Guided Shared Trees for Efficient Multicast in Large Networks

Nian-Feng Tzeng and **Prasanth Alla**

Center for Advanced Computer Studies
University of Louisiana at Lafayette
Lafayette, LA 70504

Abstract

Data delivery to multiple recipients in networks can be achieved effectively via multicast based on a shared tree structure. This paper deals with a network-layer framework for efficient multicast through shared trees which are developed with an aid of guided information for locating nearest known on-tree nodes to connect upon receiving group join requests. Such information helps to shorten end-to-end delay over a multicast shared tree so developed, called a guided shared tree (GST). A designated node, known as a guidance information node (GIN), is employed to provide guidance for a group, and the GIN is found by joining nodes through the group ID indexing into a list of candidate GINs, in a way similar to automatic core discovery for CBT. The proposed multicast framework is evaluated by simulation and is shown to be efficient, readily suitable for large networks where group members account for a fraction of total nodes and are sparsely located.

I. Introduction

Multicast usually constructs delivery trees to span all the group members, formed by keeping multicast routing tables at involved routers (referred to as tree nodes), one table entry for a tree. Hosts join multicast groups through their respective first-hop routers following the IGMP protocol [6]. Multicast trees can be built separately for different sources, like DVMRP (Distance Vector Multicast Routing Protocol) [16] or MOSPF (Multicast Extensions to OSPF) [14], or be shared by all sources of the same group, like PIM (Protocol Independent Multicast) [7] or CBT (Core-Based Trees) [1]. PIM has a provision in creating a shortest-path tree for any source which originates a large data volume exceeding a certain threshold. An IP multicast protocol able to support unicast transparently has been considered lately [4], facilitating progressive deployment of multicast by supporting unicast clouds.

It is preferred to build one shared multicast tree to reach all group members, irrespective of multicast sources, for better scalability. However, a single shared tree presents several possible problems. One known problem lies in its traffic concentration around the *core* of CBT or the rendezvous point (*RP*) of PIM-SM [9] as the tree expands, potentially prolonging the end-to-end transmission delay. To alleviate this concentration problem, multiple active cores for a shared multicast tree under CBT have been considered and evaluated [18]. The other problem results from the fact that when members are added to a shared tree constructed by any existing protocol for multicast delivery (like CBT or PIM-SM), the tree expands without guidance, disregarding nearby on-tree nodes; this way often results in a tree with a significant longer delay for data delivery than that in a tree whose growth is properly guided. For example, every group member for CBT makes use of a *shortest* path to the core (or one chosen core when multiple of them exist [18]) based on the unicast routing table entry corresponding to the core, disregarding nearby on-tree nodes which are not on the shortest path to the core. This can be observed in Fig. 1, where node *C* is assumed to be the core and node *D* is the only group member in the shared tree. When node *H* is to join the tree later, it is done through the shortest path to *C*, say, *H-G-F-C*, forming a shared tree comprising 6 on-tree nodes. This shared tree so formed is obviously less efficient than the one constructed by connecting nodes *H* and *D* directly, which can be realized by proper guidance when members join the tree.

This article deals with a network-layer framework for guiding shared multicast tree construction in an attempt to yield efficient information delivery trees, called *guided shared trees* (GSTs). A GST improves efficiency by keeping track of on-tree nodes so that any subsequent group member is joined by connecting it via a shortest path to a *nearest known* on-tree node (rather than a fixed node, like the core for CBT or the RP for PIM-SM). It is particularly advantageous in a large network where group members are sparse and account for a fraction of total network nodes, a rather common situation. This proposed framework can be realized by means of different tree guidance approaches, leading to various GSTs for a given multicast group in a network. While this work focuses on its application to intra-domain shared tree construction, the framework may also be applied to build inter-domain shared trees effectively, leading to better delivery trees than those bi-directional shared trees constructed according to BGMP [13]. Our framework requires a designated node, known as a *guidance information node* (GIN), for each group to keep track of on-tree nodes in order to provide guidance for desirable tree expansion (i.e., shaping the shared tree) when subsequent group members join. A GIN provides a small set of

