# Resource and Topology Discovery for IP Multicast using a Fan-out Decrement Mechanism

Jangwon Lee, Gustavo de Veciana
Department of Electrical and Computer Engineering
University of Texas at Austin
E-mail: {jangwlee, gustavo}@ece.utexas.edu

*Abstract*— **As the use of IP multicast sessions becomes widespread, the potential benefits derived from currently unavailable topological information on multicast distribution trees may become increasingly critical. In this paper we propose a framework for discovering the topology of *shared* multicast trees based on a novel *fan-out decrement* mechanism analogous to TTL decrementing in IP. We propose an algorithm for topology discovery based on the matrix of path/fan-out distances among a set of $E$ session members – the algorithm's computational complexity is $O(|E|^2)$. We exhibit sufficient conditions for topology discovery based on a *reduced* distance matrix, and propose a practical protocol to acquire this information requiring the exchange of $2|E|$ multicast messages of size $O(|E|)$. Finally, we show how the same approach permits nodes to discover the multicast distribution tree associated with members within their fan-out/TTL scoped neighborhoods. This permits one to reduce the computational costs while making the communication costs proportional to the size of neighborhoods.**

*Keywords*—**IP Multicast, Fan-out decrement, Topology discovery**

## I. INTRODUCTION

MULTICAST extensions to IP [1] have enabled a wide range of applications including real-time audio and video broadcasting, shared electronic white boards, software distribution and web-casting. One of the key advantages of multicasting lies, of course, in the efficient use of network resources – a single packet traverses each link in the multicast distribution tree and is replicated at fan-out points. Another advantage associated with IP multicast service, is as an abstraction for group communication, that is, users can join and leave a multicast session without explicit knowledge of its membership or of the structure of the distribution tree. Despite this clean abstraction, many IP multicast services can benefit from the explicit knowledge of topology and membership information.

For example, from the perspective of an IP multicast service user (e.g., movie distributor, advertiser) the number of subscribers in a session, their location, and their density in a specific region may be useful information. From the perspective of a network service provider, the extent to which network resources are being used (e.g. number of links and routers) by a given multicast session may be important to assess usage costs. In both cases knowledge of the *global* multicast topology would significantly facilitate resource management.[1] In a large scale multicast session, it may be the case that nodes are spatially clustered and it is desirable for these nodes to cooperate and perform common tasks, such as distributed computation and data sharing. In this case, the *local* topology and membership information for a neighborhood of a given node would be useful.

A typical use of local resource and topology discovery is in building schemes for loss recovery and congestion control in the context of multicast sessions supporting heterogeneous receivers. While a variety of approaches have been proposed to tackle this problem, e.g., [2], [3], [4], [5], [6], a common thread is to recognize that performance can be enhanced by either implicitly or explicitly exploiting the structure of the multicast distribution tree. OTERS [3], Tracer [2] and GFP [7] are examples of research efforts making use of explicit topology information via MTRACE [8] and an inference technique [9] for local loss recovery. Further motivation for exposing the multicast distribution tree is given in [9], [7] and [10].

Despite its potential usefulness, there has been surprisingly little research, see e.g., [11], [9], concerning global multicast topology discovery and even less, to our knowledge, concerning local multicast topology discovery. A large amount of work has however been devoted to Internet topology discovery, see e.g., [12], [13], [14], [15]. By contrast with multicast topology discovery, Internet topology information can be collected during long time scales (e.g., several days or even several weeks) [12], or by passive probing [16], since the physical topology remains stable over reasonably long time periods. In the case of multicast service the character of the distribution tree is only of interest when the session is active and may change dynamically throughout that period. Thus multicast topology discovery algorithms should be able to operate online and serve as practical protocol building blocks which dynamically track membership changes. As will be discussed below, these and other requirements make proposed approaches based on end-to-end measurements, [11], [9] fall short as practical solutions.

The following are some desirable characteristics that a multicast topology discovery mechanism should have.
*Accuracy:* Topology information should be "reliable" since potentially critical decisions will be based on it.
*Adaptability:* A mechanism should adapt to changes in group membership or distribution path topology.
*Low overheads:* Computational requirements at end hosts or servers and communication overheads should be low.
*Distributed:* From the perspective of robustness, it is preferable that topology information be distributed to those who need it.
*Scalability:* Mechanisms for topology discovery should be scalable and possibly allow partial (i.e., local) topology discovery.
With these in mind, in this paper we propose a new approach to multicast topology discovery. It is based on introducing a novel

This work is supported by the National Science Foundation Grant ANI-0081360 and by a research grant from Tivoli Systems.

[1]Throughout the paper a multicast topology refers to the multicast distribution tree constructed by multicast routing protocols.

fan-out decrement mechanism to IP multicast service, which is analogous to the time-to-live (TTL), or hop count, decrement mechanism currently supported in IP. As discussed in the sequel, the proposed scheme achieves all of the desirable characteristics posed above but *only* for the case where multicast service is based on *shared tree*, e.g., Core Based Trees(CBT) [17], [18], versus source tree routing.

The paper is organized as follows. In Section II, we discuss the advantages and shortcomings of previous work on this problem. Section III introduces the proposed fan-out decrement mechanism, briefly indicating some of its uses for resource and topology discovery. In Section IV we propose and analyze an algorithm for global multicast tree discovery. Section V includes comments on implementation and information exchange, and is followed by Section VI wherein we discuss a framework for partial (i.e., local) topology discovery of multicast trees. Conclusions and future work are included in Section VII.

## II. Related Work

Existing approaches to multicast distribution tree discovery can be classified into two types: those based on end-to-end measurements [11], [9], and those requiring the help of intervening network nodes [8].

The key idea underlying the first approach is that receivers sharing common paths on the multicast tree associated with a given source will see correlations in their packet losses. Thus based on the shared loss statistics for transmitted probe packets one can attempt to infer the multicast tree. This elegant approach to the problem is particularly advantageous in that it requires no support from internal nodes. It does however, potentially suffer from significant communication overheads required to periodically gather large amounts of loss data so as to adapt to changing memberships or topology, and processing overheads to assemble and perform the inference step. This is currently conceived as a centralized approach whose accuracy is unlikely to scale nicely. The approach assumes network links have steady state loss characteristics, which may or may not be realistic on the time-scales during which loss data are collected. A final point is that the approach permits identification of the "logical" multicast topology rather than the actual physical topology. As discussed in the sequel, this would mean that a session member that is at the end of a long path with no intervening fanout points, would see this section of its path collapsed to a single 'logical' link. In practice this may or may not be an appropriate abstraction of the actual topology. The key advantage of this approach lies in its applicability to inferring multicast trees without requiring modifications to, or the help from, internal nodes.

