

A Case for Tree Migration and Integrated Tree Maintenance in QoS Multicasting

Anirban Chakrabarti and G. Manimaran
Dependable Computing & Networking Laboratory
Dept. of Electrical and Computer Engineering
Iowa State University, Ames, IA 50011.
{*anirban,gmani*}@iastate.edu

Abstract

The proliferation of QoS-aware group applications coupled with the limited availability of network resources demands for efficient mechanisms to support QoS multicasting. During a life-cycle of a multicast session, three important events can occur: membership dynamics, network dynamics, and traffic dynamics. The first two are concerned with maintaining a good quality (cost) multicast tree taking into account dynamic join/leave of members, and changes in network topology due to link/node failures/additions, respectively. The third aspect is concerned with flow, congestion, and error control. There has been many solutions proposed for dealing with each of these issues. However, the issue of *tree migration* has not been addressed as part of these solutions. In this paper, we highlight the importance of tree migration as a mechanism for handling membership and network dynamics in core-based multicasting, prove that it is NP-Complete, and propose four heuristic algorithms for it. The proposed algorithms are evaluated under two performance metrics: service disruption and resource wastage. Our simulation studies show that two of the algorithms offer comparable performance to that of the other two, in addition to being highly scalable and easily implementable. Moreover, we also propose an integrated approach for group management involving both local and global tree maintenance techniques.

1 Introduction

The proliferation of QoS-aware group applications associated with recent advancements in high-speed networking is driving the need for efficient communication services satisfying the QoS requirements of such applications ([1, 2, 3, 4, 5]). Multicasting is an effective mechanism for supporting group communication. In a multicast communication, each sender transmits only one copy of each message that is replicated within the network and delivered to multiple receivers. For this reason, multicasting typically requires less total bandwidth than separately unicasting messages to each receiver. Based on the nature of membership, multicast groups are classified into two categories: *static groups* wherein the membership remains unchanged throughout the group's lifetime and *dynamic groups* wherein members join/leave the group dynamically.

It has been established that determining an optimal multicast tree for a static multicast group

can be modeled as the NP-complete Steiner tree problem ¹ in networks [2]. Consequently, a number of good, inexpensive heuristics for the Steiner problem have been proposed and reviewed extensively [5]. There have also been algorithms/protocols proposed to create multicast trees without resorting to the Steiner-tree based model. These protocols can be classified into two main approaches: source-based trees and core-based trees. Source-based tree approach uses a shortest path tree rooted at the sender/source. The branches of the tree are the shortest paths from the sender to each member/receiver of the group. When this approach is employed for many-to-many multicasting, a separate multicast tree must be constructed for each source. The Distance Vector Multicast Routing (DVMRP) [6], Multicast Extensions to Open Shortest Path First (MOSPF) [7], Protocol Independent Multicast-Dense Mode (PIM-DM) [8], and Explicitly Requested Single-Source Multicast (EXPRESS) [9] fall in the category of source-based trees.

The other type of protocol, called *core-based tree*, constructs a multicast tree spanning the members rooted at a special node, called center node or core node. Since all the senders share the same multicast tree for data transmission, these protocols are highly scalable. Core Based Tree (CBT) [10] and Simple Multicast (SM) [11] fall in the category of core-based trees. Hybrid routing protocols like PIM-Sparse Mode (PIM-SM) [8] and the Multicast Internet Protocol (MIP) [12] have been proposed that allow a receiver to switch from the shared tree to a shortest path tree.

An additional dimension to the multicast routing problem is the need to construct multicast trees that will satisfy the QoS requirements of modern networked multimedia applications. One of the most common QoS constraints is an upper bound on the delay between the transmission and receipt of a message. The problem of constructing delay-constrained low-cost multicast trees for routing to static and dynamic multicast groups has been addressed by many researchers (references in [2]). We identify the various issues associated with QoS multicasting through a “life-cycle” model.

1.1 Life-cycle of a QoS Multicast Session

A network architecture that aims to provide complete support for multicast communication is burdened with the task of managing the multicast sessions in a manner transparent to the users. This goal of transparent multicast service imposes specific requirements on the network implementation. To understand the different functionalities that such a network must provide, we show in Figure 1, the various steps and events that can take place in the “*life-cycle*” of a typical QoS multicast session [13]. The sequence of phases/steps relevant to the multicast session are (i) multicast group/session creation, (ii) multicast tree construction with resource reservation, (iii) data transmission, and (iv) multicast session tear-down. The first step deals with assigning a unique address to the multicast

¹Given a graph $G = (V, E)$, a cost function $C : E \rightarrow R^+$, and a set of nodes $D \subseteq V$, find a subgraph $H = (V_H, E_H)$ of G such that $D \subseteq V_H$ and the cost $C(H)$ (equal to the sum of the costs of the edges of E_H) is minimized.

group so that data of one group does not clash with the other. The second step is the construction of a multicast distribution tree spanning the source(s) and all the receivers, and reserving resources on the tree links. The third step is data transmission. The fourth and final step is the tear-down of the multicast session once its useful lifetime has elapsed, which includes releasing of session-specific resources such as bandwidth and group address.

The various events that can occur during this phase and the corresponding network mechanisms to handle these events are as follows (identified in Figure 1):

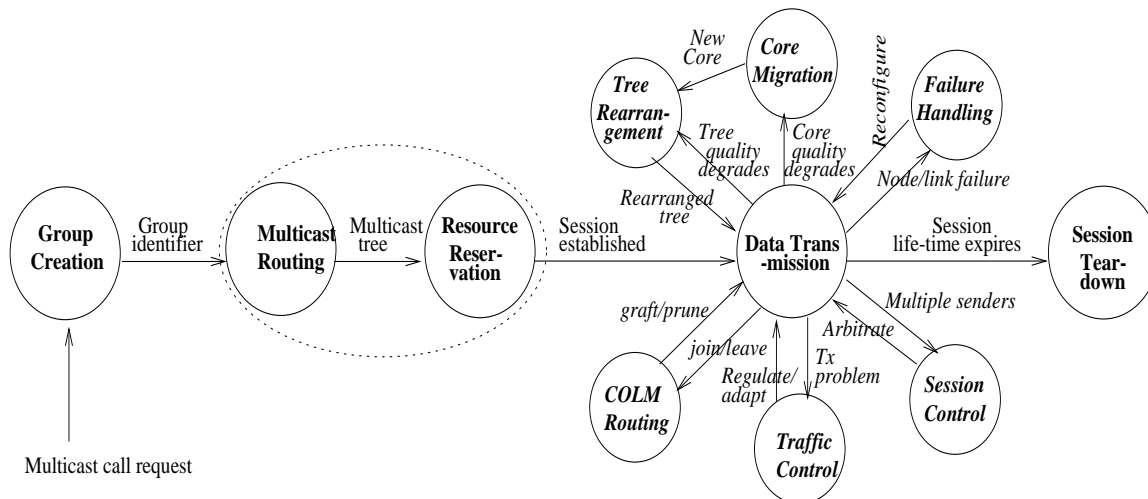


Figure 1: Life-cycle of a multicast session - an event diagram view

Group Dynamics: Since members can join/leave the group dynamically, the network must be able to track current membership during a session's lifetime. Tracking is needed both to start forwarding data to new group members and for stopping the wasteful transmission of packets to members that have left the group. A multi-stage mechanism is required to handle group dynamics, which are identified in Figure 1 as COLM (Constrained Online Multicast) routing, Tree Rearrangement, and Core Migration.