This work was support in part by the National Science Foundation under Grants EIA-9871315 and CCR-0105529, and by the Board of Regents of the State of Louisiana under Contract No. LEQSF(2000-2001)-ENH-TR-90.

candidate on-tree nodes (i.e., guidance) for a joining member to choose its best possible connecting node on the shared tree. The set of candidate nodes is decided quickly according to proximity hints (PH) of on-tree nodes. Performance outcomes of GSTs are gathered using simulation and they are indeed superior to those of their CBT counterpart, particularly in a large network where a small fraction of nodes are group members sparsely located.

The proposed framework comes with extra traffic overhead for gathering on-tree node information at a GIN and a possibly longer delay in the node joining process. Such traffic and run-time overhead, however, can be alleviated and adjusted to any desired level through three rectification steps by employing multiple GINs for each group. These steps take advantage of the fact that on-tree node information kept at GINs need not be always current or complete to make our framework function properly. Details of these steps are given in Section III. Unlike several application-layer techniques for identifying near peers with certain properties [5, 12], which require participants to perform repeated measurements of their distances from some selected points and/or to receive lists of candidates from multiple selected points, our network-layer framework calls for a joining node to communicate with merely one selected point (i.e. GIN) and only once, if at all, to receive helpful guidance. Our framework needs additional support from the infrastructure and depends upon unicast forwarding. Like an early QoS-sensitive multicast protocol, QoSMIC [11], our framework allows a GIN to be outside the shared tree serving as one centralized node to provide information (i.e., guidance) for tree growth management.

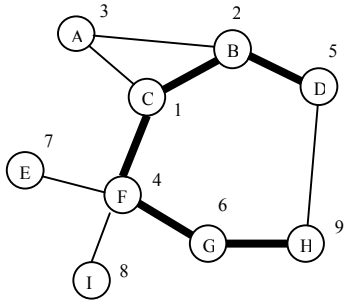

Fig. 1. Shared tree formed with core C and group members D & H under CBT, shown by bold line segments. (A number next to each node is its location indicator, LI.)

II. Related Work

Shared multicast trees are preferred over shortest-path per-source trees due to lower space overhead (for maintaining tree states at each on-tree nodes) and better scalability. In particular, a CBT shared tree spans merely the group members and incurs no cost or overhead in other parts of the network. Work has been carried out on core placement and its impact on the performance of shared trees. Specifically, the average performance of shared trees built when the core is placed optimally with respect to all nodes and to all members of a group has been assessed [17]. Naturally, the selection of an optimal core is hard if group membership changes over time, and the performance of a shared tree with an optimally placed core tends to degrade after membership changes. A later study considered randomly placing the core for each group to alleviate traffic concentration and found that such a random, per-group placement was effective, particularly when group members are localized [3].

Every on-tree router in CBT maintains group-specific state information, which involves the group address (of 28 bits for IPv4) and a list of local interfaces over which join messages for the group have been acknowledged previously [2]. On receiving a multicast data packet, it forwards one copy of such a packet over each interface on the list except the interface from which the packet arrives. Cores may be either discovered automatically by a "bootstrap" mechanism or placed manually. The former is applicable only to intra-domain core discovery and requires all routers within the domain to implement the "bootstrap" protocol or, at least, forward bootstrap protocol messages. It allows a subset of routers in the domain to be configured as CBT candidate cores. Each candidate core advertises itself periodically, and those advertised cores are collected as the candidate core (CC) list, which is unicast forwarded in the domain [2]. A local router discovers which core to use for a given group using the group ID to index into the CC-list. Automatic core discovery has been specified with the PIM-SM protocol [9]. Manual placement of cores, on the other hand, imposes a degree of administrative burden but is likely to result in better shaped trees and is the only available option for inter-domain core discovery. Our framework for guided shared trees may follow an approach similar to automatic core discovery for group members to identify the GINs, as will be described later.