The second approach to multicast topology discovery is based on using the MTRACE feature currently implemented in the IGMP protocol [19]. MTRACE enables tracing the path from a source to a destination on a given multicast distribution tree [8]. A query packet is sent from the requester to the last multicast router (on the distribution tree) prior to a given destination. This query is then forwarded hop-by-hop along the reverse path from the "last-hop" router to "first-hop" router, i.e., that to which the source is attached. While the query packet traverses the tree, each router adds a response data block containing its interface addresses and packet statistics. When the query packet reaches the first-hop router it is sent back to the requester via unicasting or multicasting.

Note that an MTRACE query provides full information, i.e., interface addresses and performance characteristics, but *only* for *one* path from a multicast source to a given destination. Thus if all members wish to know the full multicast topology for a given source, each receiver would send a query packet to its last-hop router, and query responses should be *multicast* to entire group. Note that all query traffic would visit the first-hop router which would in turn generate multicast responses. Due to this focussed load, in a large-scale multicast session, this approach may not scale. Moreover in order to enable reconstruction of the full multicast topology each packet should include a stack of interface addresses for nodes along the path from the source and the destination. Thus not only the processing but possibly the communication overheads associated with obtaining and distributing this information are high. Key advantages of this approach are that it provides full information on the multicast topology based on currently available IGMP features.

In summary, while the first approach is strictly based on using end-to-end measurements, the second relies heavily on special services at routers, thus from the perspective of required network support these are two extremes of the spectrum. Also the first approach identifies the logical topology while the second determines the physical topology including interface addresses of routers. As will be seen, our approach lies somewhere in their midst, requiring light weight cooperation from multicast capable routers to identify the physical topology (without internal interface addresses) but *only* for the case of *shared* multicast trees.

## III. Fan-out decrement mechanism

We propose a fan-out decrement mechanism for IP Multicast service, which supports the following three elements/behaviors:
1. A fan-out field in the IP Multicast packet header;
2. When a multicast packet traverses a router, corresponding to a fan-out point along the path from the source to the destination, the router decrements the fan-out field by one.
3. Multicast routers at fan-out point discard incoming packets whose fan-out fields have reached 0.

Note that these components are entirely analogous to those of the current TTL decrement mechanism. The main difference is the location where decrementing occurs: every router along the path of a packet for the TTL field while only routers corresponding to fan-out points in multicast distribution tree for fan-out field.

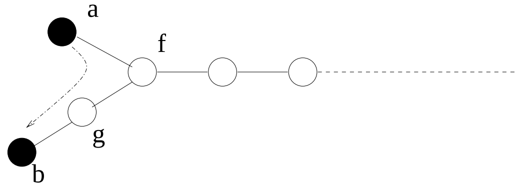

Fig. 1. Fan-out decrement mechanism illustration.

Consider the example shown in Figure 1. Suppose member $a$ multicasts a packet with its fan-out field set to 1. When the packet reaches fan-out node $f$, the fan-out field is set 0 but the

TABLE I

Parallels between IP and IP Multicast.

| IP | IP Multicast |
|---|---|
| ICMP | IGMP |
| Traceroute | MTRACE |
| TTL decrement | Fan-out decrement |

packet is duplicated and forwarded and will reach member $b$. Note that routers that are not fan-out points in the distribution tree, e.g., $g$, do not decrement the fanout field or discard packets whose fan-out field is 0.

Clearly this mechanism serves as an intuitive and natural counterpart to the TTL decrement mechanism in IP. Table I summarizes parallels between IP and IP Multicast components. The original purpose for the TTL decrement mechanism was to bound the life of packets in the network to circumvent the adverse effects of forwarding loops during routing transients. However, due to its simplicity and usefulness, the TTL decrement mechanism also found other applications in resource and path discovery, e.g., expanding ring search, traceroute [20]. We believe that, in the context of multicast service, the proposed fan-out decrement mechanism can play a similar role. For example, suppose a member in a multicast session wishes to discover the existence of another one with a given attribute but close by. Currently, it may do so using expanding ring search: i.e., multicasting a sequence of query packets with increasing TTL until an appropriate reply is received. Note that we can save time and resources by using the fan-out field to perform an expanding ring search. The possible increase in efficiency for such a search, can be seen by considering the following of two members that are only one fan-out away but a large hop count distant from each other. In this scenario, which might not be infrequent for sparse large-scale multicast sessions, an expanding ring search based on the fan-out field can quickly identify a close member. Note that this type of resource discovery is applicable to *both* source and shared tree routing protocols.

As another useful application, we propose the discovery of shared multicast trees based on the proposed fan-out decrement mechanism. Our algorithm requires that each node acquire a *distance matrix* for the current session members, which is the path and fan-out distances of pairs of members. In order to do so, packets will need to carry two additional pieces of information, Initial_TTL and Initial_fan-out, corresponding to the initial values of the TTL and fan-out fields. Clearly with this information in hand, a receiver can immediately compute its path distance and fan-out distance, i.e., number of fan-out nodes traversed, from the source. In the next section we shall develop a tree discovery algorithm based on full and reduced distance matrices. In Section V we will discuss practical issues in efficiently acquiring and distributing the required distance information.

## IV. Tree discovery algorithm

We will consider several variations of the following basic problem: given the *distance matrix* associated with the members (i.e., end hosts) of a multicast session using a shared distribution tree, determine its physical topology.

