2003

Managing dynamic groups in QoS and overlay multicasting

Anirban Chakrabarti
Iowa State University

Managing dynamic groups in QoS and overlay multicasting

by

Anirban Chakrabarti

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Program of Study Committee:
Manimaran Govindarasu, Major Professor
Arun Somani
Ahmed Kamal
James Davis
David Fernandez Baca

Iowa State University

Ames, Iowa

2003

Graduate College
Iowa State University

This is to certify that the doctoral dissertation of

Anirban Chakrabarti

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Major Professor

Signature was redacted for privacy.

For the Major Program

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Multicasting has been the most popular mechanism for supporting group communication, wherein group members communicate through a multicast data distribution tree that spans all the members of the group. In a dynamic multicast session, members join/leave the group using graft/prune mechanisms, based on locally optimal paths, which would eventually degenerate the quality of the multicast tree. Therefore, efficient mechanisms need to be invoked periodically to maintain the cost of the multicast tree near optimal. However, tree maintenance would result in service disruption for the session. Therefore, there exists a trade-off between minimizing tree cost and minimizing service disruption. The goal of this dissertation is to develop and analyze a set of efficient tree maintenance techniques that aim to balance this trade-off in QoS and overlay multicasting. To achieve this goal, the dissertation makes three key contributions. First, the design of scalable protocols, viz. tree migration and tree evolution, for maintaining QoS multicast trees. Second, the design of an efficient strategy, called partial protection approach, and its implementation methods for member join problem with path reliability being a QoS constraint. Third, the design of an efficient tree maintenance algorithm, based on the idea of mesh-tree interaction, for end-system based overlay multicasting. The proposed tree maintenance solutions have been evaluated and analyzed through a combination of simulation and analytical studies. The studies show that the proposed solutions indeed achieve a good balance between tree cost and service disruption competitively.

CHAPTER 1 Introduction

The proliferation of Quality of Service (QoS) aware group applications associated with recent advancements in high-speed networks is driving the need for efficient multicast communication services satisfying the QoS requirements of such applications [1, 2, 3, 4, 5]. These group communication applications include video conferencing, shared workspaces, distributed interactive simulations, software upgrading, and resource location. The traditional unicast model is extremely inefficient for such group-based applications as the data is unnecessarily transmitted across the network to each receiver. On the other hand, multicasting has been a popular mechanism for supporting group communication. In a multicast session, the sender transmits only one copy of each each message that is replicated within the network and delivered to multiple recipients (receivers). For this reason, multicasting typically requires less total bandwidth than separately unicasting message to each receiver.

A number of unique issues arise out of the interaction between multicasting and networked multimedia system. Specific solutions are needed which address these unique features which are characterized by:

- Data must be sent to multiple destinations whose characteristics change with time.
- The volume of data is large and requires high bandwidth.
- The value of the data is sensitive to several quality of service parameters viz., delay, jitter, loss etc.

The difference between multicasting and separate unicasting is best captured by the *host group* model [6]: "a host group is a set of network entities sharing a common identifying multicast address, all receiving any data packets addressed to this multicast address by senders

(sources) that may or may not be members of the same group and have no knowledge of the groups' membership." This definition implies that, from the sender's point of view, this model reduces the multicast service interface to a unicast one. The host group model also allows the behavior of the group to be unrestricted in multiple dimensions: groups may have local (LAN) or global (WAN) membership, be transient or persistent in time, and have constant or varying membership. Consequently, we have the following types of multicast (or host) groups:

- *dense groups* which have members on most of the links or subnets in the network, whereas *sparse groups* have members only on a small number of widely separated links.
- *open groups* are those in which the sender need not be a member of the group, whereas *closed groups* allow only members to send to the group
- *permanent groups* are those groups which exist forever or for a longer duration compared to the duration of *transient groups.*
- *static groups* are those groups whose membership remains constant in time, whereas *dynamic groups* allow members to join/leave the group.

The multicast communication paradigm poses two unique issues that are not relevant to traditional point-to-point communication systems and make many traditional point-to-point solutions inextensible to the multicast environment.

- *Receiver Control:* A point-to-point communication has two participants, the sender/server and the receiver/client. The communication is initiated and terminated by any of the participants. On the other hand, error control and flow control are driven by the sender. The receiver's job in a point-to-point environment is to receive service and in some cases provide feedback to the sender regarding the service it receives. In a dynamic multicast session the scenario changes drastically as the receivers may join and leave the multicast session at any time, and the receivers can also have personalized QoS requirements and expect personalized flow control. Therefore, traditional source-driven solutions of the traditional point-to-point communication will not be able to accommodate the re-

quirements of a multicast session, and hence receiver-controlled techniques are needed to address this problem.