Transmission Problems: This could include events such as swamped receivers (needing flow control), overloaded routers (needing congestion control), or faulty packet transmissions (needing error control). The traffic control mechanism, working in conjunction with the schedulers at the receivers and the routers, is responsible for performing the necessary control to overcome these transmission problems (identified as Traffic Control in Figure 1).

Competition among Senders: In a core-based multicasting, multiple senders compete for the same session resources. This will result in data loss due to buffer overflow, thus triggering transmission problems. This requires a session control mechanism that arbitrates transmission among the senders.

There has been significant research work in dealing with transmission problems and session

control [14]. Though these two issues need further research, we do not address them here due to their broader scope. In this paper, we focus only on the issues triggered due to group dynamics, namely, multicast tree maintenance (COLM routing, tree rearrangement, core migration). These issues are highlighted in Figure 1.

2 Issues in Core-based Multicasting

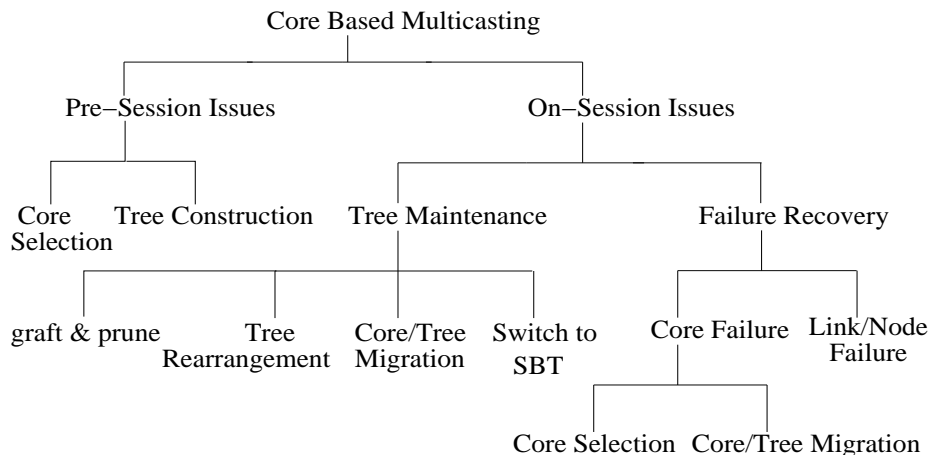


Figure 2: Issues in Core Based Multicasting

The main issues associated with core-based multicasting are shown in Figure 2. Among these, the unique issues are discussed below.

2.1 Pre-Session issues

Pre-session issues are typically issues that are tackled at the start of a multicast session. Since, these issues are handled off-line, centralized algorithms may be used to solve the problems arising out of the pre-session. Pre-session issues include core selection and initial multicast tree construction.

2.1.1 Core Selection

In core-based multicasting, since the tree is rooted at the core node, core selection is an important problem because location of the core influences the tree structure (and cost) which in turn determines the delay experienced by individual receivers. The optimal core selection is an NP-complete problem and usually requires complete network topology and exact group membership details. A number of heuristics have been proposed in [15, 16] for core selection and are evaluated in [17]. In all the core selection algorithms, a certain number of nodes called the *potential cores* express their willingness [16] to become the core of the tree. Each of them evaluate certain *weight*, which

is generally a function of delay, bandwidth, or both and ‘bid’ to become the ‘core’ node. The node having the best *weight*, is chosen as the core node.

2.1.2 Tree Construction

After core selection, multicast tree is constructed rooted at the core. As mentioned earlier, tree construction problem can be modeled as a NP-Complete and there are several centralized [18, 19] as well as distributed algorithms which produce multicast trees within a certain cost-bound (references in [2]).

2.2 On Session Issues - Failure Recovery

When the session is in progress, several problems need to be tackled. One such problem is recovery from failure. Two main types of failure can occur *viz.* Core failure and node/link failure.

- Core Failure Recovery: In core-based multicasting, core is a single point of failure. If the core fails, there is a large amount of *service disruption* as many of the receivers cannot receive the data sent by the senders. This area needs significant research attention. To the best of our knowledge, the work reported in [20] is the only work done in this area.
- Link and/or Node Failure: A link and/or node supporting the multicast session can fail which will result in service disruption. This requires mechanisms to detect link/node failure and to reconfigure (restore) the multicast tree around the faulty link/node with minimal service disruption. Note that multicast routing protocols based on underlying unicast routing are as survivable as the unicast routing protocol. If the multicast routing protocol is independent of the unicast routing protocol, it must implement its own restoration mechanism [3]. Though significant research has been done for failure recovery in unicasting [21, 22, 23], not much research has been done to handle failure in multicasting. [24] is the only work done in this area.

2.3 On-Session Issues - Tree Maintenance

In dynamic multicasting, due to continuous grafting and pruning, the multicast tree degrades over time [25]. To improve the quality (cost) of the multicast tree necessary steps have to be carried out. The steps required to improve the quality of a multicast tree are collectively referred to as *tree maintenance*. Though most of the issues discussed in this section are important and require significant research attention, in this paper, we focus our attention to the problem of tree maintenance.

Tree maintenance, as mentioned in the Figure 2, can be invoked in a local as well as global level. In case of local tree maintenance, a subset of all nodes which form the multicast tree is

involved in the maintenance process. Local tree maintenance optimizes a portion of the multicast tree, and may not be effective in optimizing the overall tree cost. In that case, the multicast tree has to be restructured completely. This is taken care of in global tree maintenance, where the current multicast tree is completely revamped and the members are moved from the old tree to the new tree in a scalable manner.

