Efficient Heuristic for Minimum Cost Trees Construction in Multi-Groups Multicast

Keen-Mun Yong, Gee-Swee Poo and Tee-Hiang Cheng

Network Technology Research Centre (NTRC)
School of Electrical and Electronic Engineering, Nanyang Technological University,
Singapore 639798.
{yong0010, egspoo, ethcheng}@ntu.edu.sg

Abstract. We propose an efficient and fast heuristic algorithm (MGST) to construct minimum cost tree for each multicast-group in multi-groups multicast. The idea is to construct just one spanning shared-tree that allows minimum cost tree of each multicast-group to be computed from this shared-tree. MGST is compared with the low overheads Core-Based Tree algorithm (CBT) and approximate optimal cost Minimum Cost Tree heuristic (MCT) through simulations. The results showed that MGST constructs trees with average cost close to the approximate optimal (7% difference). The overheads incurred from MGST are also comparable to CBT and much lesser than MCT. In addition, MGST has also shown to construct minimum cost trees much faster than MCT and comparable to CBT. To further improve the performance, "trace back" method is introduced to MGST and denoted as MGST-T. MGST-T yields average cost performance of only 2% difference from the optimal and with almost the same overheads and trees' construction time as MGST.

Keywords: Multicast, Multi-groups, Minimum cost tree, Shared-Tree.

1 Introduction

Due to increase demand of multicast services, it is not unusual to have applications that cater multicast services to a wide diversity of users. Each group of users will be assigned a multicast-group within which multicasting can take place and each request to deliver traffic within the group is known as a multicast-session. We refer this as multi-groups multicast. Each user in the group will be identified as the group's member. The tree constructed to deliver traffic for each group must be cost-efficient due to economic reason. Thus, the problem of computing minimum cost trees in multi-groups multicast is referred as the Multi-Groups Multicast problem (MGM).

The problem of constructing minimum cost tree for a multicast-session can be modeled as a Steiner tree problem which is known to be NP-complete [1], and heuristics have been developed to approximate the constructed tree cost to the optimal. The most well-known is the one from [2] which constructs minimum cost tree for each multicast-session from a graph consisting of minimum cost paths between any two nodes for all the nodes. We henceforth refer the heuristic from [2] as Minimum Cost Tree heuristic (MCT). As MCT or any other source-based algorithms constructs a tree for each session from source to its destinations, the overheads

incurred and the time to construct a tree will be large if the number of destinations is large. This is not desirable especially for applications which require the trees to be ready within a short time period. Also, effective algorithms for dynamic membership multicast [3] proposed in [4], [5] will not work well and may incur race condition if the rearranged tree is not ready before the arrival of next member's addition or removal event.

In this paper, we proposed Multi-Groups Shared Tree algorithm (MGST) for the MGM problem. The objective is to construct minimum cost trees comparable to the optimal cost using minimal overheads and time to provide reliable and efficient multicasting. We then further improve the cost performance of MGST and denote it as MGST-T. The organization for the rest of the paper is as follows. Section II states the network model. Section III gives a brief review of CBT[6] and MCT in multi-group multicast. Section IV presents the proposed algorithms. The performance of the proposed algorithms is evaluated in section V. Section VI concludes the paper.

2. Network Model

We model the network as a graph *G = (V, E)* where *V* is a set of nodes and *E* a set of edges. All nodes are assumed to be source and destination node candidate and multicast-capable. Each edge $e \in E$ connecting node x and y is denoted as $e_{x,y}$ and has a non-negative cost $C(e)$ associated with it. We assume symmetric link cost. The cost of any directed tree $T = (V_T, E_T)$ in the network is given by $C(T) = \sum_{e \in E_T} C(e)$. The set of multicast-groups is given by $Q = \{q_1, ..., q_x\}$ where each group consists of members given by $q_i = (m_i)$. The set of incoming multicast-session requests is given by $Y = \{y_1, ..., y_j\}$ and each request is expressed in the form of $y_i = \{X, s_i, D_i\}$ where s_i and D_i are the source and its set of destinations respectively, and X is the multicast-group ID which the session is requested from. In order to deliver traffic efficiently, a tree T_y is constructed for each session y.