III. Guided Shared Trees for Multicast

This work focuses on multicast within a domain (i.e., an administratively-independent network). Our efficient intra-domain multicast is realized by constructing shared trees to span all members of a group with the aid of proximity hint (PH) which enables a joining node to choose a *best* (i.e., *closest*) *known* on-tree node to connect. This PH shapes shared tree expansion to yield an efficient delivery tree, called a guided shared tree (GST), which is superior to that resulting from joins always to a fixed point (like the core of CBT or the RP of PIM-SM). Arriving at GSTs requires the use of guidance information nodes (GINs) to keep track of on-tree nodes and their associated PH. Here, we assume the existence of a single GIN for each multicast group first, followed by an extension to multiple GINs for a group. The set of on-tree nodes is kept at the GIN

corresponding to group γ, denoted by Ωγ, which is empty initially. When a node, say N_x, intends to join γ, it performs steps below in sequence.

1. Requesting the GIN to provide a small number of candidate on-tree nodes selected in Ωγ, where candidate on-tree nodes are those most closest ones to N_x based on PH used.
2. Looking up its unicast routing table to find the distances from N_x to candidate on-tree nodes and chooses the nearest one, say N_t.
3. Sending a join request to N_t through unicast forwarding until it reaches either N_t or the first on-tree node on the way, with all the visited nodes (i.e., routers) and their associated PH recorded in a list.
4. The node where this join request stops is responsible for transmitting the recorded list (of new on-tree nodes plus their associated PH) to the GIN, which then adds the list to its Ωγ.

Note that the GIN needs to provide multiple candidate on-tree nodes for the joining node to choose, according to Step 1. This is mainly because the on-tree node chosen as the *closest* using PH is not necessarily the actual closest one, as PH does not always reflect distances between pairs of nodes faithfully. In Fig. 1, for example, every node is given a location indicator (LI, denoted by a number placed next to the node) and PH of nodes is reflected by their LI's; namely, node D (with LI = 5) is closer to node F (with LI = 4) than to node B (with LI = 2). Details of LI assignment to nodes in a network will be given in Section IV. In this figure, we assume B, C, and F to be on-tree nodes and D to join. If this PH is followed to select only a single candidate on-tree node in Step 1, F is chosen, despite that B is actually closer to D than F. Our simulation has revealed that a small set of candidate nodes suffices for networks of up to 150 nodes.

When a node (i.e., router), say N_e, is removed from the GST (after its last member leaves the group), it notifies the GIN of this removal directly. Alternatively, this notification can be forwarded to the upstream connected node of N_e (with respect to the core or the RP of the shared tree). This is because the connected node may in turn be removed, and this notification can then accumulate all removed on-tree nodes trigger by a member leave before being sent to the GIN so as to lower traffic overhead.

Different groups may resort to different GINs, which are indexed using the group IDs in a way similar to the automatic core discovery stated earlier (in Section II) for CBT. A GIN serves to offer PH in response to joining requests. Implementation of PH will be detailed in the next section. Note that if the GIN replies joining requests with all on-tree nodes for group γ (kept in Ωγ), "best" GSTs result, at the expenses of far larger sets of candidate on-tree nodes. The use of PH lowers the set sizes (and thus traffic overhead) substantially (possibly by an order of magnitude) without compromising GST quality much.