- *Heterogeneity & Group Dynamics Management:* Group communication has to deal with much higher degree of *heterogeneity* than point-to-point systems. In point-to-point systems, the heterogeneity comes from network wherein different portions of the network are composed of varying switches and links of different capabilities. In group communication, in addition to the network heterogeneity, the requirements of the hosts can differ, which adds a different dimensions. In addition, dynamic multicast sessions also have to deal with hosts leaving and joining the multicast group, and manage the multicast tree to cater to the changes. Therefore, in a dynamic multicast group, the three issues of *network heterogeneity*, *host heterogeneity* and *group dynamics* need to be addressed.

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 that is 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 1.1 the various steps and events that take place in the *"life-cycle"* [7] of a typical multicast session. The sequence of phases/steps are: (i) multicast group (session) creation, (ii) multicast tree construction with resource reservation, (iii) data transmission, and (iv) multicast session tear-down. A multicast routing protocol deals with constructing a multicast tree that spans group members and takes into account both network and membership dynamics.

1.1.1 Multicast Group Creation

The first step in the multicast life-cycle is the multicast group creation. Multicast group creation has two steps: (i) Allocation of group address, and (ii) Distribution of the address to group members. The first step of group creation is to assign a unique address to the multicast

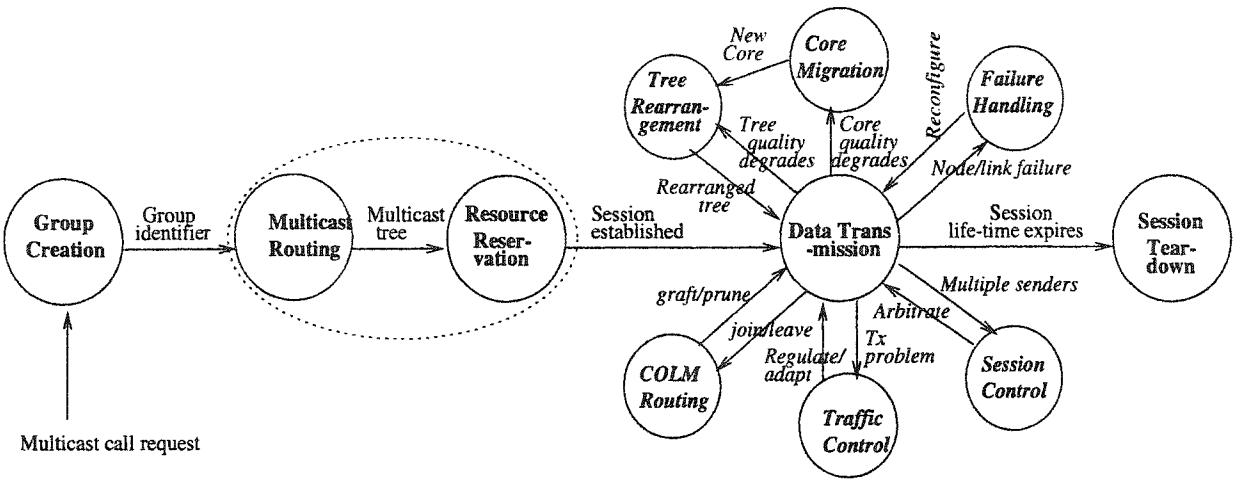


Figure 1.1 Life-cycle of a QoS multicast session - an event diagram view

group. When the group address is allocated, it needs to be distributed so that members can join the multicast group. Distribution of the group address forms the second step of the multicast group creation.

- *Allocation of Multicast Group Address:* The first step in the multicast group communication is the allocation of a unique address to the multicast group such that the data sent to a specific group does not clash with the data of other groups. The address associated with a multicast group has a lifetime assigned to it depending on the duration of the multicast session. Similar to groups, multicast group addresses can be *static* or *dynamic* in nature. In the case of a permanent group, a static group address is assigned to the multicast group. On the other hand, in case of transient groups, multicast addresses are assigned for the duration of the multicast session and can be reassigned when the multicast session gets over. It is to be noted that assigning static group addresses to a transient groups may not only result in unnecessary address usage, but also may result in insecure communication, where non-members may receive messages meant for some other group. On the other hand, allocation of dynamic addresses to permanent groups may result in unnecessary communication overhead. For the purpose of multicast address allocation, the Multicast Address Allocation (MALLOC) working group has defined three protocols which work together to form a global dynamic multicast address allocation mechanism.

These protocols include:

- A "host to Address Allocation Server" protocol used by a host to obtain one or more multicast addresses from an address allocation server within its domain.
- An intra-domain server to server protocol that address allocation servers within the same domain can use to ensure that they do not give out conflicting addresses.
- An inter-domain protocol to provide aggregatable multicast address ranges to domains, which the servers in that domain can then allocate individual multicast addresses based on those.