3 Review of CBT and MCT in Multi-Groups Multicast

3.1 Core-Based Tree Algorithm (CBT)

In CBT [6], a core node is first selected and each member of a multicast-group sends a join message to this core node to connect itself to the shared-tree constructed so far. This continues until all the members are connected via the shared-tree. Hence, each shared-tree is shared among the members of each group and the number of CBT required is proportional to the number of multicast-groups involved. However, for each group, since only a single shared-tree is employed, the number of overheads

incurred and the processing time to construct the tree ready for data delivery each time a session request is made will be low. Thus, we evaluate the performance of our proposed algorithm using CBT as the benchmark in terms of the overheads incurred as well as the processing time to ready the tree for data delivery. However, due to all paths in tree are required to travel through a core node, the selection of the core node will determine the resource and cost efficiency of the tree. This means that low cost tree cannot be a guarantee since core node selection is complex and can only be approximated most of the time and congestion may occurs at the core node leading to undesirable consequence unless a very efficient queuing model can be applied at the core node.

In this paper, to minimize the tree cost constructed by CBT for comparison with our proposed algorithm in terms of both overheads and cost, the core node is selected such that its average minimum cost to the other members in the group is the minimum compared with the rest of the nodes. After which, each member of group is added to the core-based tree by means of least minimum cost path until all the members are connected in the shared-tree.

3.2 Minimum Cost Tree Heuristic (MCT)

In MCT [2], minimum cost path between any two nodes are first computed. These paths make up the links of a minimum cost graph. In this graph, all the nodes are connected via links which corresponds to their respective minimum cost path joining between them. For each incoming multicast-session request, the tree construction starts from the source node. From the source node, the least cost link in the graph to its neighboring member node is appended to the tree constructed so far. The next least cost link from unconnected member node to this tree will then be appended and this continues until all member nodes have been connected to the tree. The links in the obtained tree are then expanded to their respective minimum cost paths. Pruning may be required to remove any unnecessary links. It has been shown in [2] that MCT is able to obtain trees with cost very close to the optimal. Thus, we evaluate the performance of our proposed algorithm using MCT as the benchmark in terms of cost efficiency. However, since a minimum cost tree needs to be constructed for each incoming session request, and the least minimum cost path is required to be searched and selected to append to the tree for connection of each member, therefore the processing time to ready the tree for data traffic and overheads incurred will be significantly larger than that of CBT.

4 Our Proposed Algorithms (MGST and MGST-T)

4.1 Details

Our proposed Multi-Groups Shared Tree algorithm (MGST) consists of two phases. The first phase is to construct a cost efficient shared-tree spanning to all nodes. The

second phase is to compute minimum cost tree for each multicast-session from the shared-tree obtained in the first phase. We assumed that the minimum cost path between any two nodes in the graph has been computed using Dijkstra algorithm and stored in the set *W*. MGST-T is an improvement of MGST where "trace back" method is introduced in the second phase.

4.1.1 Phase 1: Construction of Cost-Efficient Spanning Shared-Tree

For each iteration of the cost-efficient shared-tree construction, we consider each neighboring node of the existing shared-tree as the potential node to connect. In order to have a cost-efficient shared-tree, the selection function (SF) which selects each neighboring node to add to the tree in each iteration must be in such a way that the cost to deliver traffic via (as a transition node) or from the node (as a source node) is low. A transition node is used to relay data. Hence, for each iteration of the shared-tree construction, the SF of a particular unconnected node *j* which is a neighbor to the existing shared-tree should consider the average minimum cost to deliver traffic to each node from node *j* either as a transition node or source node.

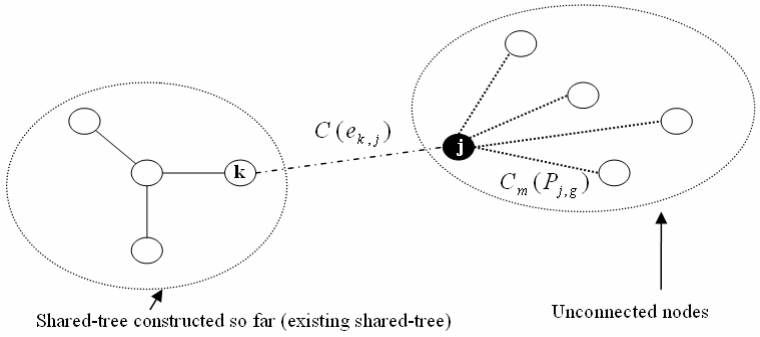


Fig. 1. An example of constructing shared-tree in MGST

From fig.1, when the next node is to be added to the sharing-tree, the SF of each potential node *j* needs to compute the average minimum cost to deliver traffic via node *j* and this consists of average minimum cost from any of the nodes in the existing shared-tree constructed so far to node *j* and the average minimum cost to any of the nodes not in the tree from node *j*, and vice versa. Since symmetric link cost is assumed, therefore only one direction needs to be considered. Since those unconnected nodes cannot be reached via the tree, therefore the computation of the average minimum cost to any of the nodes not in the tree from node *j* should be approximated by the average minimum path cost from *j* to the unconnected nodes.