Multiple GINs

Multiple GINs may be assigned to each group for preventing such problems as a single-point failure and long delayed responses to GIN requests. When multiple GINs exist, it is desirable to have the *same* number of GINs for every group. Let each group be assigned with α (> 1) GINs and the group ID be employed to index into a list of candidate GINs (similar to the CC-list for automatic core discovery described in CBT). Such a list and α both are known to all nodes (i.e., routers) in the domain. The GINs for a given group ID are those α *consecutive elements* in the list, starting from the GIN indexed by the ID, with the list treated as a circular one whose last element is followed by the first element. A joining node selects one of those α GINs to seek PH, preferably the closest one (by checking its unicast routing table for this selection). It is referred to as the first *rectification step* for GST construction, aiming to lower run-time overhead in the join process. Reports about GST changes, however, need to be forwarded to all α GINs, causing more traffic overhead in gathering PH.

Two other *rectification steps* can be employed to reduce *both* traffic and run-time overhead, at the expenses of less PH accuracy. One step calls for gathering on-tree node information in a two-phase process, as opposed to having all α GINs gather entire on-tree node information directly for the given group. Specifically, each node for a group selects its closest GIN (among α GINs) to report its on-tree node updates, and those update reports are sent to that single GIN only. Periodically, one designated GIN for a group *g*, GIN_d^g, collects all the updates (since last collection) from the remaining (α-1) GINs for group *g*, before distributing the complete updates back to those (α-1) GINs. Note that GIN_d^g can be decided in a distributed manner by those α GINs for group *g*, say the one indexed by group *g*'s ID into the candidate GINs list, i.e., the first GIN among those α consecutive ones in the list). Update collection may also be done as needed (on-demand) when a GIN notices a large number of updates. The GIN then sends its updates to all the remaining GINs directly. Clearly, this rectification step reduces traffic overhead involved in PH gathering at GINs.

Another rectification step is invoked when the number of active members for a group exceeds the predefined threshold, decided by those GINs for the group. At that time, each GIN selects the on-tree node closest (based on PH) to itself and uses that selected node in its replies to subsequent join requests received by the GIN. In other words, any GIN of the group will provide only the selected node as PH for all subsequent requests to join the group (and the node is treated like a core of CBT, but just for those nodes covered by the GIN, where coverage is determined by the joining nodes themselves individually based on distance information kept in their unicast routing tables). From that time onward, there is no need to gather on-tree node updates any more, totally avoiding traffic overhead.

IV. PH Implementation

The section states PH (proximity hints) implementation details. Every node in the domain performs (before any multicast starts) the procedures outlined next for establishing PH necessary to enable our proposed framework. The same PH holds for every group and remains unchanged until the network gets modified by adding or dropping nodes or links.

PH is realized initially by assigning contiguous integer numbers, called location indicators (LIs), to those nodes which are in vicinity. In Fig. 1, for example, the LI assignment starts from node C, with its three immediate neighbors (nodes B, A, and F) assigned with LI = 2, 3, and 4, respectively. The neighbors of B (whose LI is the smallest after C's), if any, are examined and assigned with the next numbers, if they are without LI yet. Node D is thus assigned with LI = 5. Next, F's neighbors are checked and assigned with appropriate LIs, e.g., G, E, and I with LI = 6, 7, and 8, respectively. This assignment process repeats until all nodes get their LIs. A procedure is given below to handle this assignment. It requires a list (which keeps such neighbors of all examined nodes that have no assigned LI yet), called the NtA list, and a counter, Γ (which indicates the next available number for assignment). With NtA initialized to an empty list and Γ to 1, this procedure may start from an arbitrary node in the domain. Adjacent neighbors of an examined node are put in NtA in sequence and nodes in NtA are handled in the first-in first-served manner. The node, denoted by N_{ex}, executes this procedure to gets its LI.

1. Assign Γ to N_{ex} as its LI, increment Γ by 1, and then put those N_{ex}'s immediate neighbors without LI to NtA in sequence.
2. If NtA is not empty, its first element in is the next node to execute this procedure, denoted as N_{ex}, to which NtA and Γ are sent over; otherwise, stop.
3. On receiving NtA and Γ, N_{ex} removes the first element (which is the node itself) from NtA, then go to 1.