- *Distribution of Group Address:* When a multicast service has been developed and is about to start transmission it may wish to announce itself to hosts; otherwise interested hosts will have no knowledge of the service, its location, content, etc. Several protocols have been developed and standardized which is used for multicast session announcement like Session Announcement Protocol and the Session Description Protocol. These protocols encapsulate the information about the service into an advertisement and send it to a group of listeners. The listeners can be hosts willing to join the multicast session, or servers providing session directory service to other hosts willing to join the session. The advertisement or announcement usually contains a description of the service's location, content, availability, bandwidth requirements, etc.

1.1.2 Multicast Tree Construction

Once the group is created, the next phase in the multicast session is the construction of a multicast distribution tree, spanning the source(s) and all the receivers (QoS routing), and reserving resources on the tree. Multicast route determination is traditionally formulated as a problem related to tree construction. There are three reasons for adopting a tree structure:

- The source needs to transmit a single packet down the multicast tree.
- The tree structure allows a parallel transmission to multiple receiver nodes.

- The tree structure minimizes data replication, as the branching nodes are only used for data replication.

It has been established that determining an optimal multicast tree for a static multicast tree can be modeled as a *Steiner tree problem* [8]. The Steiner tree problem can be formally defined as: Given a graph $G = (V, E)$, a cost function $C : E \rightarrow R^+$, a set of nodes $M \subseteq V$, find a subgraph $H = (V_H, E_H)$ of G such that $M \subseteq V_H$ and the cost $C(H)$ (equal to the sum of the cost of the edges in E_H) is minimized.

The Steiner tree problem, defined above, has been proved to be NP-Complete. A number of algorithms have been developed using heuristic based approaches such as KMB [9] and TM [10].

- *KMB Algorithm:* In the KMB algorithm, first the shortest paths between each pair of multicasting member nodes is computed and a closure graph is created only containing these multicasting members. The edge between each pair of nodes in the closure graph is the shortest distance between them. Then the minimum spanning tree of this closure graph is computed using Prim's algorithm and finally the edges in the closure graph are replaced with the corresponding shortest paths in the original network to get the Steiner tree.

- *TM Algorithm:* The shortest paths from all the nodes (including member and non-member nodes) to all the multicasting member nodes in the graph is computed. When the Steiner tree is built, the multicasting node is selected to join the tree which is nearest to any of the nodes in the partially generated tree. When all the multicasting member nodes have joined, the algorithm finishes. Both KMB and TM algorithms run in $O(|V|^2|M|)$ time and have an approximate worst case bound of 2.

In addition to construction of a multicast tree, QoS requirements in the form of delay, jitter, loss, reliability etc. need to be satisfied during tree construction itself. The problem becomes more complex if the multicast session is dynamic in nature and trees need to be reconstructed to deal with the group dynamics.

In addition to tree creation, resource reservation [12] is needed to guarantee QoS in terms of delay, jitter, loss, reliability etc. for multimedia applications. Hence, the data transmissions of the connections will not be affected by the traffic dynamics of the other connections sharing the common links.

1.1.3 Multicast Data Transmission

Four different kinds of run-time events that can occur during the transmission phase of a multicast session (refer Figure 1.1): (i) membership changes, (ii) node and/or link failures, (iii) transmission problem, (iv) competition among senders

- *Membership Changes:* Since group membership can be dynamic, 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 (identified as COLM (constrained online multicast) routing in Figure 1.1).

- *Node and/or Link Failures:* During the life-time of a multicast session, if any node or link supporting the multicast session fails, the service will be disrupted. This requires mechanisms to detect node and link failures and to reconfigure (restore) the multicast tree around the faulty components with minimal service disruption (identified as Failure Handling in Figure 1.1). There has been significant research work in dealing with transmission problems and session control [11].

- *Transmission Problems:* This could include events such as swamped receivers (needing flow control), overloaded intermediate nodes (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 intermediate nodes, is responsible for performing the necessary control activities to overcome these transmission problems (identified as Traffic Control in Figure 1.1).

- *Competition among Senders:* In a many-to-many multicasting, when multiple senders share the same multicast tree (resources) for data transmission, resource contention occurs among the senders. This will result in data loss due to buffer overflow, thus triggering transmission problems. A session control mechanism is required to arbitrate transmission among the senders (identified as Session Control in Figure 1.1).

In Figure 1.1, Tree Rearrangement and Core Migration can be invoked when the quality of the tree degrades due to membership dynamics or when node/link failure occurs.

1.1.4 Session Tear-Down

At some point of time, when the session's lifetime has elapsed, the source will initiate the session tear-down procedures. This involves releasing resources reserved for the session along all the links of the multicast tree and purging all session specific routing table entries. Finally, the multicast address is released and group tear-down is complete.

1.2 Dissertation Contribution

The dissertation makes the following key contributions:

- *Protocols for tree maintenance:* In this dissertation, two protocols for tree maintenance have been developed, viz., Tree Migration and Tree Evolution. The former moves the members from one tree to another all at once, while the latter staggers the process to reduce service disruption. Extensive simulation and analytical studies have been performed to evaluate the effectiveness of the protocols. The results show that Tree Evolution is able to achieve a balance between tree cost and service disruption.

