMRP) presented in [23], [24]. This solution relies heavily on the use of peer group leaders (PGLs) for the creation and maintenance of the multicast tree. Under this scheme, a joining node locates the nearest active node in the network in the following way:
  - In a 1-level PG, the node floods a request to find out which nodes are members of the multicast tree. After collecting the replies, it determines the identity of the closest active node.
  - If none of the nodes in the 1-level PG is on the multicast tree, the nearest 1-level PG with active nodes is located by the 1-level PGL, and this process continues recursively.

PGLs are used in this scheme to handle the multicast though they are already excessively occupied by unicast-related tasks. Therefore, the scheme lays overhead on the network, especially for small multicast groups, and does not scale to large networks where hierarchical routing is likely to be employed. As already noted, HMF may assign the multicast-related tasks to any switch in the PG.

Another advantage of HMF over this scheme is that HMF selects a node to which the joining switch should connect in a top-down manner, whereas this scheme selects it in a bottom-up manner. This not only results in less communication overhead, as noted above, but also allow to use a multicast routing algorithm that aims in achieving global optimization rather than only local optimization.

IV. The Hierarchical Multicast Framework

In this section we present the proposed Hierarchical Multicast Framework (HMF). HMF creates a single shared tree for every multicast group. This tree is shared by all the senders, which are assumed to belong to the group. The construction of the tree is performed by a series of join actions.

HMF is based on the definition of a set of *Multicast Group Servers (MGS)*, associated with each multicast tree. An MGS exists in every PG that participates in the tree. An MGS administers a single PG in the multicast tree, in a similar way to a PGL that administers a PG with respect to the PNNI flooding algorithm. The main purpose of an MGS is to store the structure of the multicast tree within its PG. The view of the multicast tree a single MGS stores relates only to its administered domain. More detailed information is stored in lower level MGSs.

The addition of a new leaf (an ATM switch) to the multicast

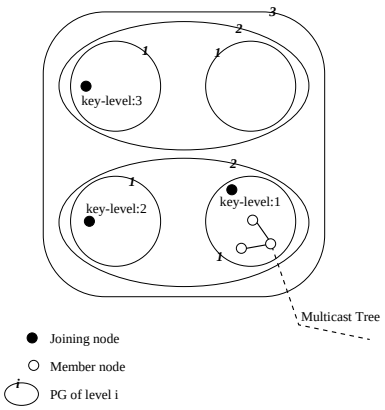

Fig. 3. Demonstration of the *key-level* concept

tree is taken care of by the MGSs of that tree in a top-down order. The leaf initiates a joining process by approaching the top level MGS. The latter contacts an appropriate lower level MGS and so forth until a switch is found in the tree, from which a physical a physical connection with the joining node is created. Routing decisions are carried out by the MGSs that govern the nodes along the newly created branch.

A. Notations

The following notations are used throughout the paper; with respect to every multicast group:

- N: The number of levels in the PNNI hierarchy.
- $MGS^l(P)$: An l-level Multicast Group Server, in charge of peer group P. Hence, $MGS^N(*)$ is the only top level MGS whereas $MGS^0(s)$ represents the multicast functionality of a single ATM switch s.
- $PG^l(S)$: An l-level PG that contains a peer group called S, where S might be a physical switch or a higher level PG. $PG^l(S)$ can be represented as a graph: (V, E) where V is a set of $(l-1)$-level child PGs and E is the set of logical links (edges) connecting the child PGs.
- *Active Peer Group:* A PG is said to be active (again, with respect to a given multicast group) if it contains switches that are members of the multicast tree. Note that by definition the parent of an active PG is an active PG as well. Also note that HMF nominates an MGS to every active PG.
- $T(P)$: A subset of peer group P, that contains the nodes and links belonging to the tree.
- *key-level:* For a switch J that joins the multicast tree, the *key-level* k is defined as:

$$k = \min_{0 \le i \le N} \{PG^i(J) | PG^i(J) \text{ is an Active Peer Group}\}$$

This definition is demonstrated in Figure 3. From the definition follows that when J is added to the multicast tree, at least $k-1$ PGs become active (one in every level i, for $0 \le i < k$).