This procedure is run at each node in the domain at most once, and every node is assigned with a unique LI. As expected, the initial assignment of LIs to nodes may not reflect node proximity well enough, possibly representing unsatisfactory PH when the domain is large. Fortunately, the quality of PH can be enhanced by taking the weighted average of LIs of a node and all its immediate neighbors, according to the next expression:

$$w_l \times LI_l + \Sigma(w_i \times LI_n^i), \quad (1)$$

where w_l and LI_l are the weight and the LI of the node (N_l), and w_i and LI_n^i are the weight and the LI of its i^{th} neighbor (N_n^i). Normally, w_l is chosen to be 0.6 (or large) and every w_i may be given with an identical value of $(1-w_l)/k$, where k is the number of neighbors. As an example, Fig. 2 depicts the enhanced LI assignments after one iteration (i.e., each node re-computes its LI once), with w_l = 0.6 and w_i = $(1-w_l)/k$. Every re-computation iteration initially improves PH representation considerably, as can be seen in Fig. 2, where the LI assignment after the second iteration is given inside the parentheses.

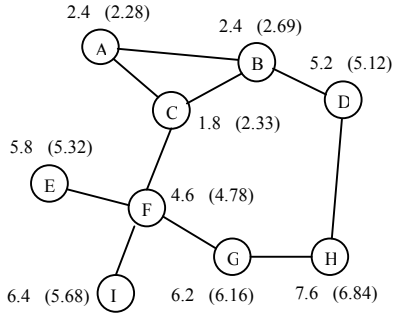

Fig. 2. LIs assigned to nodes after one iteration (and two iterations, given in parentheses)

The quality of PH representation for a node, N_l, can be expressed by:

$$\Delta(N_l) = |\ \Sigma(LI_n^i) - k \times LI_l\ |, \quad (2)$$

where LI_n^i is the LI of the N_l's i^{th} neighbor, $1 \le i \le k$. A smaller Δ signifies better PH representation. Overall PH quality of a network is indicated by the sum of all nodes' Δ values. This LI re-computation often takes just a few iterations to reach reasonably good PH representation. Note that LIs after re-computation this way may no longer be integers and be unique. This poses no problem to our framework for GST construction. In fact, when a node is added to the domain, one may assign the node with the LI of any its neighbor and, then, apply a weighted average expression to get a good LI assignment for the new node immediately.

The Δ results as a function of iteration number for two sample topologies (among many generated for evaluation, discussed in Section V) of size 50 and 150 is shown in Fig. 3, where w_l = 0.6 and w_i is identical to each other and the Δ values are normalized with respect to (i.e., divided by) the network size. As somewhat expected, PH quality in both networks quickly improves in the first few iterations and levels off gradually. In general, the number of iterations needed to obtain reasonably good PH quality depends on w_l and w_i, but it is often small for appropriate chosen w_l and w_i.

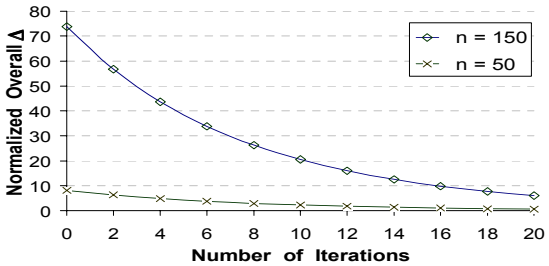

Fig.3. Normalized Δ versus iterations for n = 50 and 150.

Under this PH, each GIN of a multicast group maintains a sorted list of on-tree nodes according to their LIs assigned. On receiving a joining request from a node (say,

with LI = ε), the GIN searches the sorted list to compile a small set of candidate on-tree nodes, whose LIs are most close to ε, as a reply. It involves low run-time overhead, as a binary search can quickly find those candidate nodes.