- *QoS and Reliability Constrained Multicast Routing:* In this dissertation, a Partial Protection Approach (PPA) has been proposed, where protection is provided in some domains so that the reliability constraints of the receivers are satisfied. Three schemes, viz., Conservative, Optimistic and Hybrid schemes have been proposed and evaluated through simulation studies.

- *Tree maintenance in End System Multicasting:* This dissertation provides a new mesh management technique in End System Multicasting (ESM), which is a new paradigm in multicasting. The technique is called Mesh Tree Interaction (MTI) where mesh is constructed based on the underlying multicast tree. MTI has been compared with the existing ESM protocols and the results show that it is able to produce better quality trees than the existing protocols.

CHAPTER 2 Background and Motivation

Multicast communication is handled by creating a multicast tree spanning all the members who are part of the multicast group. In a QoS multicast session, the construction of multicast tree is important as the nature of the tree determines the QoS received by the members supported by the multicast group. Several multicast tree construction approaches have been proposed which can be classified into three main approaches: source based, core based and hybrid approaches. In addition to the construction of the multicast tree, dynamic multicast sessions require management of the multicast trees as members join and leave the multicast session. In core based multicasting, the core of the multicast tree selected "degenerates" with time, and new cores need to be selected and the members need to be scalably moved from one tree to another. Therefore, managing group dynamics includes *pre-session issues* like core selection and tree construction, and *on-session issues* like tree maintenance. This chapter details the issue of managing group dynamics and provide motivation and background for the dissertation work.

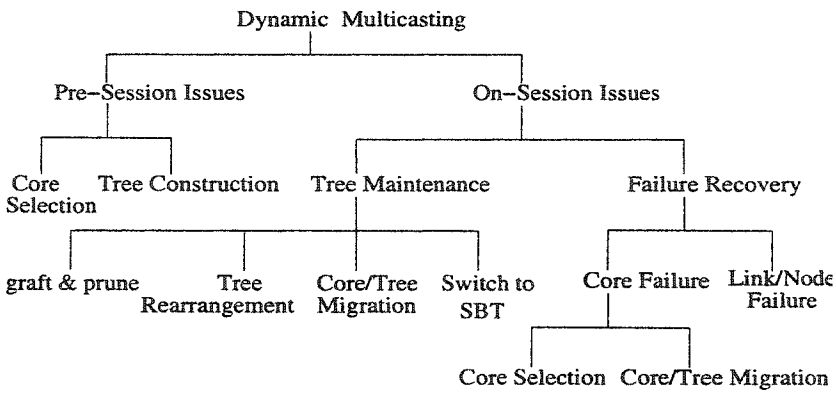


Figure 2.1 Issues in dynamic multicasting

The main issues associated with dynamic multicasting are shown in Figure 2.1. In the next few sections, the main issues viz., tree construction, core selection and tree maintenance, are discussed with background work in each of these areas.

2.1 Multicast Tree Construction

Two types of tree construction mechanisms are generally used in multicasting: (i) Source based, where the multicast tree is rooted at the source node and (ii) Core-based where the multicast tree is rooted at some node called the center node or the "core" node.

- *Source Based Protocols:* The source-based approach uses the notion of a shortest path tree (SPT) rooted at the sender/source. Each branch of the tree is the shortest path from the sender to each group member. Since the shortest path is usually the shortest delay path, the receivers in the multicast tree typically receive good delay performance. However, source-based trees introduce scalability problems as the protocol not only requires separate trees for each sender, but also each router needs to maintain per-group and per-source information. The Distance Vector Multicast Routing (DVMRP) [13], Multicast Extensions to Open Shortest Path First (MOSPF) [14], Protocol Independent Multicast-Dense Mode (PIM-DM) [15], and Explicitly Requested Single-Source Multicast (EXPRESS) [16] are examples of source-based trees.

- *Core Based Protocols:* Core-based or shared-tree protocols, construct a multicast tree spanning the members whose root is the center or *core* node. These protocols are highly suitable for sparse groups and are scalable for large networks. However, just as shortest-path trees provided excellent QoS at the cost of network bandwidth, shared trees provide excellent bandwidth conservation at the cost of QoS to the receivers. The Core Based Tree (CBT) [17] is a well-known example of a shared tree routing protocol. When a node wishes to transmit a message to the multicast group in the CBT protocol, the node sends the message towards the core. The message is distributed to group members along the path to the core and the message is distributed to the remaining members once it

reaches the core. Requests to join or leave the multicast group are processed by sending the request towards the group core. When a join request reaches a tree node, the tree node becomes the point of attachment for the new node. Conversely, when a node leaves a group, the part of the tree between the node and nearest tree node whose degree is greater than two is pruned.