2.3.1 Local Tree Maintenance

Local tree maintenance involves *graft/prune* of the members (members joining/leaving the multicast group) in a QoS-aware manner and *rearrangement* of the portions of the multicast tree which are most effected by the members joining/leaving the group. Both these techniques effect a portion of the multicast tree and hence are categorized as local tree maintenance. Both categories of local tree maintenance have been extensively studied by researchers over the years ([26, 27, 28]). The two categories are discussed below:

- Members Join/Leave: When a node wishes to join/leave a multicast group, it sends a graft/prune message towards the core of the multicast group. The message traverses till it reaches an on-tree node of the multicast tree. During join, the graft message tries to form a least cost path from the core to the joining node and also tries to satisfy the QoS requirements of the node. This is a classical Delay Constrained Least Cost Multicast Routing Problem. This problem has been shown to be NP-Complete [2] and several heuristic algorithms (both centralized and distributed) are available which give an approximate solution to the problem. This problem has received significant attention from the research world in recent years. Protocols like the QoSMIC [29], QMRP [30] and parallel probing [31] attempt to optimize the tree and satisfy the QoS requirements of the members during join/leave. However, it is to be noted that in spite of optimization at the time of join/leave the cost of the multicast tree can deteriorate. This deterioration can be because of the skewed distribution of nodes joining and leaving the multicast group. Failure of some node/links which are part of the multicast group may also result in tree deterioration.
- Tree Rearrangement: In a dynamic multicast session, it is important to ensure that member join/leave will not disrupt the ongoing multicast session, and the multicast tree after member join/leave will still remain near-optimal and satisfy the QoS requirements of all on-tree receivers [2]. One way to handle dynamic member join/leave is by reconstructing the tree every time a member joins or leaves the session. This involves migration of on-tree nodes to the new tree, which may result in a large service disruption that QoS multicast sessions may not tolerate. Another way to handle dynamic member join/leave is by incrementally changing the multicast tree by graft/prune mechanism [2]. This incremental change approach suffers

because the quality (e.g. tree cost) of the tree maintained may deteriorate over time. This technique of modifying the tree incrementally is identified as tree rearrangement. Some of the common tree rearrangement algorithms are GREEDY [26], ARIES [27] and the CRCDM [28]. All these algorithms work by monitoring the accumulated damage to the multicast tree within local regions of the tree, as members join/leave the multicast group.

2.3.2 Global Tree Maintenance - Tree Migration

If the multicast group is highly dynamic, the core of the multicast group “degenerates” [25] over time and has to be replaced by a new core. This process is called core migration. During core migration, the structure of the multicast tree is totally revamped, and thus this process falls under the category of global tree migration. Though, over the years researchers have admitted the importance of core migration ([3], [25]), the actual process of moving the members from one tree to another has never been explicitly dealt with. The only works, to the best of our knowledge, in the area of global tree maintenance is [32, 33]. [32] is the shorter version of this paper. In [33], Carlberg described a very simple core migration strategy where new core and the old core reside in the same tree. This strategy does not result in tree migration. However, it is to be noted that such a simple strategy is not always viable because new core need not always be the part of the existing tree if the group is highly dynamic. Also, core migration may be initiated because of fault in the existing tree. In that case, new core has to be shifted outside the existing tree and then tree migration becomes inevitable. Therefore, tree migration is a general strategy to optimize the cost of the tree than the simple strategy described in [33].

In this paper, we first state the tree migration problem and prove that it is NP-Complete and then propose four heuristic algorithms for the problem. The heuristic algorithms are scalable and aim at minimizing either or both of the following performance metrics.

2.4 Tree Migration Problem and Performance Metrics

In order to precisely state the problem of tree migration we define two performance metrics - service disruption and resource wastage.

- Service Disruption (SD) is defined as the amount of packet loss experienced by the receivers during the process of tree migration.
- Resource Wastage (RW) is defined as the amount of excess resource (bandwidth) that remain idle multiplied by the duration for which it is idle during the process of tree migration.

$$RW = \sum_{i=0}^{R-1} (t'_{join_i} - t_{leave_i}) \times BW(path(C_o, r_i)),$$

where R is the number of receivers, t'_{join_i} is the time at which receiver r_i joins the new tree, t_{leave_i} is the time at which r_i leaves the old tree, and $BW(path(C_o, r_i))$ refers to the sum of bandwidth used for the multicast session along the path from old core to r_i in the old tree.

Tree Migration Problem: The problem is to migrate the receivers from one multicast tree to another with the goal of minimizing Service Disruption and Resource Wastage.

Lemma: Tree Migration problem is NP-Complete.

Proof: It is well known that the optimization problems that have two *path constraints* are NP-Complete [2]. For example, delay-constrained least-cost routing problem is NP-Complete as delay and cost are path (additive) metrics. The Tree Migration Problem has two path constraints: (i) Service Disruption, which is cumulative of packet loss experienced by each receiver; (ii) Resource Wastage, which is cumulative extra bandwidth temporarily reserved along the path from the source to receivers. Therefore, Tree Migration Problem is an optimization problem with two path constraints, and hence is NP-Complete.

2.5 Heuristic Algorithms for Tree Migration

We propose the following four heuristic algorithms for tree migration.

- *Add First Delete Last (AFDL):* In this algorithm, the new tree is created first (i.e., all the receivers join the new tree) and then the old tree is deleted (i.e., all the receivers leave the old tree). This algorithm offers less service disruption, but incurs a high resource wastage.
- *Delete First Add Last (DFAL):* This algorithm is the complement of the AFDL algorithm wherein the old tree is deleted first and then the new tree is created. This algorithm incurs no resource wastage, but incurs a high service disruption.
- *Interleaved Add Delete (IAD):* This algorithm adopts AFDL on a receiver basis. Each receiver first joins the new tree and then immediately leaves the old tree without waiting for the whole new tree getting created.
- *Interleaved Delete Add (IDA):* This algorithm adopts DFAL on a receiver basis. Each receiver first leaves the old tree and then immediately joins the new tree without waiting for the whole old tree getting deleted.

The advantage of AFDL is that it incurs a low service disruption (packet loss) as the old tree is deleted after constructing the new tree. The disadvantage of AFDL is that it incurs a high resource wastage as both the trees consume resources simultaneously for some duration of time. It is to be noted that the service disruption incurred by AFDL does not necessarily be zero because a packet

sent along the old tree may find that the resources on a link released and hence the packet may be dropped. The performance of DFAL is exactly the opposite of AFDL.

The difference between IAD and IDA bears resemblance to the difference between the AFDL and the DFAL algorithms, on a per-receiver basis. Thus, the service disruption of the IAD algorithm is less than that of the IDA algorithm, and the resource wastage incurred by IAD algorithm is more than that of the IDA algorithm. The performance of both the interleaved algorithms are bounded by the AFDL and the DFAL algorithms.

2.6 Tree Migration Protocols

The protocols that implement the proposed tree migration algorithms are shown in Figures 3(a)-(d).

IAD Protocol: Figure 3(a) shows this protocol. Here, the old core first sends the “migrate” message to all the receivers. Upon receiving this message, each receiver joins the new tree by sending a “join” message to the new core. The new core responds to the “join” message by sending “reserve” message to the receiver indicating reserving of resources along the path from new core to the receiver. Once the “reserve” message reaches the receiver, the receiver sends “leave” message to the old core. The old core responds to this message by sending a “release” message to the receiver indicating the release of resources along the path from old core to the receiver in the old tree.