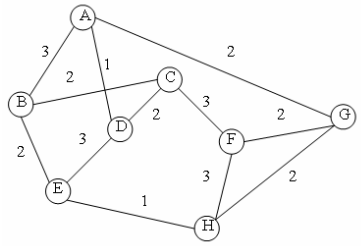
Let *H* and *R* be the set of unconnected nodes and set of connected nodes to the existing shared-tree respectively. Denoting $e_{k,j}$ as the minimum cost link between the existing shared-tree $\overline{T}_s$ and node *j* and $C_m(P_{x,y})$ as the minimum cost path between node *x* and *y*, the selection function of a unconnected node *j* can be expressed as,

$$SF(j) = \left[\frac{1}{|R|} \left[C(\overline{T}_s) + C(e_{k,j}) \right] + \frac{1}{|H|} \sum_{b \in H} C_m(P_{j,b}) \right] \quad \textbf{(1)}$$

The first term in (1) computes the average minimum cost from any of the nodes in the existing shared-tree constructed so far to node *j* and the second term computes the average minimum cost to any of the nodes not in the tree from node *j*. The node with the least SF value among the unconnected nodes is selected to append to the existing shared-tree via the minimum cost link between the existing shared-tree and the node, and this continues until all nodes are appended to the tree. For each iteration of the shared-tree construction, since the existing shared-tree makes no further changes in its structure, therefore it is independent to the cost contribution to future iteration of the construction process. This means that for the SF computation of an unconnected node during each iteration, the existing shared-tree can be viewed as a single entity, and the first term in (1) can be replaced by the average minimum cost between the existing shared-tree and the unconnected node. Thus, (1) is further simplified as,

$$SF(j) = \left[C(e_{k,j}) + \frac{1}{|H|} \sum_{b \in H} C_m(P_{j,b}) \right] \quad (2)$$

The final tree obtained is the cost-efficient spanning shared-tree. For selecting the first node to include in the shared-tree to start the construction process, it can be observed that since initially no shared-tree is constructed, therefore the first component of (2) is zero and the node with the least average minimum cost to all other nodes is selected to add into the shared-tree to start the construction process. Each node in the existing shared-tree is required to know the up-to-date configuration of the existing shared-tree so that the SF of those unconnected nodes can be computed for further tree construction. Hence, whenever a new node is added to the shared-tree, a control packet which stores the new node addition information is broadcast from the nearest on-tree neighboring node of the added node to all other nodes in the existing shared-tree. The overheads incurred by MGST in phase 1 are the total number of control packets required to complete the shared-tree construction.



A→D, 1 A→G, 2 A→B, 3 A→D→C, 3 A→D→E, 4 A→G→F, 4 A→G→H, 4	B→E, 2 B→C, 2 B→A, 3 B→E→H, 3 B→C→D, 4 B→C→F, 5 B→A→G, 5	C→B, 2 C→D, 2 C→F, 3 C→D→A, 3 C→B→E, 4 C→F→G, 5 C→F→H, 5	D→A, 1 D→C, 2 D→E, 3 D→A→G, 3 D→C→B, 4 D→E→H, 4 D→C→F, 5
E→H, 1 E→B, 2 E→D, 3 E→H→G, 3 E→B→C, 4 E→H→F, 4 E→D→A, 4	F→G, 2 F→H, 3 F→C, 3 F→G→A, 4 F→H→E, 4 F→C→B, 5 F→C→D, 5	G→F, 2 G→H, 2 G→A, 2 G→H→E, 3 G→A→D, 3 G→A→B, 5 G→F→C, 3	H→E, 1 H→G, 2 H→F, 3 H→E→B, 3 H→G→A, 4 H→E→D, 5 H→E→B→C, 5

Fig. 2. 8-nodes random network and its table of minimum cost path between any two nodes using Dijkstra algorithm. The numeric cost of each path is shown beside the path in the table.