### A. Model and Notation

We will use the physical multicast tree illustrated in Figure 2 as a reference in discussing our model.[2] The end nodes, shown as solid black circles, correspond to members of the multicast session, while internal nodes, corresponding to network routers, are shown as white circles.[3] In the sequel we will refer to internal nodes where multiple copies of a multicast packet are created as *fan-out nodes.*

We define two types of distances between nodes on a tree. The *path distance* $d_p(m,n)$ between two nodes, $m$ and $n$, corresponds to the number of links along the path between them. The *fan-out distance* $d_f(m,n)$ between two nodes, $m$ and $n$, corresponds to the number of fan-out nodes on the path between them. Note that in the case where $m$ or $n$ are themselves fan-out nodes in the tree, the fan-out distance does not include $m$ or $n$'s. to the fan-out distance is not counted. For example, $d_f(f_1,f_2)=1$ in Figure 2. We denote such path and fan-out distance as a tuple $d(m,n)=(d_p(m,n),d_f(m,n))$, e.g., for our example we have $d(e_2,e_6)=(8,4)$. Table II exhibits the *full distance matrix*, which contains the distances among all pairs of members in the multicast session shown in Figure 2. Note that this table is symmetric.

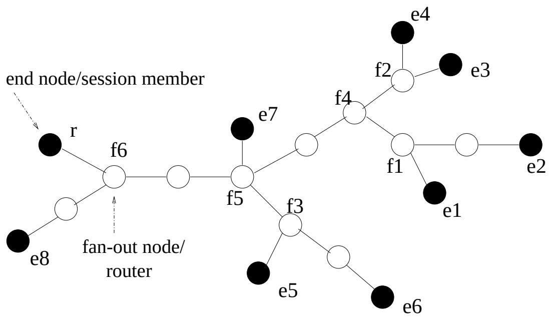

Fig. 2. Example of a physical shared multicast tree.

TABLE II

Full distance matrix for tree in Figure 2.

| | $r$ | $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |
|---|---|---|---|---|---|---|---|---|---|
| $r$ | | (7,4) | (8,4) | (7,4) | (7,4) | (5,3) | (6,3) | (4,2) | (3,1) |
| $e_1$ | | | (3,1) | (4,3) | (4,3) | (6,4) | (7,4) | (5,3) | (8,4) |
| $e_2$ | | | | (5,3) | (5,3) | (7,4) | (8,4) | (6,3) | (9,4) |
| $e_3$ | | | | | (2,1) | (6,4) | (7,4) | (5,3) | (8,4) |
| $e_4$ | | | | | | (6,4) | (7,4) | (5,3) | (8,4) |
| $e_5$ | | | | | | | (3,1) | (3,2) | (6,3) |
| $e_6$ | | | | | | | | (4,2) | (7,3) |
| $e_7$ | | | | | | | | | (5,2) |
| $e_8$ | | | | | | | | | |

When a node $m$ is connected to a link $l$, $m$ and $l$ are said to be *incident* on each other. The number of links incident on a node $m$ is called the *degree* of $m$. We say node $n$ is *adjacent* to a node $m$ if the nodes share a link.

[2]Throughout the paper, a multicast tree or a tree means a shared multicast tree, unless explicitly mentioned.

[3]In a multi-access LAN environment, an end node can be considered as a representative of all multicast members on the LAN, e.g., the one with the lowest IP address.

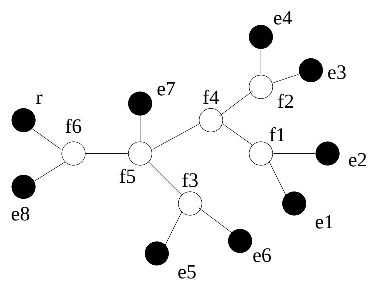

Fig. 3. The logical tree for our example.

The *logical tree* associated with a physical tree is obtained by eliminating internal nodes whose degree is 2. For example, Figure 3 depicts the logical tree corresponding to the physical tree in Figure 2. The nodes in a logical tree can be partitioned into *end nodes* $E$, whose degree is 1, and *fan-out nodes* $F$, whose degree is at least 3. In the sequel we let $|A|$ denote the cardinality of a set $A$. For a fan-out node $f \in F$ we let $AE_f$ denote the set of its adjacent end nodes in the logical tree. Thus in our example, $AE_{f_1} = \{e_1, e_2\}$. Fan-out nodes which have at least 2 adjacent end nodes and only 1 adjacent fan-out node in a logical tree, is said to be a *border* fan-out nodes. We let $BF$ denote the set of border fan-out nodes in the logical tree. For example, in Figure 3, $BF = \{f_1, f_2, f_3, f_6\}$. The notion of a border fan-out node will be useful when we consider "reduced" distance matrices in IV-D.

*Theorem 1:* A logical tree with at least two fan-out nodes has at least two border fan-out nodes, i.e., if $|F| \geq 2$ then $|BF| \geq 2$.

*Proof:* Consider one of the longest paths in the logical tree. Since $|F| \geq 2$, such a path must include at least two fan-out nodes. We argue that the nodes adjacent to the the end nodes of the path must be border fan-out nodes. Suppose one of them is not a border fan-out node. Then there is another adjacent fan-out node which is not currently on the path. This means a longer path than the current one could be constructed and leads to a contradiction. ■

Given an end node $r \in E$ we can consider the $r$-rooted logical multicast tree associated with a multicast session. We shall exhibit such trees with the root is at the top, and nodes that are equally distant from the root horizontally aligned at levels below it. Figure 4 depicts the $r$-rooted logical tree for physical tree in Figure 2.

With the introduction of the root, we can further partition the end nodes, $E$, and the fan-out nodes, $F$, according to their fan-out distances from the root. We let $E_i$ represent the set of end nodes whose fan-out distance from the root is $i$. Similarly $F_i$ denotes a set of fan-out nodes whose fan-out distance from the root is $i$. $E_i$ and $F_i$ are said to be at *level* $i$. Note that $i = 0, 1, \ldots, b$ where $b = \max_{m \in E} d_f(r, m)$. We define $E_0$ as $\{r\}$ and $F_b = \emptyset$. Figure 4 shows the example of such a partition of end and fan-out nodes.