IDA Protocol: Figure 3(b) shows this protocol. Here, the old core first sends the “migrate” message to all the receivers. Upon receiving this message, each receiver leaves the old tree by sending a “leave” message to the old core. The old core responds to the “leave” message by sending “release” message to the receiver indicating releasing of resources along the path from old core to the receiver in the old tree. Once the “release” message reaches the receiver, the receiver sends “join” message to the new core. The new core responds to this message by sending a “reserve” message to the receiver indicating reserving of resources along the path from new core to the receiver.

AFDL Protocol: Figure 3(c) shows this protocol. This is similar to IAD protocol except that the receivers expect a “tree_created” message from the new core indicating that the new tree has been created spanning all the receivers. This message is sent by the new core only when all the receivers have joined (attached to) the new tree. This means that the new core should know the membership of the multicast session.

DFAL Protocol: Figure 3(d) shows this protocol. This is similar to IDA protocol except that the receivers expect a “tree_deleted” message from the old core indicating that the old tree has been deleted. This message is sent by the old core only when all the receivers have left (detached from) the old tree. This means that the old core should know the membership of the multicast session.

Among the four protocols, the interleaved protocols (IAD and IDA) are highly scalable (with group size and network size) and easily implementable as they do not require the core nodes (old

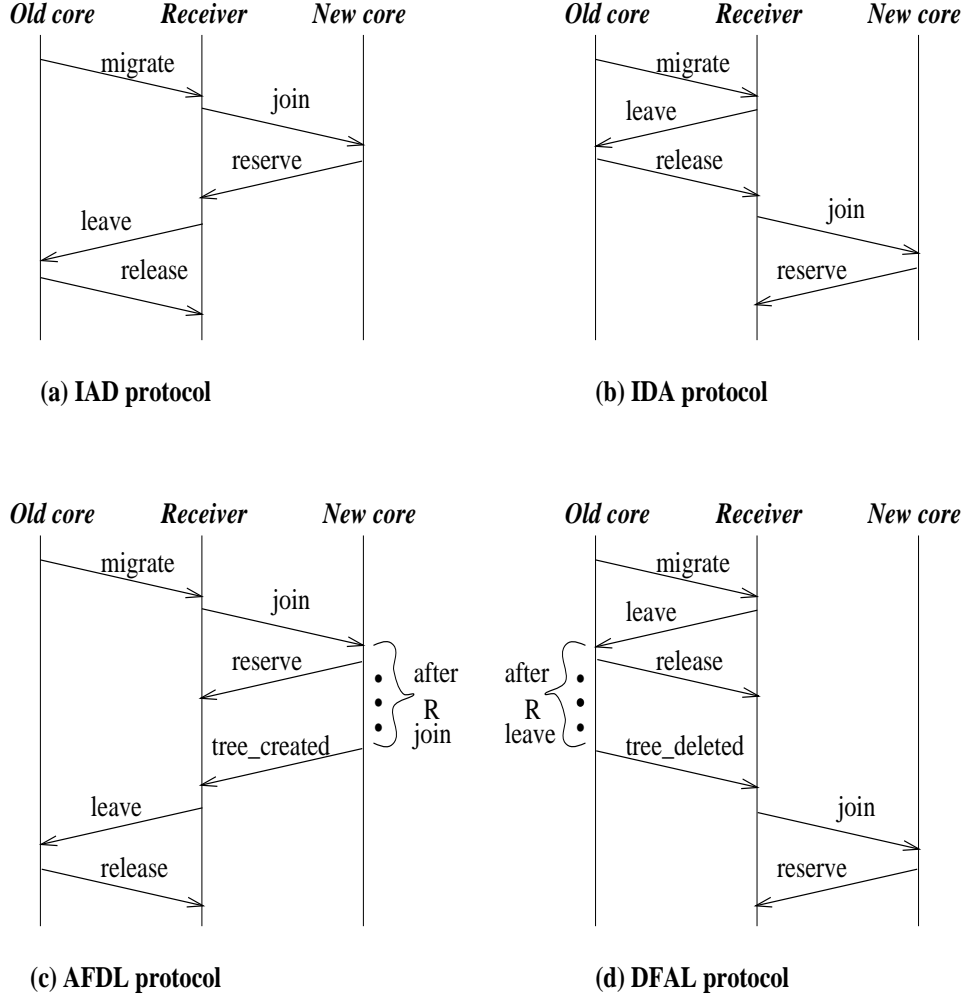


Figure 3: Proposed tree migration protocols

core and new core) have the knowledge of group membership and network topology. Whereas, the other two protocols (AFDL and DFAL) suffer due to these requirements.

3 Simulation Studies

We have conducted extensive simulation studies, using NS [34], to evaluate the effectiveness of the proposed tree migration algorithms. For our experiments, the various inputs were generated as follows.

- Random network topologies were generated based on a given input parameter “graph density”. This parameter determines the degree of the nodes and hence the connectivity of the network. Higher its value, the denser the topology is.
- The selections of source and receivers for a given multicast session are uniformly distributed

from the node set.

- For each simulation point, approximately 500 core migrations were triggered.
- The initial and subsequent selection of core node for a given multicast session is uniformly distributed from the group members (i.e., source and receivers).
- The multicast tree based on the new core was constructed using the following two different approaches.
 1. *Shortest Path Tree (SPT) Approach:* Combining the shortest (unicast) paths between the core and each receiver. The multicast tree thus created may not be cost² optimal, but is amenable for distributed implementation. Also, it is highly scalable.
 2. *Centralized Steiner Heuristic (CSH) Approach:* Using a centralized algorithm that exploits the membership and topology information. We have used KMB heuristic [18] for this purpose, whose cost ratio bound is 2. Though this approach has the potential to offer a multicast tree that has better (smaller) cost than the previous approach, its distributed implementation is difficult and hence not scalable.

The experiments were conducted in two parts: (i) evaluation of tree migration algorithms measuring resource wastage and packet loss incurred by them - the tree migration algorithms are independent of the multicast tree construction approach used - and (ii) evaluation of multicast tree construction approaches measuring the cost of the multicast tree produced by them. For both the parts, parameters such as number of receivers, graph density, and number of nodes in the network were varied. In our studies, the values of these parameters were taken to be 10, 3, and 31, respectively, when these parameters were not varied.

3.1 Evaluation of Tree Migration Algorithms

Tree migration algorithms described above are evaluated based on the defined performance metrics. The effect of (i) number of receivers, (ii) node degree and (iii) number of nodes in the network are studied.