Fig.2 shows an 8-nodes random network and the corresponding minimum cost path between any two nodes tabulated using Dijkstra algorithm. Fig.3 shows the stages of computing the spanning shared-tree using MGST from the network in fig.2. For the first stage shown in fig.3(a), node G has the least SF value of 2.86. Hence, it is selected as the first node to add to the shared-tree. Since only node F, A and H are neighbors to G in the network, therefore their SF are computed while the rest are labeled infinity. Since node A has the least SF, it is selected next to be connected to

the shared-tree shown in fig.3(b). If there is a tie in SF value, the one with the least connection link cost will be selected to be connected to the shared-tree. This is repeated shown in fig.3(c) to fig.3(f) until all the nodes are connected in the spanning shared-tree shown in fig.3(h).

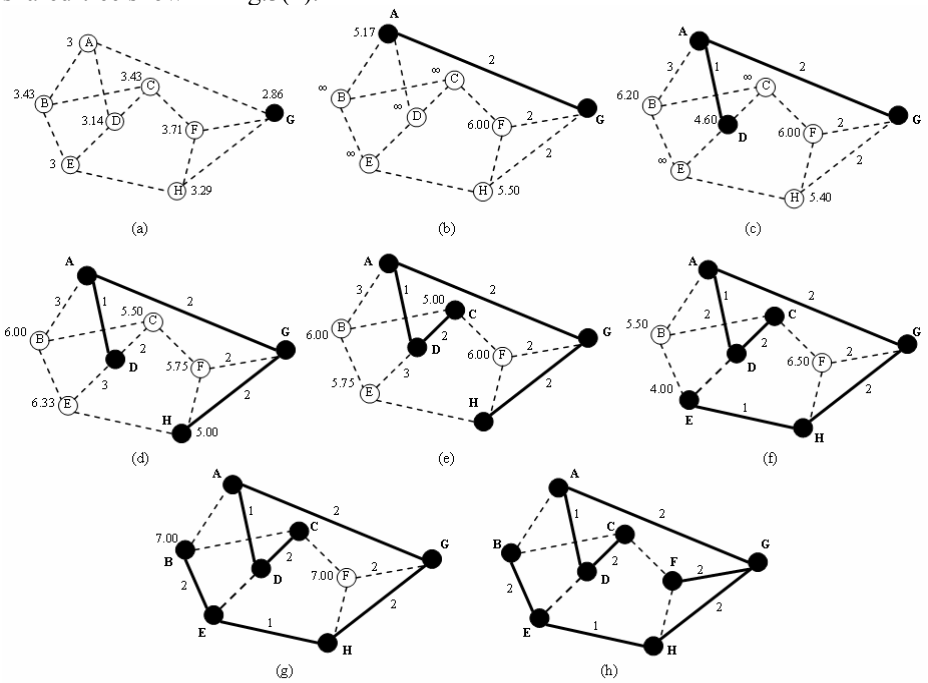


Fig. 3. (a) - (g) Seven stages in producing the cost-efficient spanning shared-tree using MGST. (h) The final cost-efficient spanning shared-tree produced by MGST.

4.1.2 Phase 2: Construction of Minimum Cost Tree for each Session

When an incoming session arrives, the source-members pair information will be stored in each data packet as part of the packet's overheads. Each packet will be sent from the source via the shared-tree constructed in section 4.1.1. Each node in the shared-tree upon receiving the packet will check if its descendent sub-tree has any member nodes. If there is, the packet will be duplicated and transmitted out, while no further action will be done if otherwise.

Fig.4 shows the tree constructed for a session with source node at C and destination nodes at E, G, F, A and B by MGST, MCT and CBT. The arrows indication in fig.4 shows CBT is a core-based tree with core at G while MGST and MCT are source-based trees. G is selected as the core for CBT because it has the least average minimum cost to the other destination nodes. It can be observed that both MGST and MCT have tree cost of 12 units while CBT has tree cost of 13 units. Although it is fairly simple to obtain minimum cost tree through pruning of the shared-tree in MGST, one disadvantage is that minimum cost tree may not be obtained for all cases. For example, if only a handful of nodes are members, then the

session-tree obtained via pruning from the spanning shared-tree may consist a large number of non-leaf nodes which are not necessary (redundant nodes) to connect the destination nodes. This leads to redundant cost in the tree and increases the tree cost instead of minimizing it.