B. Joining a Multicast Group

Adding a new switch to a multicast tree is the most complex process in HMF. For clarification, we first describe the simple case where a switch joins the tree when its 1-level PG is active. Namely, another switch in the same 1-level PG has already joined the tree. Only then we address the general case, where the 1-level PG of the joining node is not necessarily active. For simplicity, throughout this section it is assumed that the underlying multicast routing algorithm aims at finding the shortest path between the joining node and the existing tree. However, as already noted a multicast algorithm based on other considerations can be used as well.

B.1 The Simple Case: the 1-level PG of the Joining Node is Active (key-level $= 1$)

As an example, consider the hierarchical network depicted in Figure 4. In this figure a 3-level hierarchical network is described. To simplify the figure, MGSs of level n also serve as MGSs of levels 1 through $n-1$. Assume that switch B.4.1 wishes to join the multicast tree. It therefore sends a JOIN-REQ message to the highest level MGS—$MGS^3(*)$ which resides in switch D.4.3. The latter finds out that B.4.1 is placed in an active 2-level PG, PG(B). It therefore forwards the message down the hierarchy to $MGS^2(B)$ that happens to be B.3.2. MGS B.3.2 then finds out that the joining switch resides in an active 1-level PG (B.4). It therefore forwards the join message to the MGS of B.4, namely to B.4.3. Based on the underlying routing protocol and the topology view B.4.3 has, B.4.3 computes a shortest path within its PG from the joining switch B.4.1 to a switch in T(B.4). Let this switch be B.4.2. To this end, B.4.3 must know not only the structure of B.4, but also the structure of T(B.4).

MGS B.4.3 then constructs a JOIN-PATH message containing the information about the computed route. This message is sent to the connecting switch (B.4.2). Upon receiving the message, B.4.2 sends a BRANCH-SETUP message to establish the branch to B.4.1. When B.4.1 receives the BRANCH-SETUP message, the joining process is completed.

The JOIN-REQ and JOIN-PATH messages are sent from an i-level MGS to an $(i-1)$-level MGS by means of a special multicast signaling VC established between every MGS to its parent MGS. We could have used the reverse order of the same signaling VCs in order to inform J of the connecting switch L. However, this would significantly increase the delay and communication overhead of HMF.

In the simple example considered above, where the joining switch resides in a 1-level active PG, the BRANCH-SETUP message can be viewed as a regular PNNI SETUP message that establishes a unicast VC. However, we shall see later that in the general case, the BRANCH-SETUP message should also nominate MGSs in PGs that were non-active and became active due to the new branch.

B.2 The General Case: key-level > 1

We now describe the more general case, where the 1-level PG of the joining node, J, is not active. In such a case, the scheme becomes more complicated since some inactive PGs need to become active. HMF identifies these PGs, and nominate an MGS in each of them. Consider Figure 5 and let the joining node be J=A.1.2. J joins the tree when PG(A) is inactive. It sends a JOIN-REQ message to $MGS^3(*)$—switch D.4.3. D.4.3 realizes that J belongs to an inactive 2-level PG. Based on the underlying multicast routing algorithm, D.4.3 determines the 2-level PG path between A.1.2 and the tree. As noted before, sup-

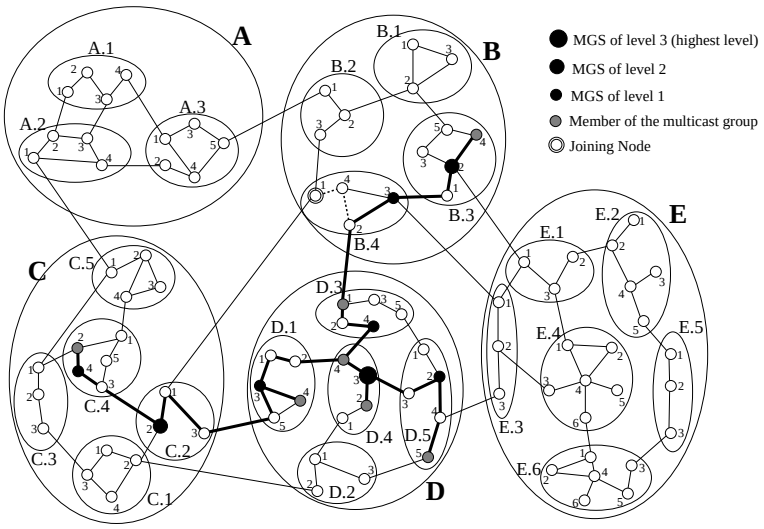

Fig. 4. Adding a Switch to the Tree - the Simple Case

pose that the multicast routing algorithm searches for the active PG closest to J. In such a case $MGS^3(*)$ would select either the path [C,A] or the path [B,A]. Let the selected path be [C,A]. A JOIN-PATH message is therefore sent to $MGS^2(C)$, namely to switch C.2.2.