If a node $p$ immediately precedes node $c$ on the path from the root to $c$, then $p$ is the *parent* of $c$ and $c$ is the *child* of $p$. Nodes having the same parent are said to be *siblings*. We let a *sibling set* denote an *exhaustive* collection of siblings sharing the same parent. Note that for a given rooted logical tree there are several types of siblings:

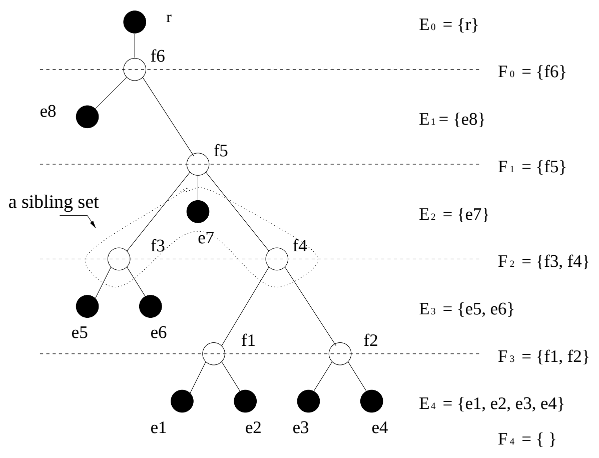

Fig. 4. The $r$-rooted logical tree for our example.

- Type 1 (*Mixed Siblings*): An end node $e$ at level $i$ can be the sibling of a fan-out node at level $i$.
- Type 2 (*Fan-out node Siblings*): Fan-out nodes at the same level can be siblings.
- Type 3 (*End node Siblings*): End nodes at the same level can be siblings.

In Figure 4, the sibling sets $\{f_3, e_7, f_4\}$, $\{f_1, f_2\}$ and $\{e_5, e_6\}$ exemplify these types of relations respectively.

A node $d$ is said to be a *descendant* of a node $n$, if $n$ is on the path from the root to $d$. Note that from the above definition, $n$ can be its own descendant. Given a fan-out node $f \in F$, we define a *reference* node of $f$, denoted by $r(f)$ to be any end node which is a descendant of $f$. Reference nodes will be used in checking sibling relations for fan-out nodes, since there is no explicit information for fan-out nodes in the distance matrix.

Note that the level ordering and filial relationships discussed above are always with respect to a given rooted logical tree. However, for simplicity we have not included the specified root in our notation.

## B. Algorithm using the full distance matrix

In this section we discuss an algorithm to discover a tree given the full distance matrix. The algorithm includes two parts. Based on fan-out distances, one first discovers the logical tree, and then based on path distances, one determines the hop count lengths associated with links in the logical tree. The steps of the algorithm can be summarized as follows:

1. Logical tree discovery:
   - Select a root.
   - Perform a level ordering on end nodes, $E_i, i = 0, \ldots b$.
   - Perform bottom up discovery of $F_i, i = 0, \ldots b$ and sibling/parent relationships among nodes.
2. Physical tree discovery:
   - Perform bottom up discovery of path distances associated with the logical tree's links.

Below we outline the details associated with these steps.

### B.1 Logical tree discovery

The first task is to select a root for the logical tree. In general any node could be selected, however since we intend the discovery algorithm to be carried out in a distributed fashion at each end node we shall assume without loss of generality that each end node considers itself to be the root of the tree. We let $r \in E$ be the root for our ongoing example. Next, we partition the end-nodes into sets $E_i, i = 0, \ldots b$, based on their fan-out distances from the root. This is done by checking $r$'s row in the distance matrix.

The key task in the logical tree discovery step is to progressively identify complete sibling sets in a bottom up fashion. Note that each sibling set is associated with a unique, previously unknown, parent fan-out node at a higher level of the logical tree. Thus we can progressively determine not only $F_i, i = 0, \ldots b-1$ but the filial relations among the rooted tree's nodes. We shall start at the bottom, setting $i = b$. The key step will be at each level $i$, to discover complete sibling sets among $E_i$ and $F_i$ and create the associated set of parent fan-out nodes, $F_{i-1}$, at the next level. The following lemma will enable us to check whether two nodes in $E_i \cup F_i$ are siblings.

*Lemma 1: Sibling Checking Lemma*

1. Suppose $e \in E_i$ and $f \in F_i$ then they are siblings iff

$$d_f(e, r(f)) - d_f(f, r(f)) = 2.$$

2. Suppose $f_a, f_b \in F_i$ then they are siblings iff

$$d_f(r(f_a), r(f_b)) - d_f(f_a, r(f_a)) - d_f(f_b, r(f_b)) = 3.$$

3. Suppose $e_a, e_b \in E_i$ then they are siblings iff $d_f(e_a, e_b) = 1$.

The proof of the lemma is straightforward. In the first case, $e$ and $f$ are siblings iff $d_f(e, f) = 1$, so the lemma follows by noting that we can compute $d_f(e, f)$ based on $f$'s reference node $r(f)$ as $d_f(e, r(f)) - d_f(f, r(f)) - 1$. In Figure 5 sibling nodes $f_4$ and $e_7$ exemplify this case. For the second case, note that $f_a$ and $f_b$ are siblings iff $d_f(f_a, f_b) = 1$. The lemma follows by computing this distance based on reference nodes for associated fanout nodes, i.e.,

$$\begin{aligned} & d_f(f_a, f_b) = \\ & \quad d_f(r(f_a), r(f_b)) - d_f(f_a, r(f_a)) - d_f(f_b, r(f_b)) - 2. \end{aligned}$$

Siblings $f_3$ and $f_4$ in Figure 5 exemplify the second case. The final case is clear and can be easily checked using $e_a$'s (or $e_b$'s) row in the distance matrix.

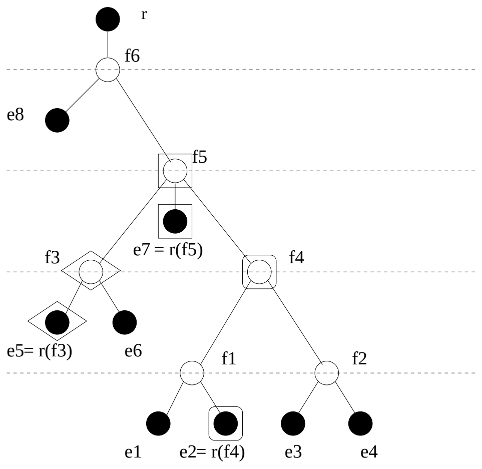

Fig. 5. Illustration of sibling checking criteria.