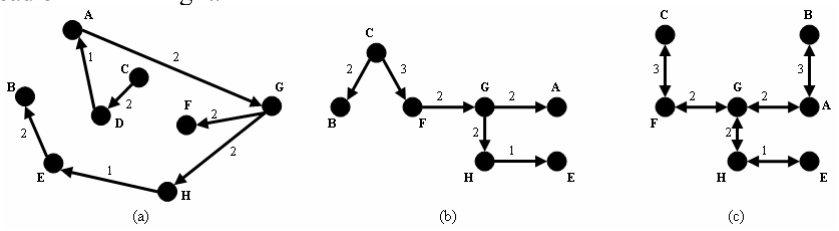


Fig. 4. An example of constructing minimum cost tree for a session with source node at C and destination nodes at node E, G, F, A, B in network given in fig.2. (a) Session-Tree produced by MGST. (b) Session-Tree produced by MCT. (c) Session-Tree produced by CBT.

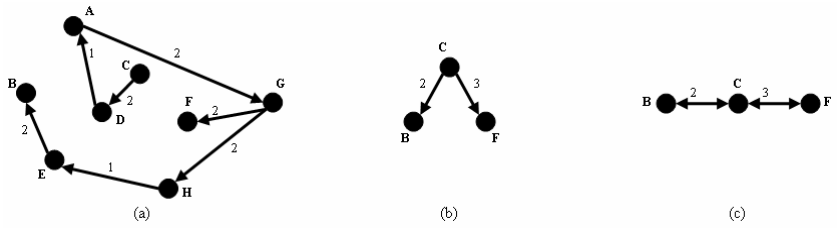


Fig. 5. An example of constructing minimum cost tree for a session with source node at C and destination nodes at node F, B in network given in fig.2. (a) Session-Tree produced by MGST. (b) Session-Tree produced by MCT. (c) Session-Tree produced by CBT.

Fig.5 shows one such example where a session has source node at C and destination nodes at B and F. As node B and F are leaf-nodes in the shared-tree obtained from MGST with respect to the source C shown in fig.5(a), therefore no edges can be pruned and the total session-tree cost is 12 units. This is in big contrast to the tree obtained from MCT and CBT both with session-tree cost of 5 units.

To solve this, we introduced "trace back" method to phase 2 of MGST to trace back the minimum cost link between each node of the shared-tree and the current active tree constructed. Each node in the shared-tree upon receiving the packet will check if a direct minimum cost link to the current active tree existed. If so, the minimum link cost is compared to the cost of shared-tree path between the nearest "anchor" node towards the direction of the source and the node. This "anchor" node is defined as the ingress node of the shared-tree path which can be replaced by the minimum cost link to the particular node without creating any dis-joint trees as a result. Hence, "anchor" node can be the source node; ingress of the minimum cost link; a destination node or a node with out-degree greater than one. The minimum cost link replaces the shared-tree path if it has smaller cost. Two extra control packets are required. One packet is required to prune the replaced path and the other packet to activate the replacing link. The idea is to remove redundant nodes in the tree.

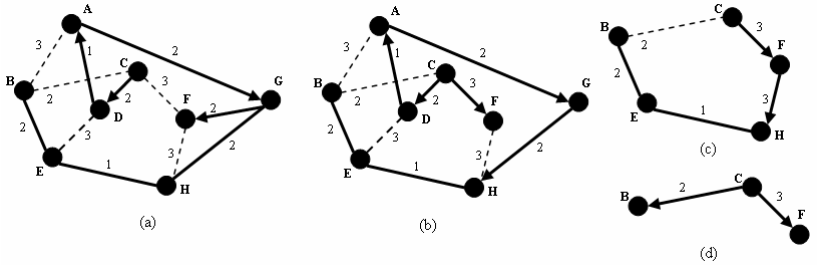


Fig. 6. (a) – (d) Four stages to construct minimum cost tree in MGST-T. (d) The final session-tree for session with source at node C to destination node B and F.