Switch C.2.2 computes a 1-level route within PG(C) from the multicast tree to the logical link connecting PG(C) to PG(A). The resulting path, [C.4, C.5], is attached to a JOIN-PATH message sent to the MGS of C.4, namely C.4.4. The latter repeats the process by finding a route that crosses C.4 towards C.5: [C.4.3, C.4.5, C.4.1]. This implies that a new branch should be set up between J and switch C.4.3. Let C.4.3 be referred to as L.

When L (C.4.3) receives the message from C.4.4, it constructs a BRANCH-SETUP message and sends it towards the joining switch J (A.1.2). This message contains a DTL stack that was created during the HMF execution and describes the route from L to J, as selected by the specific underlying routing algorithm. In our example, the DTL stack is as follows:

Level	MGS	Route
1	C.4.4	[C.4.3, C.4.5, C.4.1]
2	C.2.2	[C.4, C.5]
3	D.4.3	[C, A]

Physical border nodes along the created branch (the dashed line in Figure 5) should perform two operations:

- Calculate the route within their PGs. Switch C.5.4, for example, replaces the lowest level route in the BRANCH-SETUP message with a new route [C.5.4, C.5.2, C.5.1]. Switch A.2.1 calculates new routes for both level 1 and level 2 of the DTL stack, and so forth. This is a normal procedure performed by an entry node when a regular unicast SETUP message is received. However, it can be performed according to the specific underlying multicast routing algorithm.
- Nominate MGSs in their PG: a 1-level border node (A.1.1) should nominate only one MGS, a 2-level border node (A.2.1) should nominate 2 MGSs, and so forth.

A newly created MGS can deduce from the BRANCH-SETUP message the identity of other related MGSs. It uses this information in order to create a signaling VC with each of them. When the joining node receives the BRANCH-SETUP message, the process is completed.

As described above, in HMF only active PGs need to maintain information about the tree. This is one of the most important features of HMF, that enables its scalability.

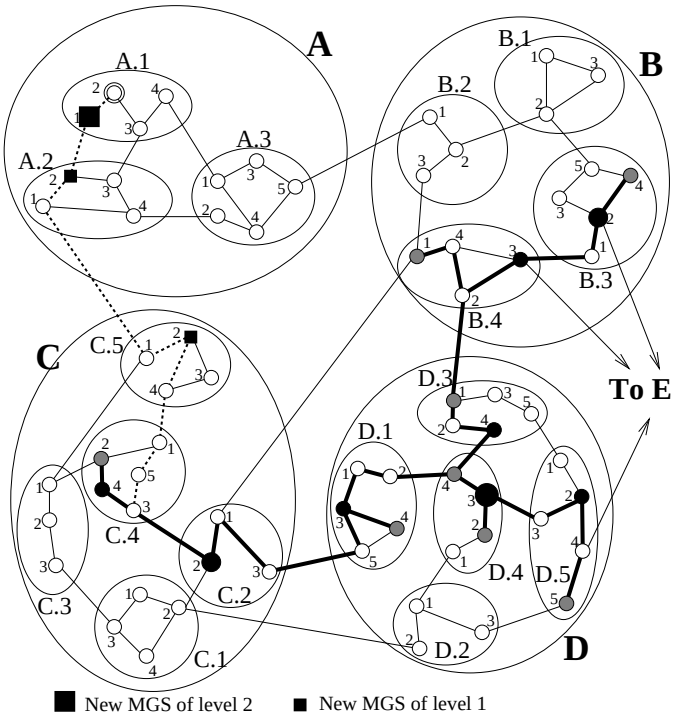

Fig. 5. Adding a Switch to the Tree - the General Case

We now summarize the four phases required for adding a new switch to a multicast tree:

1. J sends a JOIN-REQ(J) message to $MGS^N(*)$.
2. The JOIN-REQ message is sent down the MGS hierarchy towards the PG of J. In every level l, starting with $l = N$, the message is sent from $MGS^l(PG^l(J))$ to $MGS^{l-1}(PG^{l-1}(J)$.