3.1.1 Effect of Number of Receivers

The total packet loss is plotted in Figure 4(a) for varying number of receivers. It is obvious to see that the total packet loss incurred by the algorithms increases with increasing number of receivers. Among the four algorithms, it can be seen that the packet loss experienced by AFDL is the lowest and DFAL is the highest confirming the very nature of their designs. The interleaved versions of

²Cost of the multicast tree is sum of cost of the tree edges.

these algorithms lie in between the performances of these extremes with IAD being closer to (in fact overlaps with) the AFDL, and IDA being closer to the DFAL algorithm.

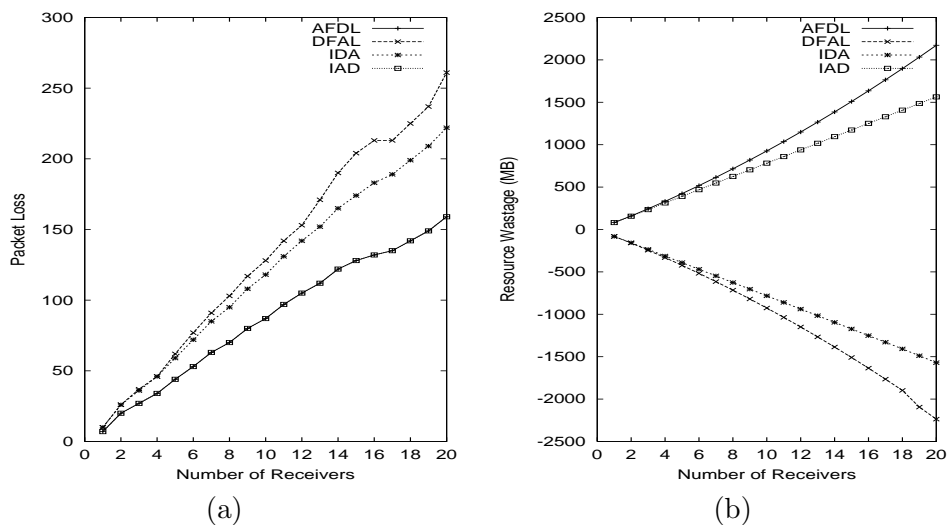


Figure 4: Effect of number of receivers on (a) packet loss and (b) bandwidth wastage

In Figure 4(b), the total resource wastage is plotted for varying number of receivers. Also, it is obvious to see that the total resource wastage incurred by the algorithms increases with increasing number of receivers. The resource wastage incurred by AFDL is the highest because it constructs the new tree before deleting the old tree, thus resulting in full/portion of both old and new trees consuming resources during the tree migration process. Algorithm IAD incurs the next highest resource wastage as it is the interleaved version of AFDL. Both DFAL and IDA do not incur any resource wastage because the receivers are first deleted and then added. The graph shows negative resource wastage for DFAL and IDA which means that the receivers have left (i.e., not part of) the old group earlier than they joined the new group.

3.1.2 Effect of Network Connectivity

The effect of graph density on packet loss for different tree migration algorithms is shown in Figure 5(a). As the graph density (average node degree) increases, the packet loss decreases almost linearly for all the four algorithms. This is because as the topology becomes denser, the chances of finding shortest paths to the receivers become higher. As a result, the packet loss decreases with increasing graph density. The resource wastage incurred by the algorithms decreases with increasing graph density for the same reason (shown in Figure 5(b)). The relative performances of the algorithms for both the metrics are similar to that in the previous study.

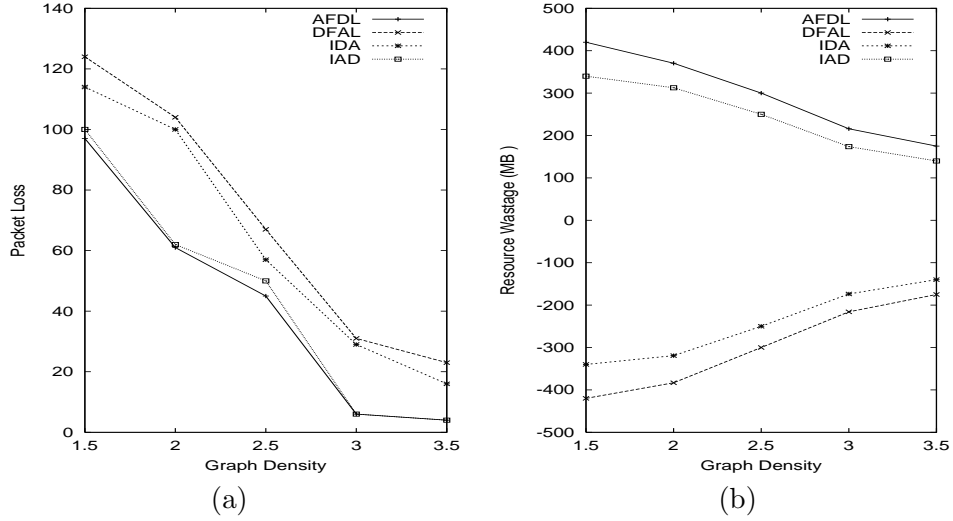


Figure 5: Effect of network connectivity on (a) packet loss and (b) bandwidth wastage

3.1.3 Effect of Number of Nodes

Figures 6(a) and 6(b) show the effect of varying number of nodes in the network on packet loss and resource wastage, respectively. Both Packet Loss and Resource Wastage increase with increasing number of nodes. The reason for this behavior is attributed to the sparse nature of the groups as the number of nodes increases. That is, for a given group size and graph density, the group becomes sparser when the number of nodes in the network increases because the number of hops separating the members increases. When the groups become sparser, the resultant trees have more links and hence more resource wastage and packet loss. The relative performances of the algorithms are the same as in sections 3.1.1 and 3.1.2.