Fig.6 shows an example of the "trace back" method in MGST. From the shared-tree obtained in fig.3(h), since F is the first node from C which has minimum cost link connecting to the current active tree constructed, therefore the cost of path from C to F in the shared-tree and the cost of minimum cost link between C and F are compared. Since the minimum link from C to F has cost of only 3 units, while the path from C to F in the shared tree has 7 units, therefore the path is replaced with the minimum cost link. However, since G has another link connected to H beside F, therefore only the link between G and F in the path is replaced with the link between C and F shown in fig.6(b). The next node which has minimum cost link connecting to the current active tree constructed is H. As the minimum cost link between F and H is smaller than from C to D to H, the path C to D to H is replaced with C to F to H shown in fig.6(c). And finally at B, since C to B has smaller cost than from C to F to H to E to B, thus the final session-tree is obtained in fig.6(d). This tree has the same low cost of 5 units compared to MCT and CBT shown in fig.5(b) and (c). This is a vast improvement compared to the tree obtained without "trace back" in MGST shown in fig.5(a). For easier reference and distinction of the improvement made to our proposed algorithm, we denote MGST-T as the algorithm with "trace back" method introduced.

4.2 Complexity of MGST and MGST-T

The worst case complexity to construct the shared-tree is $O(|V|^2)$ as each node is considered iteratively to append to the shared-tree. As no further action is required other than identifying pruned edges and tracing minimum cost link when session request arrives, by normalizing the complexity of all identification processes whereby tree information is known as $O(1)$, the complexity to set up session-tree is $O(1)$. Thus, the total complexity of MGST is $O(|V|^2)+O(|Y|)$ where Y is the total set of multicast-session requests. If the worst case complexity is such that each node performs "trace back" in MGST-T, then the total worst case complexity of MGST-T will be $O(|V|^2)+O(|2Y|)$ since again all the "trace back" links and the replaced paths are all known and do not require additional searching. The worst case complexity for CBT and MCT are $\sum_{q_i \in Q} O(|q_i|^2)+O(|Y|)$ and $\sum_{y_i \in Y} O(|D_i|^2)$ respectively.

5. Performance Evaluations

5.1 Performance Metrics

5.1.1 Cost Competitiveness (CC)

It is a measure of a given algorithm's average tree cost performance with respect to an optimal algorithm. Finding optimal cost tree is proven to be NP-complete. As MCT constructs trees with cost approximate to the optimal, therefore, it is used as the benchmark for cost efficiency. Let $R = \{T_1, ..., T_x\}$ and $U = \{B_1, ..., B_x\}$ be the set of trees constructed by a given particular algorithm and by MCT respectively for a total of $x = |Y|$ multicast-session requests. The CC of a given algorithm is expressed as $CC = \frac{1}{x}\sum_{y \in Y}\left(\frac{C(T_y)}{C(B_y)}\right)$. A low CC is desired as it means that the given algorithm is able to construct trees with cost not too far off from the optimal cost algorithm.

5.1.2 Overheads Incurred

We defined overheads incurred as the total number of control packets required to set up the tree ready for multicasting. For CBT, the overheads incurred are the total number of join messages required to construct the trees for all groups. For MCT, the overheads incurred are the total number of control packets sent to each node in the existing tree of each iteration for all iterations of the construction process. For MGST, the overheads consist of the control packets required in constructing the spanning shared-tree. Additional overheads are incurred for MGST-T as mentioned earlier. The overheads incurred should be low to allow lesser resource used for setting up trees.

5.1.3 Tree Construction Processing Time

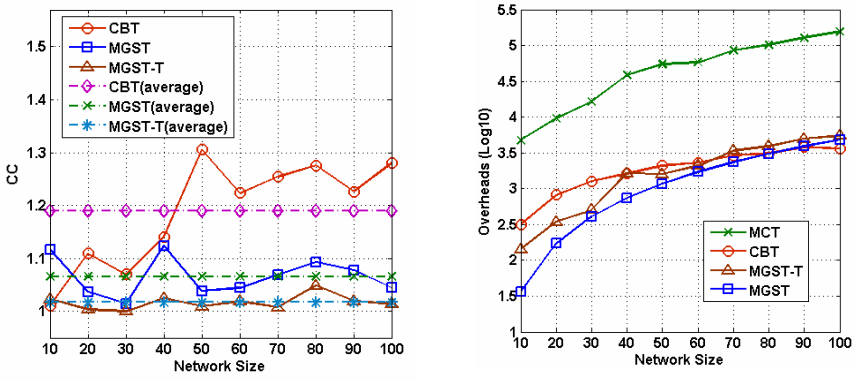
To measure the complexity of the algorithms we measure the average processing time required to construct a tree for each multicast-session. This does not include the spanning shared-tree construction time at the initialization stage for MGST since we are only interested in the time efficiency between the period when a session request is made and its corresponding tree ready for data transmission.