This phase ends when the JOIN-REQ message reaches the MGS of the lowest level active PG that contains J — the key-MGS.
3. After the key-MGS is reached, at every level l:

- A route of $(l-1)$-level PGs between the existing tree and J is determined. The first PG along this route must be active, whereas the other PGs are inactive.
- A JOIN-PATH message is then forwarded to the active lower level PG. This message contains information about the route found so far, for levels l, $l+1$, ..., $key\text{-}level-1$.

This process is repeated until L is located, and a DTL-list describing the route between L and J is generated.
4. A connection is established between L and J, and MGSs are assigned to all the PGs along that route. These PGs must be inactive, because otherwise the route would have been created from the last active PG. The branch from L to J is setup along the path calculated during the previous phase, using BRANCH-SETUP messages.

B.3 MGS nomination

One of the most difficult tasks performed by HMF is nominating a new MGS for every PG that becomes active. Theoretically, there are two ways to perform this task: top-down or bottom-up. We first show that a top-down scheme would not work well, and then propose a bottom-up scheme.
In the top-down scheme, the MGSs of level $key-1$ are directly nominated by the key-MGS. They then nominate the $(key-2)$-level MGSs, and so forth. However, the key-MGS does not know the exact topology of its administered PG. Hence, it can nominate an MGS only after somehow collecting topological information for every relevant lower level PG, thereby increasing the complexity of HMF considerably.
We propose to use a bottom-up scheme, performed during the setup of the branch from L to J in phase 4, in the following way. A border node of level $l>1$ determines the $(l-1)$-level PG where the l-level MGS will reside. It attaches the identity of this MGS to the BRANCH-SETUP message. When the border node of the selected PG receives the BRANCH-SETUP message, it repeats the process by determining the $(l-2)$-level PG that will contains the MGS, and so forth. This process ends when a border node of 1-level PG determines the specific physical node that will serve as the l-level MGS.
In terms of the example consider in Figure 5, MGSs are nominated for PGs of both 1-level (C.5, A.2, A.1) and 2-level (A) PGs. $MGS^1(C.5)$ and $MGS^1(A.2)$ are nominated by the 1-level border node at the entry to their PG—switches C.5.4 and A.2.1 respectively. $MGS^2(A)$, however, is through the following process:

- The 2-level border node of A for this particular BRANCH-SETUP, namely switch A.2.1, determines that $MGS^2(A)$ should reside in PG A.1.
- The border node at the entry to A.1, namely switch A.1.1, deduces from the BRANCH-SETUP message it receives that a 2-level MGS should be appointed in its 1-level PG. It then decides to nominate itself as $MGS^2(A)$. In addition it should nominate a 1-level MGS for A.1, and for some reason it decides to take this task as well. In general, the entry switch may select the MGS randomly or based on knowledge regarding the load imposed on every switch.

In principle, the above process could be simplified if we agree that an l-level physical border node will serve as an MGS of all the l levels. However, the border nodes are usually the bottleneck of an ATM network [25] and therefore will probably not be able to assume further tasks.

B.4 Joining the Multicast Group: Discussion

As explained above, two simultaneous activities occur during the third phase of HMF: the connecting switch L is chosen, and the route from L to J is computed. The routing computations during this phase are performed by the MGSs of different hierarchy levels. One may argue that L could have computed the route towards J as if it was a regular unicast setup process thereby avoiding the need to collect routing information when the JOIN-PATH message is forwarded from one MGS to another. However, this would be a bad approach for the following reasons:

- The considerations made by the unicast routing algorithm might be different from those made by the underlying multicast routing algorithm. For example, some multicast algorithms may choose a longer path that goes through regions which contain hosts that are likely to join the group in the future. By determining the route between L and J during the propagation of the JOIN-PATH message, we actually allow the multicast routing algorithm to determine how the tree would look like. Note, however that our scheme is still compatible with PNNI, because the BRANCH-SETUP message has a DTL-stack exactly like every unicast SETUP message has (see Section II). The difference is that HMF computes this stack based on considerations of the multicast algorithm, rather than of the considerations of the unicast algorithm.
- L does not hold any information about the structure of the tree in higher levels of the hierarchy. Hence, it might choose a route that goes through active PGs (e.g. C.4.3, may choose the path [C, B, A] in our example). Trying to setup a new branch through such PGs can lead to multiple MGSs in the same PG, which contradicts one of the basic assumptions of HMF.
- Another, less significant, reason is the issue of computational complexity. Parts of the route from L to J are already computed by the MGSs during the selection of L. Therefore, it makes no sense to repeat this computation after L is selected.