In order to discover complete sibling sets among the nodes we let $C = E_i \cup F_i$ denote the set of nodes that need to be considered. Select any node $c_1 \in C$ and determine the set of *all* of its siblings $S_1$, including $c_1$, by checking each of the remaining nodes in $C$ using Lemma 1. Now let $C := C \setminus S_1$ and proceed iteratively until there are no more nodes in $C$. Suppose this process terminates after $k$ steps, then $k$ disjoint sibling sets $S_1, \ldots S_k$ are obtained. For each of these, generate a parent node $f_j$, $j = 1, \ldots, k$ and place it in the set $F_{i-1}$ of fan-out nodes at the next level up. Also define the reference node $r(f_j)$ for each parent, $f_j$, to be any end node which descends from $f_j$. At this point one can proceed in discovering siblings and parents at the next level up. This procedure continues until the logical tree topology is determined.

### B.2 Discovery of path distances of logical links

Once we have identified the logical tree, we need only to find path lengths associated with its logical links to determine the physical tree. The key idea is captured by the following lemma, which determines path distances of logical links between a border fan-out node $f \in BF$ and its adjacent end nodes $AE_f$.

*Lemma 2:* Suppose $f \in BF, m, n \in AE_f$ and $k \in E, k \neq m, n$ then

$$\begin{aligned} d_p(m, f) &= [d_p(m, n) + d_p(k, m) - d_p(k, n)]/2, \\ d_p(n, f) &= [d_p(m, n) - d_p(k, m) + d_p(k, n)]/2. \end{aligned}$$

The proof of this lemma follows directly by decomposing path lengths into their constituent components – consider Figure 6. Moreover for any additional node, $e \in AE_f \setminus \{m, n\}$, the path

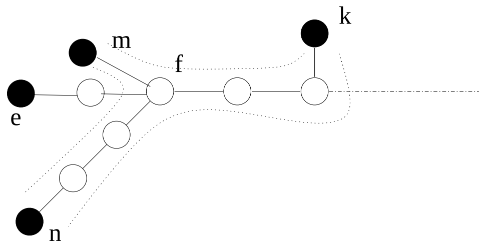

Fig. 6. Path distance calculation at a border fan-out node $f$.

distance $d_p(e, f)$ can be computed to be $d_p(m, e) - d_p(m, f)$. Observe that to determine the lengths of the logical links from a border fan-out node $f$ to all its adjacent end nodes $AE_f$ we only require two rows of the distance matrix, where at least one is associated with one node in $AE_f$.

Note that for any rooted logical tree, if $f \in F_{b-1}$ then $f \in BF$. Thus by Lemma 2 all the lengths for logical links at the bottom level can be computed. In order to proceed systematically in a bottom up fashion, we propose to prune the tree and update the path distance matrix. At level $i$, all links and end nodes $E_i$ whose distance to their parents have been computed are pruned. Then all fan-out nodes at level $i-1$, i.e., $F_{i-1}$, became end nodes at level $i-1$. In this pruned tree, all $f \in F_{i-2}$ are border fan-out nodes, which guarantees that the path distance calculation step can again be performed for level $i-1$.

As a result of pruning, the path distance matrix for the new tree must be generated. This is done by eliminating entries associated with all the pruned end nodes, and adding a new entry, for each fan-out node $f$ that becomes an end node of the new tree. Table III is the path distance matrix for the pruned tree in Figure 7.

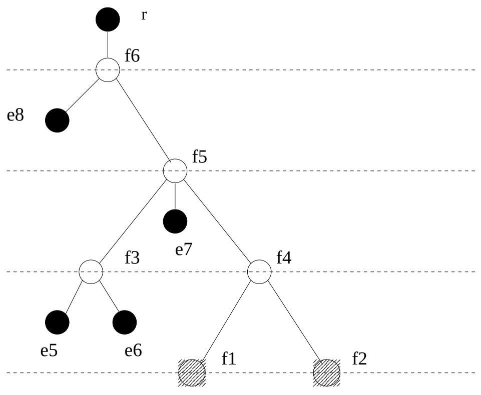

Fig. 7. The pruned tree of Figure 4 at Level 4.

TABLE III
PATH DISTANCE MATRIX FOR THE TREE IN FIGURE 7.

| | $r$ | $f_1$ | $f_2$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |
|---|---|---|---|---|---|---|---|
| $r$ | | 6 | 6 | 5 | 6 | 4 | 3 |
| $f_1$ | | | 2 | 5 | 6 | 4 | 7 |
| $f_2$ | | | | 5 | 6 | 4 | 7 |
| $e_5$ | | | | | 3 | 3 | 6 |
| $e_6$ | | | | | | 4 | 7 |
| $e_7$ | | | | | | | 5 |
| $e_8$ | | | | | | | |

### *C. Computational complexity*

The computational complexity for the proposed algorithm can be roughly evaluated as follows. The level ordering step is $O(|E|)$. The bottom up step in the logical topology discovery phase can be shown to be $O(|E|^2)$. Indeed there are at most $|E| - 2$ fan-out nodes in the tree and determining siblings associated with each parent fan-out node has a cost of at most $|E|$. Path distance computations to obtain the physical topology are also quadratic. So the overall computational cost is $O(|E|^2)$.

### *D. Reducing the required distance information*

There is in fact a large amount of redundant information in the distance matrix. This motivates us to ask the following question: What is the minimal required distance information in order to discover a tree? To answer this question, we will define our unit of information as an end node's entire row table which includes path/fan-out distances from the end node to all other end nodes in a tree. Let $NE$ denote the set of end nodes whose row tables are available when performing topology discovery. Our goal is to find a reduced set $NE$ such that the topology of the multicast session can still be determined. Note that the algorithm described in Section IV-B requires the full distance matrix, i.e., $NE = E$.

*Theorem 2:* Given a shared multicast tree with $|F|$ fan-out nodes the following conditions on the set $NE$ of available rows in the distance matrix are sufficient to allow topology discovery:
1. If $|F| = 1$ then $|NE| \geq 2$.
2. If $|F| \geq 2$ then $NE$ should include at least one node in the set of end nodes $AE_f$ associated with each border fan-out node $f \in BF$.