The two approaches mentioned above have their own advantages and disadvantages. The two approaches are compared as follows:

- Core based schemes are receiver driven approaches and adapt well to dynamic scenarios.
- Performance of core-based schemes depend largely on the selected core, which is an NP-Complete problem [18], unlike that of the source-based schemes.
- Core based schemes are more suited for sparse groups unlike the source-based schemes.

To take advantages of the trade-offs provided by the two protocols, some hybrid protocols are also being developed. These protocols create core-based multicast trees at the start and then switch to a source based tree when the performance of the multicast tree degrades. Examples of such protocols are PIM-sparse [15] and Multicast Internet Protocol (MIP) [19].

2.2 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 [18, 20] for core selection and are evaluated in [21]. In all the core selection algorithms, a certain number of nodes called the *potential cores* express their willingness [18] 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. Several popular core selection heuristics are described below:

- *Optimal Center Location:* In OCBT [22] algorithm, the center is chosen by calculating the potential cost of the tree when rooted at each network node and choosing to locate the tree at the node that gives the lowest maximum delay among all those with minimum cost. This is an expensive solution, and hence not very practical.

- *Random Center Location:* In this type of technique, a node is randomly chosen as the center of the multicast tree. CBT [17] and PIM [15] use a variation of this scheme, as in both these schemes, the first node joining the group is chosen as the center. Random center location algorithms are simple to implement, however the performance of the multicast trees are greatly influenced by the center and hence result in unpredictable delay performance.

- *Minimum Center Location:* This approach requires calculating the potential cost for trees rooted at each group member and then choosing the member with the lowest cost. It has been shown that [23] this algorithm performs close to the Optimal Center Location algorithm under most cases.

2.3 Tree Maintenance

In dynamic multicasting, the multicast tree degrades over time due to continuous grafting and pruning [24]. 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, this dissertation focuses on the problem of tree maintenance.

Tree maintenance, as mentioned in the Figure 2.1, can be invoked at a local as well as global level. In the 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 case of global tree maintenance, 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 affected by the members joining/leaving the group. Both these techniques affect 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 ([25, 26, 27]). 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 until 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 [3] 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 [28], QMRP [29] and parallel probing [30] 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. Details of three common routing protocols are given below:

- *QoSMIC:* In QoSMIC, the search of candidate paths proceeds in two main directions, namely local search and tree search. In local search, all the nodes in the neighborhood are searched through a flooding procedure limited by a time-to-live (TTL) value. The

tree search procedure is invoked when the local search fails to get an on-tree node in the neighborhood. In the tree search procedure, the joining node contacts a designated Manager router, which in turn orders a subset of on-tree nodes to send a BID request to the joining node. The joining node then decides to join the on-tree node providing the best path.

- *QMRP:* The QMRP protocol consists of a single path and a multi-path mode. The protocol starts by finding the shortest path route to the source of the multicast group. If resources cannot be met with a single path mode, it switches to multi-path mode by backtracking to the previous hop. The previous hop then sends the request packet to all adjacent nodes. This process is continued until the source of the multicast tree is reached. While QoSMIC users a centralized manager, QMRP tries to create the path using a distributed approach.

- *Parallel Probing:* The protocol was originally proposed for QoS unicast routing and can be adopted to QoS multicast routing also. The objectives of the protocol are to minimize the path setup time and resources reserved along the multiple candidate paths. To achieve this, the member sends multiple probes using different heuristics for each probe, to an intermediate destination (ID). Upon receiving the first probe message the ID initiates a parallel probe to the next ID. Upon receiving later probes, the ID releases the resources reserved between current ID and the previous ID (or members) by those later probes.

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 [3]. 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 the graft/prune mechanism [3]. Due to continuous grafting and pruning, the quality of the multicast tree deteriorates over time. In that case, a part of the multicast tree is rearranged to improve the quality of the multicast tree. This technique of rearranging the part of the tree is identified as tree rearrangement. Some of the common tree rearrangement algorithms are GREEDY [25], ARIES [26] and the CRCDM [27]. 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" [24] 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, thus this process falls under the category of global tree migration. Though, over the years researchers have admitted the importance of core migration ([4], [24]), 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 are [31, 32]. In [32], Carlberg described a very simple core migration strategy where the 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 the 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, the new core has to be shifted outside the existing tree and then tree migration becomes inevitable. Therefore, tree migration is a more general strategy to optimize the cost of the tree than the simple strategy described in [32].