5.2 Simulation Model

Waxman method [7] was used to generate random networks ranging from 10 to 100 nodes with average node's outdegree of 3 to 4 to analyze the performance of MGST and MGST-T. Readers are to refer to [7] for details in the method. Each link has cost ranging between 1 to 10 units. 100 multicast-session requests are randomly generated. Each of the requests is to perform multicasting in an arbitrary selected multicast-group involved. The member size of each group is defined uniformly distributed when

all groups have the same member size, and non-uniformly distributed when the groups have different member size. For non-uniform distribution, we defined a member size range K and uses 50% of the network size as the mid-point of the range. This means that if the range is K on a network of 100 nodes, the member size distribution ranges from $50-K$ to $50+K$.

5.3 Simulation Results and Discussions



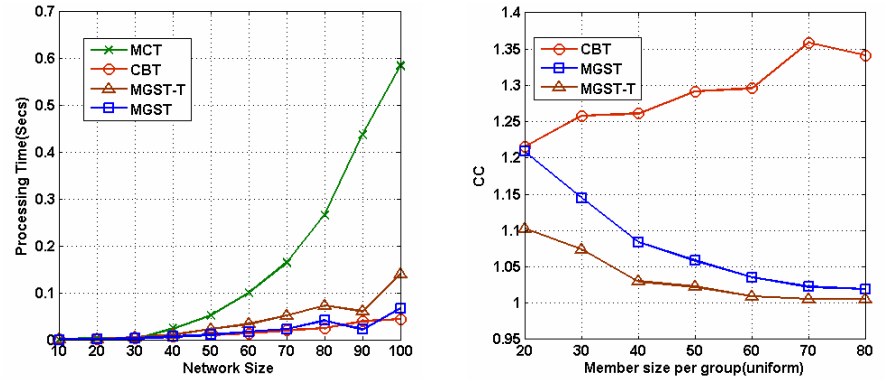
(a) (b)

Fig. 7. (a) CC vs. Network size. (b) Overheads Incurred in Log_{10} scale vs. Network size. (group size of 100 and uniform per group's member size of 50% of network size).

Fig.7(a) shows that CBT has an average CC of 1.2 while MGST and MGST-T have average CC of 1.07 and 1.02 respectively. This means that MGST and MGST-T are able to construct trees with cost very close to the optimal and performs much better than CBT. MGST-T has even lower CC than MGST due to the "trace back" method introduced to remove any redundant nodes in the trees to minimize costing. Fig.7(b) shows that the overheads incurred by MGST and MGST-T are smaller than CBT for smaller networks and comparable to CBT for larger networks. This is because the spanning shared-tree gets larger with larger network and thus requires more overheads. It is interesting to note that MGST and MGST-T have much lower overheads incurred compared to MCT (optimal cost heuristic) but yet able to yield also similar cost performance as shown in fig.7(a).

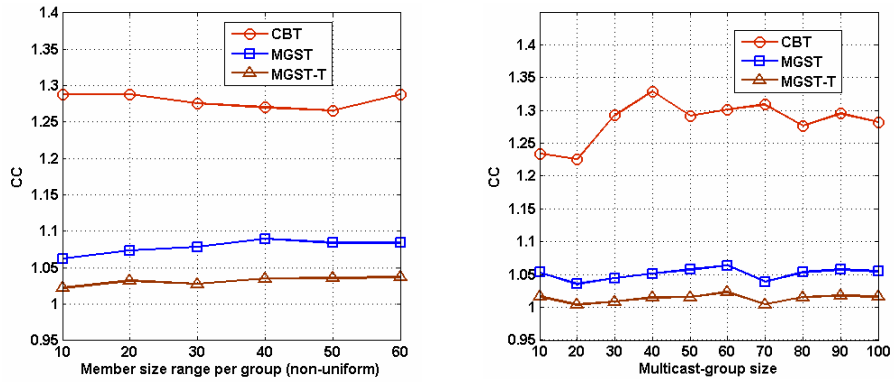
Fig.8(a) shows that the average time taken for MCT to ready a session-tree increases exponentially with increases in network size, while the increase is much more gentle for MGST, MGST-T and CBT. This is due to the employment of shared-trees which allows sessions to use it to derive the delivery trees without having to reconstruct them from scratch each time. From fig.7(a) , fig.7(b) and fig8.(a), it can be concluded that MGST and MGST-T are able to construct minimum cost trees using minimal overheads and within a very much shorter time. Fig.8(b) shows that as the member size per group increases, CBT increases in CC while MGST and MGST-T decreases in CC. This is because MGST and MGST-T construct trees out of a shared-tree which uses the average minimum cost from a node to the rest as the selection criteria to add each node iteratively. This means that we assume all other nodes are