*Proof:* Consider the first case. If $|F| = 1$, the discovery of the logical topology is straightforward, i.e., all nodes are 1 fan-out distant from each other. This can be determined based on a single row table. Note that by Lemma 2 if two row tables are available, one can compute all path distances from a fan-out node to its adjacent end nodes. This establishes the condition for the first case.

Now suppose that $NE$ includes one node from each set $AE_f$ associated with border fan-out nodes $f \in BF$. We show that the logical topology can be determined as follows. Select any node $r \in NE$ as the root and perform a level ordering on end nodes based on $r$'s row table. Note that during our bottom up phase, we will be able to assign a reference node in $NE$ to each generated fan-out node, since every fan-out node in a rooted logical tree, has at least one border fan-out node as its descendant. This guarantees that all the required information is available to use Lemma 1 for sibling checking.

Next we show that subject to given conditions, the physical topology can also be discovered. Note that by Theorem 1, if $|F| \geq 2$ then $|NE| \geq 2$. Recall that by Lemma 2, in order to know the path lengths associated with logical links from a border fan-out node, e.g., $f$, to its adjacent end-nodes, we only need two row tables of which at least one node should be in $AE_f$. Since $NE$ contains at least one in $AE_f$, and $|NE| \geq 2$, all path distances to $f$ can be computed. The path length computation can once again be carried out by pruning, starting from the bottom level to the top. ■

Note that the computational complexity of topology discovery based on the reduced distance matrix remains $O(|E|^2)$.

## V. OBTAINING DISTANCE INFORMATION

In this section we discuss implementation issues concerning how members of the multicast session can selectively acquire sufficient distance information to discover the topology of the multicast tree. The elements necessary in our proposed framework are:
1. Fan-out decrement mechanism at multicast routers.
2. Initial path/fan-out field in packets for allowing a receiving host to obtain distance information from the sender to itself.
3. Shared multicast tree for preserving path symmetry between members.[4]

Assuming that the above requirements are satisfied, first, we discuss how each member can obtain the full distance matrix. Suppose every member periodically multicasts a *heartbeat* packet to the whole group. The role of the heartbeat packet is two-fold: 1) it serves as an indication of the liveness of the sending host, which is necessary if the algorithm is to adapt to changing membership or topologies; and 2) it enables receiving members to obtain their fan-out/path distances from the sender. Note

[4]To create shared multicast trees, CBT or PIM-SM [21] might be used. However, note that PIM-SM is not applicable to our model. This comes from the fact that in a PIM-SM shared tree mode, the sender's packet goes to the core first and then the core multicasts it to the others. Thus there is no way for each member to acquire other members' distance information.

that senders which persistently multicast data packets to the session may not need to send heartbeat packets, as long as initial values for the TTL and fan-out fields are included in the IP multicast packet's header. Whenever a member receives a heartbeat from other members, the member can build/update its row in the session's distance matrix, where each member is identified by its IP address. In addition to periodically sending heartbeat packets, each member becomes a *reporter* and periodically multicasts a *report* packet to the session which contains its own row table. Thus, eventually each session member would have access to the full distance matrix.

Theorem 2 suggests that it would suffice for only one node among adjacent members of each border fan-out node to generate report packets. The above approach has two advantages over the full distance distribution method. First, it reduces the number of reporters in a session, which results in significant reduction of communication overheads since report packets can be large relative to heartbeat packets. Second, it can also reduce memory storage space required at end-hosts. In order to enable this type of reporting, one must however identify border fan-out nodes, and then select a unique reporter for such a node. This in itself requires that the network topology be known a priori, which is not practical.

As a compromise between full distance matrix distribution and the impractical second approach discussed above, we propose the following rules to determine which end hosts should serve as reporters:

*Rule 1:* A member will serve as a reporter if there is at least one other member which is 1 fan-out distant from it and it has the smallest IP address.

*Rule 2:* A member will serve as a reporter if all other members in a session are 1 fan-out distant from it and it has the largest IP address.

Note that the first rule guarantees that there will be a reporter selected from set of adjacent members to a border fan-out node – there may also be some additional reporters. The second rule ensures that if the tree has but one fan-out node, there will be at least 2 reporters. Thus with these two rules enforced, the sufficient conditions stated in Theorem 2 will be satisfied.

Note that these rules can be applied by nodes in a decentralized fashion in that they need only to check their own row table without any computation. This approach would of course reduce network traffic to acquire the required distance information. Also note that in this context the minimum number of reporters is 2 while the the maximum number of reporters is $\lfloor |E|/2 \rfloor$.[5] In general the communication complexity to acquire the distance matrix would be $2|E|$ multicast messages, i.e., a heartbeat and report packet per session member, where the size of heartbeat packets is $O(1)$ while that of reports is $O(|E|)$.

## VI. Local topology discovery framework

If a multicast session involves a huge number of members, the proposed global topology discovery scheme may not be workable. In particular, the communication, computation and storage overheads may be unwarranted.

Note also that since each row in the distance matrix contains each members' IP address, for large multicast trees this may include a lot of data, eventually requiring reports to be partitioned across several packets.[6] Moreover, in a large scale multicast session, members may not be interested in discovering the entire distribution tree. Instead they may only be interested in a local view of the multicast tree's structure. This is, for example, the case in the context of applications for local loss recovery where members only wish to identify other members within a given neighborhood. Thus it would be advantageous if the proposed framework could also be used to discover a restricted local topology while reducing the overheads associated with acquiring this information.

Let us consider an instance of this problem for a session member $r \in E$. Let a *neighborhood* $N_r$ be the set of members that share a particular attribute, including $r$ itself. Note that there is quite a bit flexibility in defining $N_r$. For example the neighborhood could correspond to $FN_r^k$, the set of members within the k fan-out scope from $r$ including $r$ itself, or the set of members that serve as DNS servers and are in $FN_r^k$. Given such a neighborhood, we define the *induced physical and logical trees* as follows.

*Definition 1:* Given a neighborhood $N_r \subset E$ of a node $r \in E$ in a multicast tree, we let the $N_r$ *induced physical tree* be the subtree connecting $r$ to the members of its neighborhood $N_r$. We define the $N_r$ *induced logical tree* as the logical tree associated with the $N_r$ induced physical tree.