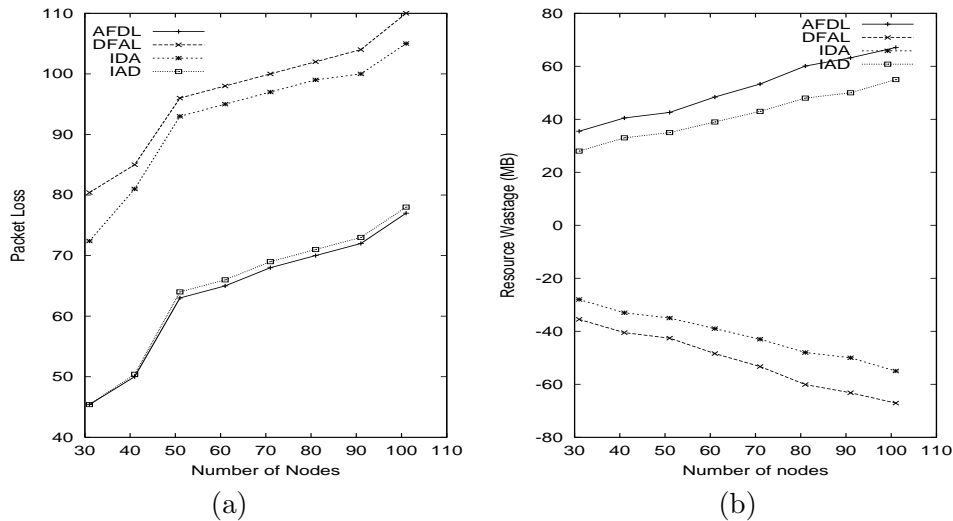


Figure 6: Effect of number of nodes on (a) packet loss and (b) bandwidth wastage

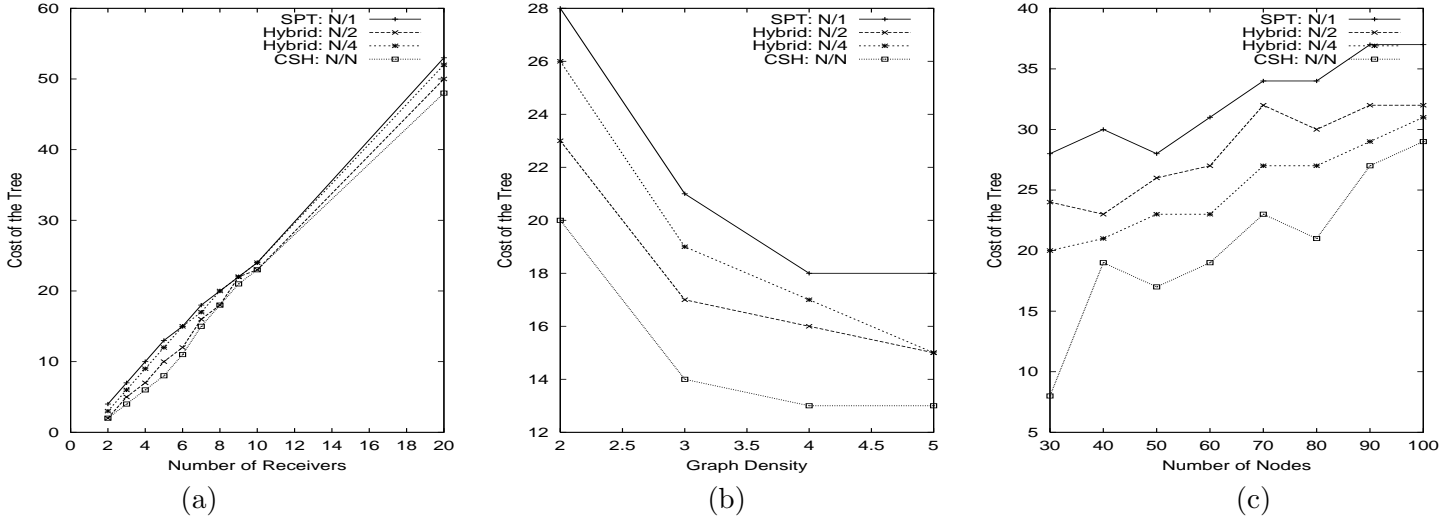


Figure 7: Effect of (a) number of receivers, (b) network connectivity and (c) number of nodes on tree cost

3.2 Evaluation of Multicast Tree Construction Approaches

Here, we present the studies that were carried out to evaluate the cost of the multicast tree produced by different multicast routing approaches. Let N be the number of receivers in the group. The receivers are divided into two partitions, one with K receivers and the other with $N - K$ receivers. In our study, the multicast tree was constructed using Centralized Steiner Heuristic (CSH) approach spanning K receivers and the remaining $N - K$ receivers were grafted (join) to this tree, using Shortest Path Tree (SPT) approach. When $K = N$, the algorithm reduces to CSH approach on N receivers; when $K = 1$, it is reduces to SPT approach. The value of K captures the tradeoff between the practicality of the approach and the cost of the tree offered by the approach. As mentioned earlier, $K = N$ (i.e., CSH) is the best in terms of tree cost and worst in terms of practicality, whereas $K = 1$ (i.e., the SPT) is the other way. In our simulation studies, we considered the following cases: $K = 1, K = N/4, K = N/2$, and $K = N$. The simulation results presented here are aimed at quantifying this tradeoff between the approaches.

In Figure 7(a), the cost of the multicast tree is plotted for four different values of K by varying the number of receivers. As the number of receivers increases, it is obvious to see that the cost of the tree also increases almost linearly. The figure also shows that for the CSH approach (referred to as N/N in the figure), the cost of the multicast tree is the minimum. The cost of the tree offered by SPT approach (referred to as N/1 in the figure) is the maximum. The cost of the tree in case of the other two algorithms $K = N/2$ and $K = N/4$ lie somewhere in between these two extreme cases. The interesting point to note here is that the cost of the tree offered by the SPT approach is very close to that of the CSH approach.

In Figure 7(b), the cost of the tree for varying graph density is plotted. When the graph density is increased, the cost of the tree decreases for most cases. The reason being the possibility of finding better shortest paths in denser networks and hence better cost trees as discussed before. The relative costs of the trees offered by SPT and CSH approaches remain more or less the same with increase in network density.

In Figure 7(c), the cost of the tree for varying number of nodes is plotted. The cost the tree increases with increasing number of nodes because of the sparse nature of the groups as discussed in section 3.1.3. The relative tree costs among different cases of the algorithms exhibit similar behavior as figures 7(a) and 7(b).

4 Integrated Tree Maintenance Framework

The main motivating factor behind any tree maintenance technique (local or global) is to improve the cost of the multicast tree. However, any tree maintenance technique involves certain amount of service disruption to the receivers. The local tree maintenance effects only a part of the multicast tree, so the nodes which are not part of the rearranging region are not effected by the maintenance process. Global tree maintenance, on the other hand, effects nearly whole of the multicast tree. Thus, local tree maintenance techniques offer less service disruption than the global tree maintenance techniques. However, global tree maintenance techniques improve the quality of the tree much effectively than the local tree maintenance techniques. By combining all the techniques within the maintenance framework, the trade-off between service disruption and the quality of the tree is effectively captured. This serves as a motivating factor for the integrated approach which combines the advantages of local and global tree maintenance techniques. In [35], we proposed a global tree maintenance technique called *tree evolution* which provides a balance between service disruption and tree cost. The integrated framework developed in this paper, switches between tree migration and tree evolution depending on number of cores the system can support.

4.1 Tree Evolution

Tree evolution can be informally defined as “tree migration over a period of time”. Whereas tree migration occurs almost instantaneously, the migrations of individual nodes under tree evolution are delayed until the QoS requirements of the members are violated. The goal of this delay is to buffer receivers against the effects of frequent core changes. The other nodes (whose QoS requirements are not violated) start an “evolution timer”, and evolve when the timer expires. In order to accomplish this goal, the behavior of the multicast tree must be slightly modified.