members during the shared-tree construction. Thus, as the member size increases, more member nodes contribute to the average minimum cost computation from one node to all others, thus increasing the accuracy of the selection to construct a cost efficient shared-tree. Even at smaller member size, CC of MGST is comparable to CBT, and as the member size increases, MGST and MGST-T performs much better by having much smaller CC than CBT. It can be observed in fig.8(b) that MGST-T has even lower CC than MGST. This shows that our theory of removing redundant nodes via "trace back" method is correct.



(a) (b)

Fig. 8. (a) Average tree construction processing time vs. Network size (group size of 100 and uniform per group's member size of 50% of network size). (b) CC vs. Uniform member size per group (group size of 100 on 100 nodes network).



(a) (b)

Fig. 9. (a) CC vs. Non-uniform member size range per group (group size of 100 on 100 nodes network). (b) CC vs. multicast-group size (uniform per group member size of 100 on 100 nodes network).

Fig.9 shows that the CC performance of MGST and MGST-T are still much better than CBT and are close to the optimal when the member size per group is non-

uniform.. Fig.9(b) also shows that MGST and MGST-T perform much better than CBT when multicast-group size varies from 10 to 100 nodes. MGST-T lies very close to CC=1, indicating MGST-T's consistency in constructing trees with cost close to the optimal. Again, it can be observed that MGST-T performs better than MGST due to the removal of redundant nodes in the delivery trees. From the above figures, it is easy to conclude that MGST and MGST-T are able to construct minimum cost trees within a very short time and with overheads comparable to the optimal.

6. Conclusion

In this paper, we proposed efficient algorithms to construct a single spanning shared-tree used to compute each incoming multicast session tree. We explained the need for an algorithm that constructs cost-efficient multicast-trees within a very short processing time with as little overheads as possible. To lower tree cost even further, "trace back" is introduced to phase 2 of MGST and we denote this improved algorithm as MGST-T. Intensive simulations carried out on MGST and MGST-T against approximate optimal cost heuristic such as MCT and low overheads algorithm such as CBT, showed that MGST and MGST-T are able to compute trees within a very short processing time; with cost comparable to optimal cost MCT and overheads incurred comparable to low overheads CBT. In conclusion, our proposed MGST and its improved MGST-T are able to construct trees with approximate optimal cost in multi-groups multicast much faster than existing algorithms using much lesser overheads. However, it should be noted that MGST and MGST-T are only limited to non-delay constraint scenarios where delays requirement is largely relaxed. With more multicast applications having strict delays requirement, it will be worthwhile to expand MGST and MGST-T to delay-constraint environment as our future work.

References

1. R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, Complexity of Computer Computations, pp.85-103. Plenum Press, New York, 1972.
2. L. Kou, G. Markowsky, and L.Berman, "A fast algorithm for Steiner Trees, "Acta Informatica", vol.15.pp.141-145, 1981.
3. Aaron Striegel and G. Manimaran, "A survey of QoS Multicasting Issues", IEEE Comm. Magazine, pp.82-85 Jun. 2002.
4. Siram Raghavan, G. Manimaran and C.Siva Ram Murthy, "A Rearrangeable Algorithm for the Construction of Delay-Constrained Dynamic Multicast Trees", IEEE/ACM Transactions on Networking,vol.7, No4., pp. 514-529, August 1999.
5. F. Bauer and A. Verma, "ARIES: A rearrangeable inexpensive edge-based on-line Steiner algorithm", IEEE J. of Selected Areas in Comm., vol 15, pp. 382-397, April 1997.
6. A. Ballardie, P.Francis, and J.Crowcroft, "Core Based Trees (CBT) and Architecture for Scalable Inter-domain Multicast Routing", ACM SIGCOMM'93, pp. 85-89, Aug. 1993.
7. B. Waxman, "Routing of multipoint connections," IEEE Journal of Selected Areas in Communication, vol.6, pp.1617-1622, Dec. 1988.