V. Performance Evaluation and Comparison

Performance of GST under different network sizes is evaluated using *ns*-2 (Network Simulator-2) [10]. The simulation results are compared with those of CBT, in terms of three performance metrics: mean delay, number of on-tree nodes, and cost. The first metric reflects the average time taken to multicast a packet from its source to all members, and a more efficient delivery tree leads to a smaller mean delay for the same multicast group. The second metric registers how many nodes (routers) are involved in the shared tree built for a multicast group. Each on-tree node maintains states associated with the tree, and thus it is desirable to keep such nodes as few as possible for a given group. The last measure indicates the total cost incurred, on an average, when delivering one multicast packet to all members, given that each network link is randomly assigned with a cost ranging from 1 to 10 units [10]. A shortest path intends to traverse those links with lowest combined costs.

Simulation Scenario

Different network topologies for a range of sizes have been generated following the tiers network generation model [8] for evaluation. We used only the lowest tier to capture the domain topology, with a random distribution for node degrees [15] ranging from 1 to 6. For each network with size n, we assigned up to $n/5$ network nodes randomly as sources, each generating multicast data packets with size 1K bits at the constant rate of 1 packet per 9 *ms*. It is observed that the packet generation rate has little impact on performance metrics gathered. Multiple group joining sequences were produced for evaluating each network topology under a given network size, with their results averaged for demonstration.

Every multicast packet is time-stamped when generated by a source in order to measure its delay (in *ms*) to reach each group member. Such a delay measured at each member is accumulated to get the mean delay. Data values presented in subsequent figures are the average results over 4 different topologies each under 5 joining sequences, for a given n.

While a GIN need not be on the constructed multicast tree and multiple GINs may exist, in our simulation, the GIN is connected to the tree (acting like a core of the multicast tree in CBT). When a new joining node sends a request to GIN for guidance information (i.e., candidate on-tree nodes), the request also serves to establish a *temporary* connection to the tree, effectively lowering run-time overhead in the join process. This temporary connection permits the joining node to start receiving multicast packets in a latency identical to that of CBT. Once the joining node decides the nearest on-tree node to connect (according to candidate on-tree nodes provided by the GIN), a permanent connection is established between the joining node and the tree, while at the same time, the temporary connection is removed, finishing the joining process. Our simulation results presented below illustrated GSTs built with single GINs only and without rectification steps (outlined in Section III) involved.

Results and Comparison

To assess the impact of the set size (i.e., number of candidate on-tree nodes chosen) on quality of GST built, we varied the set size in our simulation for two network sizes (n = 50 and 150), with results depicted in Fig. 4, where the mean delay was measured after all group members have joined, and LI described earlier serves as proximity hints (PH). LI assignments after various re-computation iterations governed by Eq. (1), with $w_l = 0.6$ and $w_i = (1-w_l)/k$ for a node having k neighbors, were examined. For a given network size and a set size, the mean delay reduces as more iterations (denoted by #) are applied, where # = 0 is the initial LI assignment. When the number of iterations exceeds 3, however, delay reduction becomes negligible even for the network size of 150 (and in fact, the results for # ≥ 2 are all identical for n = 50 and are not shown explicitly in Fig. 4). Separately, the mean delay drops considerably as the set size increases initially under # = 0, and it exhibits little shrinkage after the set size goes beyond 6 for # ≥ 2. If the number of iterations equals 3 or more, the set size of 5 is adequate and can be chosen for use practically.

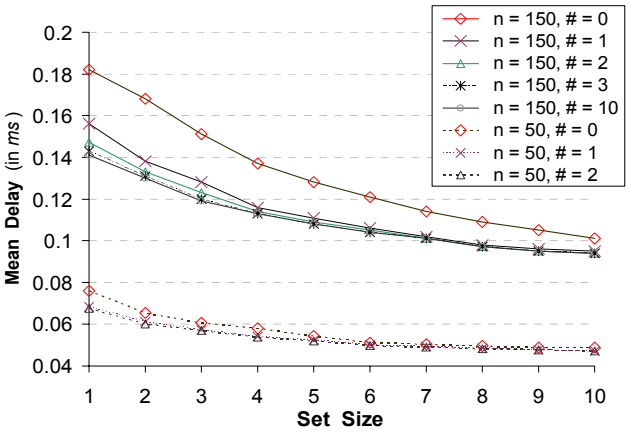

Fig. 4. Mean delay versus set size for n = 50 and 150.