HMF aims at minimizing the information a joining node must have in order to join the tree. This property is essential in an environment where many multicast groups will be active at the same time, but only a small fraction of the nodes in the network is likely to join each group. In HMF a joining node needs only know the identity of the highest level MGS. The identity of $MGS^N(*)$ can be found by means of some directory service like DNS or LDAP. If a specific multicast group is only intended for a subset of the network which fully resides in a PG of some level l, the top level MGS of the group can be defined for this PG, rather than for PG(*). This will reduce communication and time complexity.

Deadlocks and loops are prevented due to a central control of MGSs. During the time when branches are established or deleted, the multicast tree is not stable. However, the scope of this instability is bounded to the PG where changes are made. For that reason, an MGS suspends JOIN-PATH messages that

need to traverse an unstable child PG, until the setup process of a previously joined switch nominates an MGS for that PG.

C. Leaving a Multicast Tree

Suppose that a switch wants to leave a multicast group. If the switch is a leaf of the multicast tree, its branch is pruned. If it is not a leaf, it cannot be pruned. However, it should notify its MGS that it does not belong to the multicast group anymore.

The mechanism for branch pruning in HMF is based on the use of a PRUNE message initiated by the leaving leaf. This message is then repeatedly forwarded by the nodes along the pruned branch, until it is blocked by a switch that is a member of the group, or by a switch that is the origin of another branch.

As an example, consider Figure 5 and assume the case where switch A.1.2 leaves the multicast group. It therefore sends a PRUNE message to switch A.1.1, thus initiating the removal of the the branch [A.1.2, A.1.1, ..., C.4.5, C.4.3] from the tree. The pruning ends only at C.4.3, because it is a fork of another branch. A significant by-product of the pruning is the deactivation of PG A (including A.1 and A.2) and PG C.5. When an PG becomes inactive, its MGS is cancelled, and the MGS of its parent MGS is informed. $MGS^1(C.5)$ (switch C.5.2), for example, is informed of the pruning by the physical border node C.5.4. Consequently, C.5.2 informs $MGS^2(C)$ (switch C.2.2) that C.5 became inactive, and stop functioning as an MGS. The same mechanism is used recursively in PG A.

In general, the MGS of a PG where pruning is performed is informed either by:

- A switch that receives a PRUNE message, but stops the PRUNE cascade, or by
- A border node, that is the last switch in the PG that belongs to the multicast tree. In this case, the MGS notifies its parent MGS.

V. HMF Properties

In the following section we investigate several aspects of multicast tree performance: protocol execution load, setup time, tree cost and memory complexity. As already mentioned, each of the existing hierarchical multicast schemes bears at least one severe disadvantage with respect to these aspects of performances.

A. Protocol Execution Load

In order to measure the processing load imposed by the HMF on the network switches, we count the number of switches that need to process a signaling message when a new switch joins the tree. Consider a uniform hierarchical network, where different PGs have identical internal topology regardless of their hierarchy level. Let d be the size of the average path in a PG, namely the mean number of child PGs a message needs to pass in order to cross the PG including the entry and exit nodes.

Theorem 1: The communication processing load imposed by HMF during the join process is in the same order of the processing load imposed by the setup of a unicast VC.

We prove this by showing that in both cases the number of switches that process a signaling message is $O(d^N)$ where N is the level of the topmost PG.

Lemma 1: The maximal number of switches residing along a path between two switches whose lowest common PG is in level m is d^m.

The proof is by induction. The claim clearly holds for a 1-level PG. Suppose that it holds for an $(m-1)$-level PG. In the case where the lowest common PG resides in level m, the path crosses on the average d $(m-1)$-level PGs, and the total number of switches is therefore $d \cdot d^{m-1} = d^m$.