Rather than adding or removing existing links through traditional join/leave messages, the multicast tree is slowly evolved. A node evolves by converting shared links between the new tree

and old tree to be part of the new core and adding links wherever necessary. Therefore, pruning in the multicast tree under tree evolution occurs only when the path to the new core differs or *splits* from the old tree. This pruning is done to remove possible cycles from the tree as well as unnecessary links from the multicast tree.

The reasons for modifying the join/leave behavior are three-fold. First, the new join/leave behavior must be modified in order to allow members to correctly evolve to the new tree. Under tree evolution, join messages must be propagated until they reach a tree that is part of the core requested rather than simply stopping once an on-tree node has been reached. Otherwise, it is possible that a node would never evolve to the new tree if its path to the new tree lies through the existing multicast tree. Second, service disruption in the multicast tree can be minimized by not pruning the shared links between old and new core. Third, the new join/leave messages will prevent over-allocation of bandwidth since links can be converted from an old core to the new core without resource reallocation.

Evolution provides a trade-off between service disruption and tree cost. By delaying the process of migration over a period of time, evolution offers less service disruption and tolerance against “core thrashing”. On the other hand, trees under evolution has higher costs than those in case of migration.

4.2 Integrated Tree Maintenance Approach

In this subsection, we propose an integrated approach (shown in Figure 8) for tree maintenance which consists of both local tree maintenance mechanisms (graft/prune and tree rearrangement) and global tree maintenance mechanisms (tree migration and tree evolution). To the best of our knowledge, ours is the first work which investigates global tree maintenance techniques and integrates them with the local tree maintenance techniques.

The various tree maintenance techniques are invoked at different time scales and are event-driven:

- Graft/prune is invoked at a shorter time-scale and the triggering events are join and leave.
- Tree rearrangement is invoked at a medium time-scale and the triggering event is based on a quality index of a portion of the tree.
- Tree migration and tree evolution are invoked at a larger time-scale and the triggering events are based on the quality of the core and the quality of the tree.

Graft/Prune: The first component of the integrated framework is the member join/leave. When a member joins a multicast group, cost of the overall multicast tree should be taken into account in addition to the QoS requirements of the members.

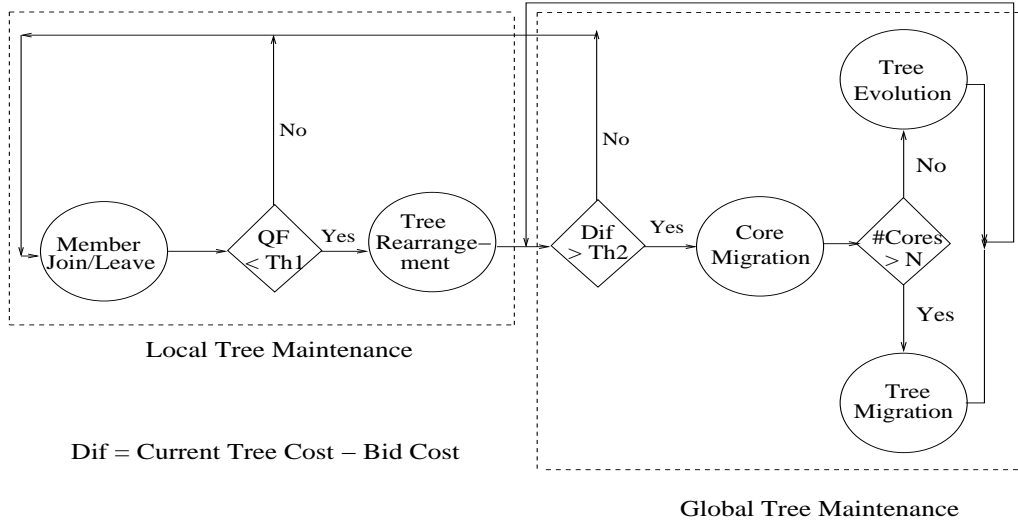


Figure 8: Integrated Tree Maintenance Approach

Tree Rearrangement: An on-line multicast routing algorithm must take into account two important and possibly contradicting goals: cost-reduction and minimization of service disruption. Therefore, a balance needs to be struck between these goals by employing tree rearrangement technique that monitors the quality of the tree and triggers tree rearrangement when the quality of the tree degrades below a threshold. For tree rearrangement, we consider CRCDM protocol described in [28]. In this protocol, multicast tree is divided into regions and each region defines a *Quality Factor (QF)* indicating the usefulness of the multicast region or M-Region. Higher QF indicates higher usefulness, as this means large number of member nodes which were in that region have since being deleted from the group. If the QF of a region falls below a threshold ($Th1$), then the region is rearranged.

Tree migration & Tree evolution: Tree rearrangement optimizes a portion of the multicast tree, but the overall cost of the tree may not be optimal. Hence, the current core needs to be migrated if there exists a node (new core) which can produce a multicast tree whose cost is better than the cost of the current tree by a threshold $Th2$. In Figure 8, decision to migrate or evolve depends on the number of successive cores that the group allows. It is to be noted that $N = 1$ means that evolution behaves similar to migration. As the number of successive cores in the tree increases, tree cost increases and service disruption decreases. To obtain a balance between service disruption and tree cost, the value of N is chosen to be 2 or 3. Extensive simulation studies are carried in [35] to justify the choice of N . As an example, let $c_1, c_2 \dots c_n$ be the n successive cores in the multicast session. Let the number of successive cores allowed in the group be also n . Let $n + 1$ be the $(n + 1)^{th}$ successive core. After core migration all members belonging to core c_1 migrate to c_{n+1} . Other nodes evolve as usual. Thus by migrating, the number of successive cores are kept

fixed at n , which helps to keep the overall cost of the tree in check.

5 Conclusions

In this paper, we highlighted the importance of tree migration as a mechanism for handling membership and network dynamics in multicasting and proposed heuristic algorithms and protocols for it. The proposed algorithms were evaluated under two performance metrics: service disruption and resource wastage. Our simulation studies show that the algorithms IAD and IDA offer performance comparable to that of AFDL and DFAL, in addition to being highly scalable and easily implementable. Our tradeoff studies on multicast tree construction approach have shown that the distributed SPT approach offers tree costs that are comparable (in most cases) to that of the centralized CSH approach, in addition to being highly scalable and easily implementable. The choice of the right combination of multicast tree construction approach and tree migration algorithm depends on various performance goals and varies with applications. In particular, the SPT based multicast tree construction with IAD/IDA tree migration algorithm is very attractive in terms of scalability and implementation complexity. Trade-off between service disruption and tree costs exists at a higher granularity between local tree maintenance and global tree maintenance techniques. To capture this trade-off, we also proposed an integrated scheme which encompasses local and global tree maintenance techniques. Currently, we are experimenting the integrated tree maintenance framework for various triggering events and time-scales.