We simulated a range of network sizes (n) and found that the results follow a similar trend for different n values. In Fig. 5, performance results for n = 150 are illustrated, where the set size is chosen to be 5 and the number of group members varies from 2 to 30. The LID bars refer to the results of our GSTs with LI after # iterations served as PH, whereas ODT (optimum distribution tree) bars denote those of "best" GSTs possible to be built by making the GIN reply with all on-tree nodes in response to each joining request. CBT clearly exhibits worse results than LID in terms of the

three performance metrics. When there are 20 group members, for example, CBT experiences a mean delay of 0.13 *ms*, in contrast to 0.10 *ms* of LID with # = 3, translating to a 23% reduction. As shown in Fig. 5(c), LID with # = 3 yields a more pronounced cost savings (of about 26%) when compared with CBT, for the case of 20 group members. Meanwhile, ODT leads to a slightly lower delay (but larger traffic overhead) than LID with # = 3, and the gaps of the other two performance metrics are negligible, as depicted in Fig. 5(b) and Fig. 5(c). This signifies that LID with # = 3 constructs trees close to what can be best possibly built, even for a small set size of only 5. The effects of PH enhancement through LI re-computation are noticeable for the mean delay measure, as shown in Fig. 5(a), when the LID results under # = 3 and those under # = 0 are contrasted.

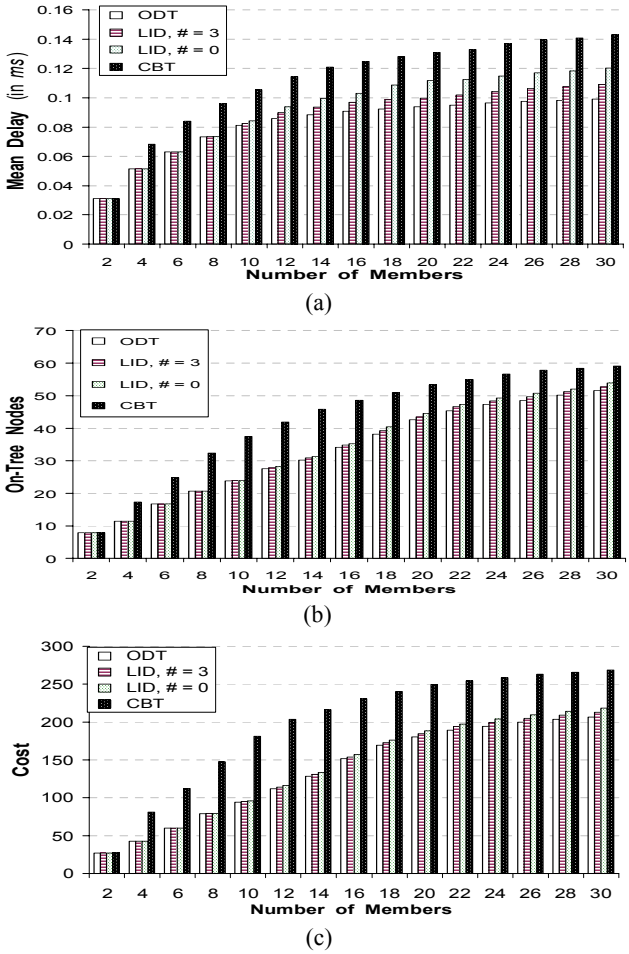

Fig. 5. Performance results vs. group members for n = 150.