From Lemma 1 follows that the number of nodes involved in the processing of a unicast VC setup messages is at most d^m. Hence, the processing load imposed by a unicast VC setup is $O(d^N)$.

We next estimate the processing load imposed by HMF for the join process. To this end, we distinguish between the following phases:

1. The joining node J sends a signaling message to $MGS^N(*)$. Whether J establishes a VC to this end, or it uses an embedded "packet switched signaling network", the processing complexity is $O(d^N)$.
2. Forwarding the JOIN-REQ N times, from one MGS to another over existing VCs. N switches (MGSs) need to process a signaling message during this phase.
3. Forwarding a JOIN-REQ from the 1-level MGS of L to L. At most d switches will have to process the JOIN-REQ during this phase.
4. The BRANCH-SETUP message travels from L to J. By Lemma 1, $O(d^k)$ switches are involved, where k is the key-level in the particular join process.
5. Finally, a signaling VC needs to be established between every PG that becomes active and its parent PG. These VCs will serve switches that will join the multicast tree later. Therefore, for a dense tree the contribution of the processing load during this phase per a single joining switch is negligible. In any case, by Lemma 1, the number of switches involved in the setup of a signaling VC between an $(i-1)$-level MGS and its parent is $O(d^i)$. Therefore, for setting up k such VCs between a 1-level MGS to its parent, and then from the 2-level MGS to its parent and so forth, the total processing complexity would be $O(\sum_{i=1}^{k} d^i) = O(d^{k+1})$. This concludes the proof of Theorem 1.

B. Tree Size

The cost of the tree is mainly determined by the underlying "flat multicast routing algorithm". However, the purpose of this section is to show how efficient is the tree generated by a given multicast routing algorithm when implemented in the contest of HMF, comparing to the tree generated by the same algorithm in flat network. To this end, we ran simulations on a randomly generated hierarchical networks, and on the associated copy of the underlying flat networks. The aggregation mechanism for additive metrics in our hierarchical network model is that each PG advertises one metric denoting the average delay needed for crossing it between all border node pairs.

In every simulation instance, we constructed the same multicast group in the two hierarchical and flat networks by repeatedly adding and deleting nodes from the group. We used the function