References

- [1] C. Diot, W. Dabbous, and J. Crowcroft, "Multipoint communications: A survey of protocols, functions and mechanisms," *IEEE JSAC*, vol.15, no.3, pp.277-290, Apr. 1997.
- [2] J. Hou and B. Wang, "Multicast routing and its QoS extension: Problems, algorithms and Protocols," *IEEE Networks*, Jan./Feb. 2000.
- [3] L. Sahasrabuddhe and B. Mukherjee, "Multicast routing algorithms and protocols: A tutorial," *IEEE Network*, Jan./Feb. 2000.
- [4] J.C. Pasquale, G.C. Polyzos, and G. Xylomenos, "The multimedia multicasting problem," *Multimedia Systems*, vol.6, no.1, pp.43-59, 1998.
- [5] M. Ramalho, "Intra- and Inter- domain multicast routing protocols: A survey and taxonomy," *IEEE Communications Surveys and Tutorials*, vol.3, no.1, pp.2-25, Jan.-Mar. 2000.
- [6] S. Deering, C. Partridge, and D. Waitzmann, "Distance vector multicast routing protocol," *RFC 1075*, Nov. 1988.

- [7] J. Moy, "Multicast Extensions to OSPF", *RFC 1584*, Mar. 1994.
- [8] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei, "The PIM architecture for wide-area multicast routing," *IEEE/ACM Trans. Networking*, vol.4, no.2, pp.153-162, Apr. 1996.
- [9] Hugh. W. Holbrook, and David.R. Cheriton, "IP Multicast Channels: Express Support for Large Scale Single Source", *Proc. ACM SIGCOMM*, Sept. 1999.
- [10] T. Ballardie, P. Francis, and J. Crowcroft, "Core-based trees (CBT): An architecture for scalable inter-domain multicast routing," in *Proc. ACM SIGCOMM*, pp.85-95, 1993.
- [11] R. Perlman, C.-Y. Lee, A. Ballardie, J. Crowcroft, Z. Wang, and T. Maufer "Simple multicast: a design for simple, low-overhead multicast", *Internet Draft*, Feb. 1999.
- [12] M. Parsa and J.J. Garcia-Luna-Aceves, "A protocol for scalable loop-free multicasting, *IEEE JSAC*, vol.15, no.3, April 1997.
- [13] A. Striegel and G. Manimaran, "A Survey of QoS Multicasting Issues," in *IEEE Communications*, vol. 40, no. 6, pp.82-87, June 2002.
- [14] M. Handley, H. Schulzrinne, E. Schooler, J.Rosenberg, "Session Initiation Protocol (SIP)", *Proposed Internet Standard, RFC 2543, Internet Engineering Task Force (IETF) Multiparty Multimedia Session Control (MMusic) Working Group*, Mar. 1999.
- [15] C. Donahoo and E. Zegura, "Core selection methods for multicast routing," in *Proc. ICCCN*, 1995.
- [16] D.G. Thaler and C.V. Ravishankar, "Distributed center location algorithms," *IEEE JSAC*, vol.15, no.3, pp.291-303, Apr. 1997.
- [17] E. Fleury, Y. Huang, and P.K. McKinley, "On the performance and feasibility of multicast core selection heuristics," in *Proc. ICCCN*, pp.296-303, 1998.
- [18] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," *Acta Informatica*, vol.15, no.2, pp.141-145, 1981.
- [19] H. Takashami and A. Matsuyama, "An approximate solution for the Steiner tree problem in graphs," *Intl. J. Math Educ. in Sci. and Technol.*, vol.14, no.1, pp.15-23, 1983.
- [20] G. Manimaran and A. Chakrabarti, "A Scalable Approach for Core Failure Recovery in Multicasting", *Proc. Intl. Conf. on Advanced Computing and Communication (ADCOM)*, pp.191-196, Dec. 2000.

- [21] A. Banerjea, "Simulation study of the capacity effects of dispersity routing for fault-tolerant real-time channels," in *Proc. ACM SIGCOMM*, pp.194-205, 1996.
- [22] S. Han and K.G. Shin, "A primary-backup channel approach to dependable real-time communication in multihop networks," *IEEE Trans. Computers*, vol.47, no.1, pp.46-61, Jan. 1998.
- [23] R. Sriram, G. Manimaran, and C. Siva Ram Murthy, "An integrated scheme for establishing dependable real-time channels in multihop networks," in *Proc. ICCCN*, pp.528-533, 1999.
- [24] L. Schwiebert and R. Chintalapati, "Improved fault recovery for core based trees," *Computer Communications*, vol.23, no.9, Apr. 2000.
- [25] C. Donahoo and Zegura, "Core migration for dynamic multicast routing," in *Proc. ICCCN*, 1996.
- [26] B. Waxman, "Routing of multipoint connections," *IEEE JSAC*, vol. 6, pp.1617-1622, Dec. 1988.
- [27] F. Bauer and A. Verma, "ARIES: A rearrangeable inexpensive edge-based on-line Steiner algorithm," *IEEE JSAC*, vol. 15, pp.382-397, Apr. 1997.
- [28] R. Sriram, G. Manimaran, and C. Siva Ram Murthy, "A rearrangeable algorithm for the construction of delay-constrained dynamic multicast trees," *IEEE/ACM Trans. Networking*, vol.7, no.4, pp.514-529, Aug. 1999.
- [29] M. Faloutsos, A. Banerjea, and R. Pankaj, "QoS MIC: Quality of service sensitive multicast internet protocol", in *Proc. ACM SIGCOMM*, Sept. 1998.
- [30] Shigang Chen, Klara Nahrstedt, and Yuval Shavitt, "A QoS-Aware Multicast Routing Protocol", *IEEE JSAC*, vol.18, no.12, Dec. 2000.
- [31] G. Manimaran, Hariharan Shankar Rahul and C. Siva Ram Murthy, "A new distributed route selection approach for channel establishment in real-time networks", *IEEE/ACM Trans. Networking*, vol.7, no.5, pp.698-709, Oct. 1999.
- [32] A. Chakrabarti, G. Manimaran, "A Case for Scalable Multicast Tree Migration," *IEEE GLOBECOM*, Nov. 2001.
- [33] K. Carlberg, "QoS Multicast using single metric Unicast Routing", *PhD thesis, University College London*, Oct. 1999.
- [34] UCB/LBNL/VINT Network Simulator - ns (version 2), Available at www.mash.cs.berkeley.edu/ns/.

- [35] A. Chakrabarti, A. Striegel, and G. Manimaran, “A Case for Tree Evolution in QoS Multicasting,” in *Proc. IEEE/IFIP Intl. Workshop on QoS (IWQoS)*, pp.116-125, May 2002.