For example, consider the neighborhood $N_r = \{r, e_5, e_8\}$ of $r$ in a physical multicast tree shown in Figure 8. The region that has been outlined corresponds to the physical tree induced by $N_r$ while Figure 9 depicts the $N_r$ induced logical tree.

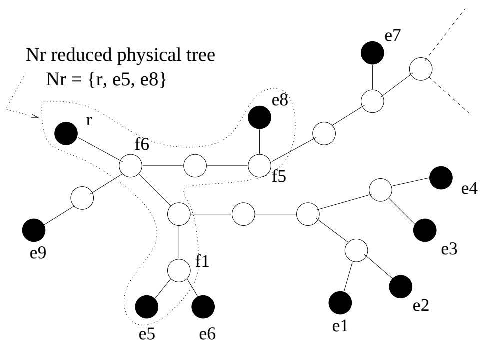

Fig. 8. A physical multicast tree.

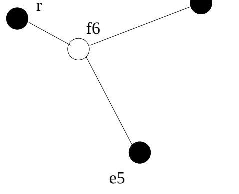

Fig. 9. The $N_r$ induced logical tree. ($N_r = \{r, e_5, e_8\}$)

[5]The notation, $\lfloor\ \rfloor$, is a floor operator.

[6]In order to reduce communication overheads, one might consider reports that include only incremental changes in data. This must, however, be done with care in a dynamic scenario as new members need to eventually acquire sufficient information to discover the tree.

Note that an $N_r$ induced logical tree simply shows the logical relationship among members in $N_r$, and it might include logical links that hide fan-out nodes in the global multicast tree. For example, the logical link from $f_6$ to $e_5$ in Figure 9 actually represents 3 physical links and 2 fan-out nodes.

*Definition 2:* Given a neighborhood $N_r \subset E$ of a node $r \in E$ in a multicast tree, the *local multicast topology discovery of* $N_r$ is defined as determining the $N_r$ induced logical tree topology, as well as path/fan-out distances for its logical links.

Local topology discovery can be based on an $N_r$ *restricted* distance matrix including only row and column entries associated with the nodes in $N_r$. This problem can be viewed as a restricted version of the global topology discovery problem presented in Section IV. It is relatively easy to see that one can, with some care, apply the same methods developed for global topology discovery in this context.

We propose to perform local topology discovery by first determining the $N_r$ induced logical topology applying the algorithm in Section IV-B.1. In this step, we in fact determine the subtree induced by $N_r$ on the *global logical topology*. This is illustrated in Figure 10 for the local topology discovery problem associated with $FN_r^3$ in the multicast session Figure 8. Note that the subtree enclosed in the dashed line need not be the desired $N_r$ induced logical tree. In particular, the subtree obtained by using our previous algorithm on the restricted set may include fan-out nodes, e.g., $f_3$ and $f_4$ in the above example, which would not be part of the $N_r$ induced logical topology, see Figure 11. Once such nodes are pruned, the structure of the $N_r$ induced logical tree has been discovered along with the fan-out distances associated with its logical links.

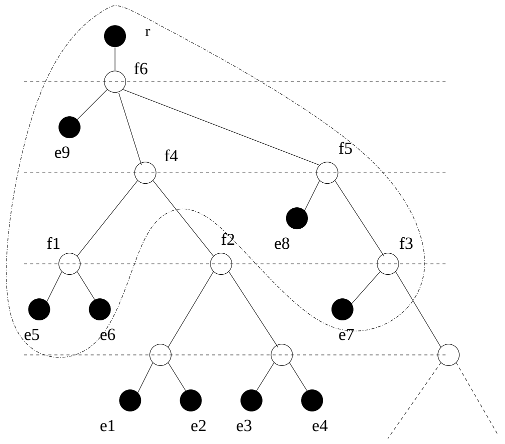

Fig. 10. The $r$ rooted logical tree of Figure 8.

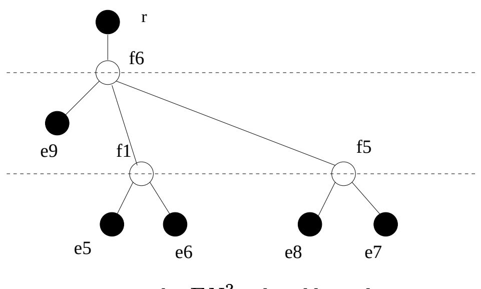

Fig. 11. The $FN_r^3$ induced logical tree.

Next, based on Lemma 2, one can identify the path distances of the logical links in the $N_r$ induced logical subtree. Note that certain path metrics would not, and in fact can not, be identified based on the $N_r$ restricted distance matrix. For example, node $f_4$ is not present in the induced logical subtree, and thus the path lengths $f_1$ to $f_4$ and $f_4$ to $f_6$ would not be determined, however the overall path metric associated with the logical link from $f_1$ to $f_6$, can be identified.

In summary, discovering an $N_r$ induced logical tree's topology and the associated logical links' distances requires basically the same steps as discussed for the global case. It should be clear that the computational complexity of local topology discovery is quadratic in the size of the neighborhood, and storage requirements would also depend on the size of the the neighborhood.

In principle a neighborhood can be any set of members sharing a particular attribute. However, below we will focus on local topology discovery, i.e., that associated with neighborhoods having spatial proximity on the multicast tree. Thus we will define both fan-out and TTL scoped neighborhoods for a given node. We let $TN_r^l$ denote the set of members in a multicast group that are within an $l$ limited TTL scope from $r$ including $r$ itself. In general one can define a jointly scoped neighborhood, e.g., $N_r = FN_r^k \cap TN_r^l$, for each node in a network and proceed to discover the induced logical trees based on restricted distance matrices.

The remaining question is how each node would acquire the restricted distance matrix associated with its $k$ fan-out and $l$ TTL scoped neighborhood. The following simple protocol suffices:

1. Each member periodically sends heartbeat packets with the fan-out scope set to $2k - 1$ and the TTL scope set to $2l - 2$.
2. Each member periodically sends a report packet with $k$ and $l$ set as the fan-out and TTL scopes respectively.