$$P_{event}(s) = \frac{\alpha(n-s)}{\alpha(n-s) + (1-\alpha)s}$$

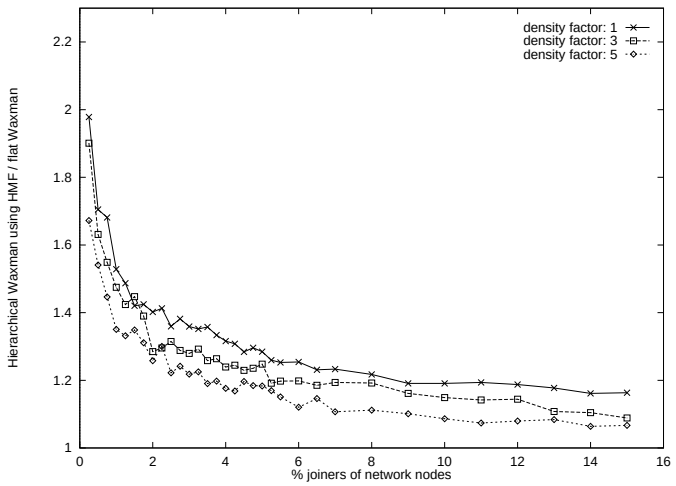

Fig. 6. Waxman Tree Cost — Hierarchical vs. Flat Networks

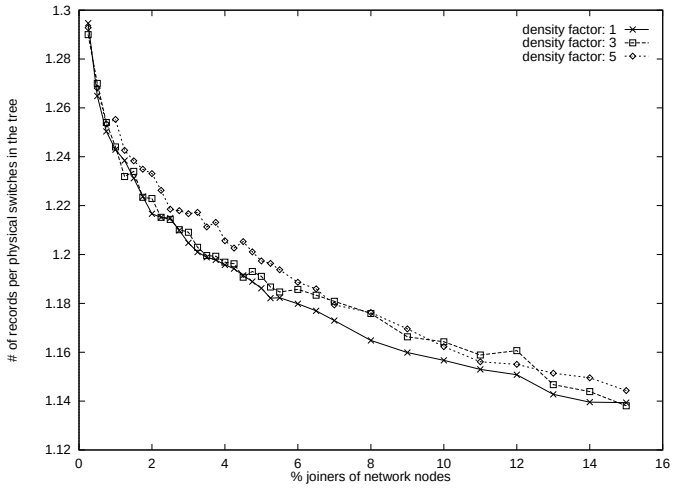

Fig. 7. Number of nodes in the generated tree: $\frac{Logical}{Physical}$

to determine the probability that an event is a node addition or deletion, where n is the total number of physical nodes in the network, s is the current size of the multicast group, and α is the expected fraction of nodes in the multicast group.

To simulate "locality of interest", where nodes from the same neighborhood are more likely to join the same multicast group, we use a *density factor*, denoted τ. After uniformly choosing a candidate node J for addition, the probability of adding J to the group is $P_{density}(k) = \tau^{-k}$ where k is the key-level of J. With probability $1 - P_{density}(k)$, J is ignored and another candidate is selected. Due to the space constraint, we show the results achieved only for Waxman [8] algorithm.

Figure 6 shows the ratio between the cost of the tree generated by this algorithm when adapted to run with HMF, and the cost of the tree generated by this algorithm in a flat network. We generated a 3-level network with an average of 25 nodes per PG (approximately 15,000 physical nodes). We ran simulations for $\alpha \in [0 \ldots 15]$, and with density factors of 1, 3 and 5.

It is evident that ratio between the performance in a flat networks and in hierarchical network ranges from almost 2 in very small groups ($\alpha = 0.25\%$), when the members of the group are evenly spread in the network ($\tau = 1$), to approximately 1.3 when the size of the group reaches 3% of the network population. As expected, the hierarchical Waxman multicast routing algorithm performs better for dense multicast groups because when the group is denser, the key-level for an arbitrary joining node becomes lower, and the aggregation error decreases. Consequently, the decisions made by the multicast routing algorithm are more accurate.

C. Memory

As opposed to previously proposed schemes [20], [21], [22], HMF requires only the MGS of an active PG to store a record for the multicast tree. Hence, one record is needed for every multicast group in every PG. In practice, small local multicast group span only few PGs, and impose a much lower memory complexity. We used our simulation model to measure the number of all active PGs, including 0-level PGs (i.e switches that sit on the tree data path) as a function of the number of physical nodes in the tree. The results are presented in Figure 7. It is evident that the memory cost per every switch on the tree data path ranges between 1.15 and 1.3.

VI. Conclusions

We presented a novel framework for the creation and management of multicast trees in hierarchical networks. The framework is tailored for the ATM PNNI hierarchical model. The essence of HMF is the distribution of information and routing decisions between dynamically nominated Multicast Group Servers (MGS).

The proposed framework addresses disadvantages of the existing schemes without imposing new disadvantages: It creates cost efficient and hot-spot-free trees compared to CBT based algorithms such as [21] and [22] (which disregard available link-state information). Unlike [20],[21] and [22] the proposed scheme imposes marginal storage requirements on the network nodes. Finally, the communication cost associated with the new mechanism is of the same order as that of PNNI-unicast, as opposed to expensive flooding which is performed in [23], [24].

Other advantages of HMF are as follows:

- The MGS functionality for different multicast groups can be spread between all physical nodes. This prevents overloading specific nodes, such as the peer group leaders.
- Routing decisions concerning the addition of a new node to the tree are performed in a top-down manner. This enables the underlying flat multicast routing algorithm to employ global optimization considerations rather than local ones.
- Inactive nodes and inactive PGs need not store any information about the multicast tree.

We achieve these advantages through an MGS nomination mechanism, whose role is to appoint new MGSs to PGs that become active during the addition of new nodes to the tree.

We showed that when using the Waxman routing algorithm, HMF generates efficient trees, exploits compact memory space, and imposes acceptable setup and protocol execution load.

VII. Acknowledgment

We would like to thank Jenny Sannikov, Nelly Bluvshtein and Yoav Manor for their help in coding the simulation software.

References

[1] B. Awerbuch and D. Peleg, "Sparse partitions," in *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, IEEE, Ed., St. Louis, MS, Oct. 1990, pp. 503–513, IEEE Computer Society Press.