VI. Conclusion

This paper has dealt with efficient multicast over shared trees developed under proper guidance when group members join. One node is selected to provide such guidance for each group, called a GIN (guidance information node). Each node to join the group finds its GIN using the group ID indexed into a list of GINs. The use and implementation of proximity hints (PH) are discussed. Simulation results have revealed that the proposed network-layer framework indeed constructs more effective multicast delivery trees (in terms of performance measures of interest examined) than its CBT counterpart. Re-computing LI (location indicators) for a few iterations (say, 3) is shown to enhance PH considerably, permitting a GIN to choose small sets of candidate on-tree nodes (of 5) in order to keep involved traffic overhead low.

References

[1] A. Ballardie, P. Francis, and J. Crowcroft, "Core-Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing," *Proc. ACM SIGCOMM'93*, Sept. 1993, pp. 85-95.

[2] A. Ballardie, "Core Based Trees (CBT) Multicast Routing Architecture," Internet RFC 2201, Sept. 1997.

[3] K. Calvert, E. Zegura, and M.Donahoo, "Core Selection Methods for Multicast Routing," *Proc. Int'l Conf. on Computer Communications Networks*, Sept. 1995.

[4] L. Costa *et al.*, "Hop By Hop Multicast Routing Protocol," *Proc. ACM SIGCOMM '01*, Aug. 2001, pp. 249-259.

[5] J. Crowcroft and K. Carlberg, "Application-Level Multicast Architectural Requirements for APEX," IETF Draft, draft-crowcroft-apex-multicast-00.txt, *Feb. 2001*.

[6] S. Deering, "Host Extensions for IP Multicasting," Internet RFC 1112, Aug. 1989.

[7] S. E. Deering *et al.*, "The PIM Architecture for Wide-Area Multicast Routing," *IEEE/ACM Trans. on Networking*, vol. 4, pp. 153-162, Apr. 1996.

[8] M. Doar, "A Better Model for Generating Test Networks," *Proc. IEEE Global Telecommunications Conf. (GLOBECOM*), Nov. 1996.

[9] D. E. Estrin *et al.*, "Protocol Independent Multicast – Sparse Mode (PIM-SM): Protocol Specification," Internet RFC 2362, June 1998.

[10] K. Fall and K. Varadhan, *The ns Manual*. UC Berkeley, LBL, USC/ISI, and Xerox PARC, Jan. 2001. Available at http://www.isi.edu/nsnam/ns/doc/index.html.

[11] M. Faloutsos, A. Banerjea, and R. Pankaj, "QoSMIC: Quality of Service Sensitive Multicast Internet Protocol," *Proc. ACM SIGCOMM '98*, Sept. 1998, pp. 144-153.

[12] C. Kommareddy, N. Shankar, and B. Bhattacharjee., "Finding Close Friends on the Internet," *Proc. 9th IEEE Int'l Conf. Network Protocols,* Nov. 2001.

[13] S. Kumar *et al.*, "The MASC/BGMP Architecture for Inter-Domain Multicast Routing," *Proc. ACM SIGCOMM '98*, Aug. 1998, pp. 93-104.

[14] J. Moy, "Multicast Extension to OSPF," Internet RFC 1584, Mar. 1994.

[15] H. Tangmunarunkit *et al.*, "Network Topology Generators: Degree-Based vs. Structural," *Proc. ACM SIGCOMM '02*, Aug. 2002.

[16] W. Waitzman, C. Partridge, and S. Deering, "Distance Vector Multicast Routing Protocol," Internet RFC 1075, Nov. 1988.

[17] L. Wei and D. Estrin, "The Trade-offs of Multicast Trees and Algorithms," *Proc. 1994 Int'l Conf. on Computer Communications Networks*, Sept. 1994.

[18] D. Zappala and A. Fabbri, "An Evaluation of Shared Multicast Trees with Multiple Active Cores," *Proc. Int'l Conf. on Networking* (*ICN'01*), July 2001.