The idea underlying this scheme is quite simple. First, each node $r$ should receive reports from all members of its neighborhood, thus report packets should be scoped as indicated above. Second, since an $N_r$ restricted distance matrix contains path and fan-out distances among all pairs of members in $N_r$, they have to know of each other's existence and the associated distances. Note that $2k - 1$ and $2l - 2$ are the maximum possible fan-out and TTL distances between members in $N_r = FN_r^k \cap TN_r^l$. Thus it should be clear that the proposed fan-out and TTL scopes on heartbeat packets ensure that the $N_r$ restricted distance matrix acquired by a node $r$ is complete. Note that we have assumed an a priori uniform selection of $k$ and $l$ for all nodes. This poses the question of how they might be 'optimally' chosen and whether they might be selected in a non-homogeneous decentralized fashion. This would of course depend on applications.

Assuming nodes share information in this fashion, one can significantly reduce the communication overhead associated with topology discovery, in terms of the number of heartbeat and report packets seen on *any* link in the multicast tree and the size of the report packets. Indeed, although the same total number of packets, $2|E|$, will be sent as in the global discovery case, these packets are scoped and hence will not be seen by all links and members. In particular, a rough estimate for the number of messages seen by a member would be the size of its $2k - 1$ fan-out and $2l - 2$ TTL scoped neighborhood. Similarly the size of report packets would is no longer be $|E|$ but proportional to the size of the neighborhoods.

## VII. Conclusions and Future Work

The proposed framework for both global and local topology discovery of shared multicast trees has significant advantages over current approaches, particularly in terms of simplicity, adaptability and scalability. It is based on the addition of a new fan-out decrement mechanism to IP Multicast functionalities. However this new service is simple to implement, and provides an efficient way of scoping in the context of IP Multicast sessions. Furthermore, as the use of multicast sessions and applications becomes increasingly widespread, the possible benefits resulting from the availability of topological information may warrant the addition of this mechanism.

In this paper, first we propose an algorithm which can discover the topology of the shared multicast tree based on a full distance matrix, with the analysys of computational complexity. Second, we provide sufficient conditions to achieve the same result with a reduced distance matrix. Third, we show how reduced distance information could be acquired efficiently by exchanging a small number of multicast packets with an analysis of explicit communication overheads, i.e., the minimal number of packets injected in the network and the size of the packets. Finally, we consider both concepts and practical issues in the context of local topology discovery enabling nodes to discover the distribution tree within their fan-out and TTL scoped neighborhoods.

Future work on the topology discovery problem will focus on additional implementation issues, e.g., the impact of packet losses and incomplete distance matrices on discovery, or additional reductions in overheads achieved via incremental updates. In addition, we are considering more scalable approaches to global topology discovery, involving the use of local discovery combined with hierarchical distribution of topology information. The applications that we have in mind for these techniques, involve enhancing network utilization by partitioning or clustering multicast session members, distributed algorithms based on local topology, and methods for assessing the actual network resource costs to support multicasting.

## References

[1] S. E. Deering, *Multicast Routing in a Datagram Internetwork*, Ph.D. thesis, Stanford University, 1991.

[2] B. N. Levine, S. Paul, and J. J. Garcia-Luna-Aceves, "Organizing multicast receivers deterministically according to packet-loss correlation," in *Proceedings of ACM Multimedia'98*, 1998.

[3] D. Li and D. R. Cheriton, "OTERS(On-Tree Efficient Recovery using Subcasting): A reliable multicast protocol," in *Proceedings of IEEE ICNP'98*, 1998.

[4] J.C. Lin and S. Paul, "RMTP: A reliable multicast transport protocol," in *Proceedings IEEE Infocom 1996*, 1996.

[5] I. Kouvelas, V. Hardman, and J. Crowcroft, "Network adaptive continuous-media applications through self-organised transcoding," in *Proceedings of the Network and Operating Systems Support for Digital Audio and Video*, 1998.

[6] C. Papadopoulos, G. Parulkar, and G. Varghese, "An error control scheme for large-scale multicast applications," in *Proceedings IEEE Infocom 1998*, 1998.

[7] S. Ratnasamy and S. McCanne, "Scaling end-to-end multicast transports with a topologically-sensitive group formation protocol," in *Proceedings IEEE International Conference of Network Protocols, ICNP'99*, 1999.

[8] W. Fenner and S. Casner, "A traceroute facility for IP Multicast," IETF draft, draft-ietf-idmr-traceroute-ipm-06.txt, March, 2000.

[9] S. Ratnasamy and S. McCanne, "Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements," in *Proceedings IEEE Infocom'99, New York*, 1999.

[10] X. Yang and L. W. Lehman, "Inferring characteristics of multicast trees," Dec 1998. MIT 6.896 Topics in Computer Networks term project paper.

[11] R. Caceres, N.G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley, "Loss-based inference of multicast network topology," in *Proceedings 1999 IEEE Conference on Decision and Control*, 1999.

[12] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet map discovery," in *Proceedings IEEE Infocom 2000*, 2000.

[13] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, "On the placement of Internet instrumentation," in *Proceedings IEEE Infocom 2000*, 2000.

[14] R. Siamwalla, R. Sharma, and S. Keshav, "Discovering Internet topology," www.cs.cornell.edu/skeshav/ papers/discovery.pdf.

[15] Y. Breitbart, M. Garofalakis, C. Martin, R. Rastogi, S. Seshadri, and A. Silberschatz, "Topology discovery in heterogeneous IP networks," in *Proceedings IEEE Infocom 2000*, 2000.

[16] M. Stemm, R. Katz, and S. Seshan, "A network measurement architecture for adaptive applications," in *Proceedings IEEE Infocom 2000*, 2000.

[17] T. Ballardie, P. Francis, and J. Crowcroft, "Core based trees(CBT): An architecture for scalable inter-domain multicast routing," in *Proc. ACM SIGCOMM'93*, 1993.

[18] A. Ballardie, "Core Based Trees(CBT) Multicast Routing Architecture," RFC 2201, Sep., 1997.

[19] B. Cain, S. Deering, and A. Thyagarajan, "Internet group management protocol, version 3," IETF draft, draft-ietf-idmr-igmp-v3-04.txt, June, 1997.

[20] S. Keshav, *An Engineering approach to computer networking:ATM Networks, the Internet, and the Telephone Network*, Addison-Wesley, Inc., 1997.

[21] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification," RFC 2362, Jun., 1998.