[2] ATM Forum PNNI SWG 94-0471R13, *ATM Forum PNNI Draft Specifications*, Mar. 1996.

[3] T. Ballardie, P. Francis, and J. Crowcroft, "Core based trees (CBT) - an architecture for scalable inter domain multicast routing," in *ACM SIGCOMM*, 1993.

[4] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei, "The PIM architecture for wide-area multicast routing," *IEEE Transactions on Networking*, pp. 153–162, 1996.

[5] E. Gilbert and H. Pollak, "Steiner minimal tree," *SIAM J. Appl. Math.*, vol. 1, 1968.

[6] P. Winter, "Steiner problem in networks: A survey," *NETWORKS*, pp. 17:129–167, 1987.

[7] E. Aharoni and R. Cohen, "Restricted dynamic Steiner trees for scalable multicast in datagram networks," *IEEE/ACM Transactions on Networking*, vol. 6, no. 3, June 1998.

[8] B. Waxman, "Routing of multipoint connections," *IEEE JSAC, J. Selected Areas of Communication*, vol. 1, Dec. 1988.

[9] K. Kumar and J. Jaffe, "Routing to multiple destinations in computer networks," *IEEE Transactions on Communications*, pp. 31:343–351, 1983.

[10] M. Imase and B. Waxman, "Dynamic steiner tree algorithm," *SIAM J. Disc. Math.*, vol. 1, Aug. 1991.

[11] J. Kadirire, "Comparison of dynamic multicast routing algorithms for wide-area packet switched networks," in *IEEE INFOCOM*, 1995.

[12] C. Huitema, *Routing in the Internet*, Prentice Hall, 1995.

[13] L. Kleinrock and F. Kamoun, "Hierarchical routing for large networks; performance evaluation and optimization," *Computer Networks*, vol. 1, pp. 155–174, 1977.

[14] D. Peleg and E. Upfal, "A trade-off between space and efficiency for routing tables," *Journal of the ACM, JACM*, vol. 36, no. 3, pp. 510–530, July 1989.

[15] W. Lee, "Topology aggregation for hierarchical routing in ATM networks.," in *ACM SIGCOMM Computer Communication Review*, Apr. 1995.

[16] W. Lee, "Spanning tree method for link state aggregation in lrage communication networks," in *IEEE INFOCOM*, 1995.

[17] B. Awerbuch and Y. Shavitt, "Topology aggregation for directed graph," Tech. Rep. 98-14, DIMACS, Feb. 23 1998.

[18] B. Awerbuch, Y. Du, B. Khan, and Y. Shavitt, "Routing through networks with hierarchical topology aggregation," *Journal of High Speed Networks*, vol. 7, no. 1, 1998.

[19] E. Felstaine, R. Cohen, and O. Hadar, "Crankback prediction in hierarchical ATM networks," in *IEEE INFOCOM*, 1999.

[20] G. Armitage, "Support for multicast over UNI3.0/3.1 based ATM networks," Contribution to the ATM Forum Technical Committee, November 1996, RFC 2022.

[21] R. Vekateswaran, C.S. Raghavendra, X. Chen, and V.P. Kumar, "Hierarchical multicast routing in ATM networks," in *IEEE International Conference on Communications v 3 1996*, Dallas, TX, USA, 1996.

[22] R. Vekateswaran, C.S. Raghavendra, X. Chen, and V.P. Kumar, "A scalable, dynamic multicast routing algorithms in ATM networks," in *1997 IEEE International Conference on Communications, Towards the Knowledge Millenium. ICC '97*, New York, NY, USA, 1997.

[23] R. Vekateswaran, C.S. Raghavendra, X. Chen, and V.P. Kumar, "Drmp: A distributed multicast routing protocol for ATM networks," in *Proceedings of the 1997 IEEE ATM Workshop*, 1997.

[24] R. Vekateswaran, C.S. Raghavendra, X. Chen, and V.P. Kumar, "Support for group multicast in PNNI," Contribution to the ATM Forum Technical Committee, February 1997, ATM FORUM/97-0076.

[25] E. Felstaine and R. Cohen, "On the distribution of routing computation in hierarchical ATM networks," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, Dec. 1999.