2.4 End System Multicasting

With the advent of overlay networks [33, 34], where the nodes arrange themselves in an overlay, researchers are examining the possibilities of having network functionalities like packet

forwarding [35], multicasting [36, 37] etc. at the application layer. As an alternative to IP multicasting, researchers have proposed the overlay multicasting approach [36, 37, 38, 39], wherein the complex multicasting features like replication, group membership management and multicast routing are implemented at the application layer, assuming only the end-systems or hosts are responsible for multicasting. End System Multicasting (ESM) is an example of overlay multicasting. In ESM, the end-systems organize themselves in an overlay spanning tree for data delivery. Each link in the ESM spanning tree corresponds to the unicast path in the actual physical network. As all the complexities are handled by the hosts rather than by the routers, ESM offers some distinct advantages over its IP counterpart. The advantages are: (i) ESM is easier to implement, as there is no complexity required at the routers. Therefore, it is also scalable. (ii) Complex functionalities like congestion control and reliable data transfer are handled separately at the unicast level, and therefore manageable. (iii) Adding security features to multicasting is easier as routers are not involved.

In spite of these advantages, ESM has some issues which need future research attention. (i) The quality of the multicast tree produced using ESM is worse than that produced using IP multicasting. In multicasting the quality of a multicast tree may refer to the cost of the multicast tree, or the average delay to all the receivers, or some other optimization metric. (ii) Since each node in the ESM tree is a host, therefore the nodes have limited capability in terms of bandwidth and processor capabilities. This is abstracted as the *fanout constraint* which identifies the number of outgoing links that the multicast tree can support. (iii) The multicast sessions are unreliable as they depend on the hosts for data transmission.

Therefore, IP multicasting offers performance benefits while ESM offers simplicity in implementation. ESM is still in its infancy and many issues need to be resolved before it can become a serious alternative to IP multicasting. Recently, efforts to combine IP and ESM has also taken place [40]. In this dissertation, this new trend in multicasting research is recognized and solutions are proposed for building efficient and scalable multicast trees for ESM.

2.5 Motivation and Objectives

Managing group dynamics remains one of the most challenging areas in the field of QoS multicasting. In a highly dynamic group multicast trees 'degenerate' over time. Tree rearrangement [25, 26, 27] maintains a part of the multicast tree. Another method for tree maintenance involves reconstructing the multicast tree and moving the members from the old tree to the new one. This type of tree maintenance reduces the cost of the multicast tree by incurring huge amounts of service disruptions to the members. Therefore there exists a trade-off between tree cost and service disruption. No research effort has been undertaken which adequately deals with trade-offs between service disruption and tree cost issues. This dissertation attempts to fill this void and develops protocols and algorithms to maintain multicast trees in scalable manner which provide trade-offs between tree cost and service disruption.

The development of efficient protocols which establish multicast sessions based on QoS constraints and can handle node/link failures is an important problem. Robustness of a path to the user can be measured by the reliability of the path. The reliability of a path depends on dynamics factors such as the link characteristics, congestion on the path and so on. Since service disruption depends on the reliability of the path, therefore it is important to take reliability into account during multicast member join. Efficient multicast join/leave protocols based on different path and link constraints ([28, 29]) have been studied in the literature. Also, several efforts have been undertaken to deal with node/link failures ([41]). However, efficient protocols that establish connections based on QoS and reliability constraints have not been studied in the literature. This dissertation is one of the few works which includes reliability as a QoS parameter and develops tree maintenance protocols taking reliability into account.

End System Multicasting (ESM) is fast becoming an alternative to QoS multicasting. ESM tries to reduce some of the disadvantages of the IP multicasting architecture. Hence, it provides a trade-off between performance and deployability. More details on this area in provided in Chapter 6. Application layer multicasting offers different sets of challenges to handle group dynamics. Therefore the protocols developed for IP multicasting cannot be trivially extended to application layer This dissertation also deals with developing scalable tree maintenance

techniques in ESM also.

The dissertation deals with managing group dynamics in QoS multicasting which has the following objectives:

- To develop scalable protocols for tree maintenance for QoS multicasting, which achieves a trade-off between tree cost and service disruption.
- To develop scalable protocols for member join/leave using reliability as the QoS parameter.
- To develop scalable tree maintenance protocols for End System Multicasting, which achieves a balance between performance and scalability.

2.6 Dissertation Organization

The objectives mentioned in the previous sections are realized through (i) The development of tree maintenance approaches in QoS multicasting, (ii) The development of a reliability constrained multicast routing protocol, and (iii) The development of group management techniques in End System Multicasting. The rest of the dissertation is organized as follows:

- In Chapters 3 and 4, novel methods of tree maintenance are proposed namely, Tree Migration and Tree Evolution respectively. Tree Migration is a more abrupt method of moving the members from one tree to another, while Tree Evolution is more adaptive.
- In Chapter 5, a novel approach of multicast member join using reliability constraints called Partial Protection Approach (PPA) is developed. In this chapter three different schemes to implement PPA are also being discussed and evaluated using simulation studies.
- In Chapter 6, multicast tree maintenance mechanisms are developed for End System Multicasting, which is a new paradigm of multicast research.
- In Chapter 7, some concluding remarks are made and pointers are provided for future research directions.

CHAPTER 3 Tree Migration

When the multicast group is highly dynamic, the core of the multicast group "degenerates" [24] 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 a new tree needs to be created in place of the old tree. Though, over the years researchers have admitted the importance of core migration ([4], [24]), the actual process of moving the members from one tree to another has never been explicitly dealt with. To the best of our knowledge, other than our work [31] the only work in the area of global tree maintenance is [32]. [31] is part of the research dissemination. In [32], 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 [32].

As part of the research, two techniques of migrating members from one tree to another have been explored. The approaches are called *tree migration* and *tree evolution*. While the former involves migration of members from one tree to another, the latter is a more adaptive one. In this chapter, the research on tree migration with results and studies are discussed. The details of tree evolution are found in the subsequent chapters.

3.1 Different Tree Migration Protocols

Tree migration involves migration of individual members from the old core to the new core. The tree migration protocols have to be *resource efficient* and should offer minimum *service disruption* to the receivers. Four protocols of tree migration have been developed, which enforce the trade-off between service disruption and resource wastage in varying degrees. In case of Add First Delete Last (AFDL) protocol, all the members join the new tree and then get deleted from the old tree. The reverse happens in case of Delete First Add Last (DFAL) protocol. Interleaved Add Delete (HAD) and Interleaved Delete Add (IDA) are per receiver implementation of AFDL and DFAL respectively. The protocols are illustrated in Figures 3.1(a)-(d).

IAD Protocol: Figure 3.1(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 3.1(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.

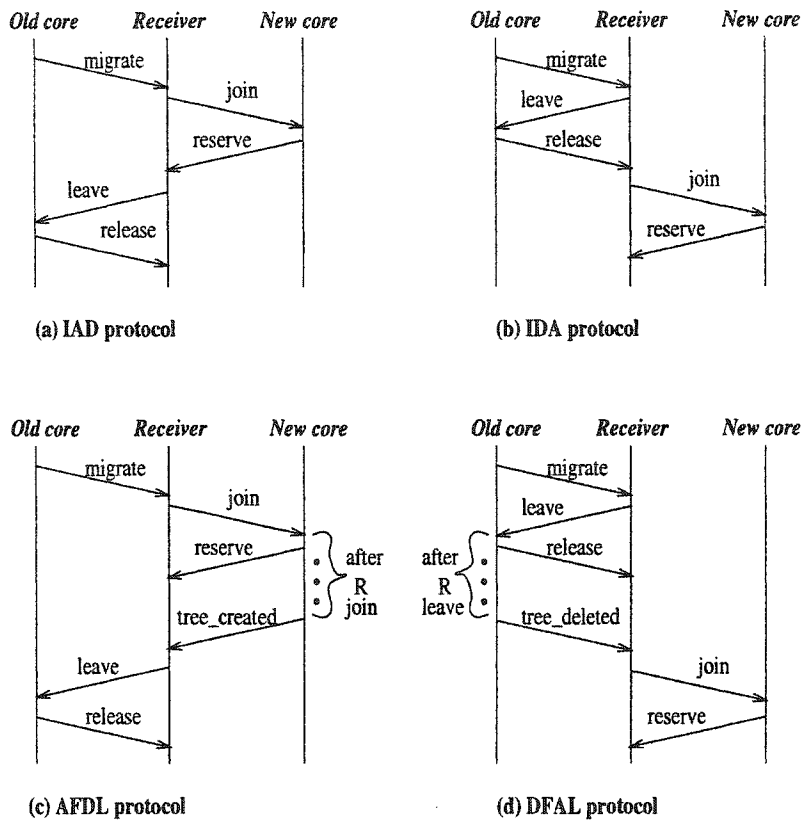


Figure 3.1 Proposed tree migration protocols

AFDL Protocol: Figure 3.1(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 3.1(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 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.

3.2 Performance Studies

To evaluate the effectiveness of the proposed tree migration algorithms, simulation studies were carried out using NS [42]. For the 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, denser the topology. [1]
- 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).

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 cost 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. KMB heuristic [9] has been used 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.

[1] The random topologies were generated by randomly adding links to a random tree maintaining the graph density and keeping the graph connected.

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.

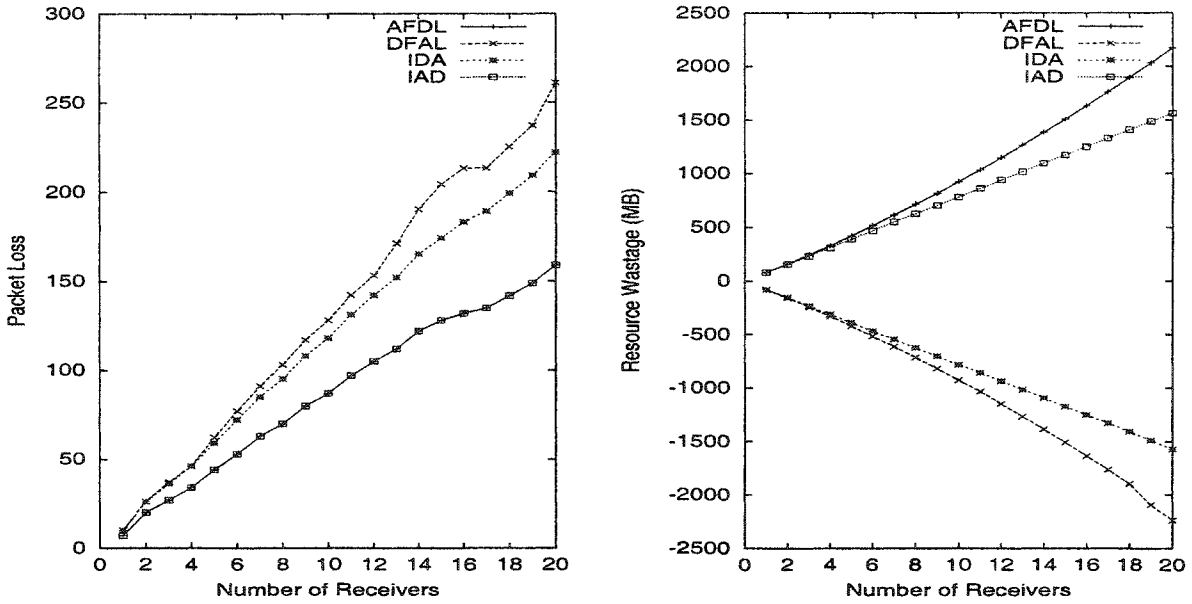
3.2.1 Evaluation of Tree Migration Algorithms

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.

Effect of Number of Receivers

The total packet loss is plotted in Figure 3.2(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 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.

In Figure 3.2(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



(a) (b)

Figure 3.2 Effect of number of receivers on (a) packet loss and (b) bandwidth wastage

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.

Effect of Network Connectivity

The effect of graph density on packet loss for different tree migration algorithms is shown in Figure 3.3(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 3.3(b)). The relative performances of the algorithms for both the metrics re similar to that in the previous study.

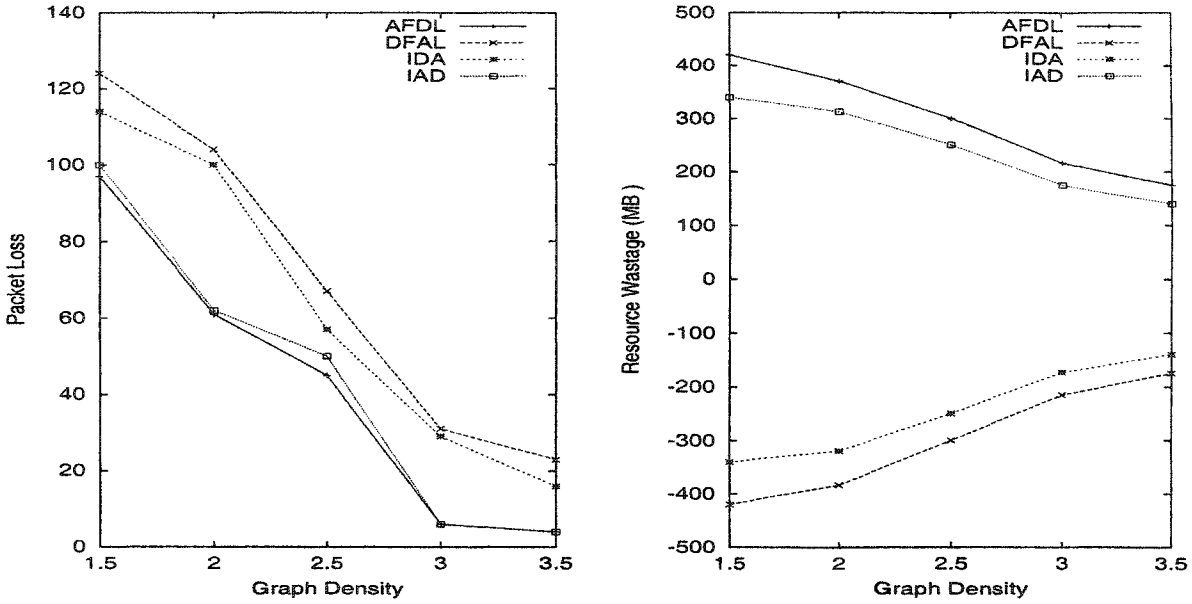


Figure 3.3 Effect of network connectivity on (a) packet loss and (b) bandwidth wastage

Effect of Number of Nodes

Figures 3.4(a) and 3.4(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 the 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 section 3.2.1.

3.2.2 Evaluation of Multicast Tree Construction Approaches

Here, the studies that were carried out to evaluate the cost of the multicast tree produced by different multicast routing approaches, are presented. Let N be the number of receivers in the group. The receivers are divided into two partitions, one with K receivers and the

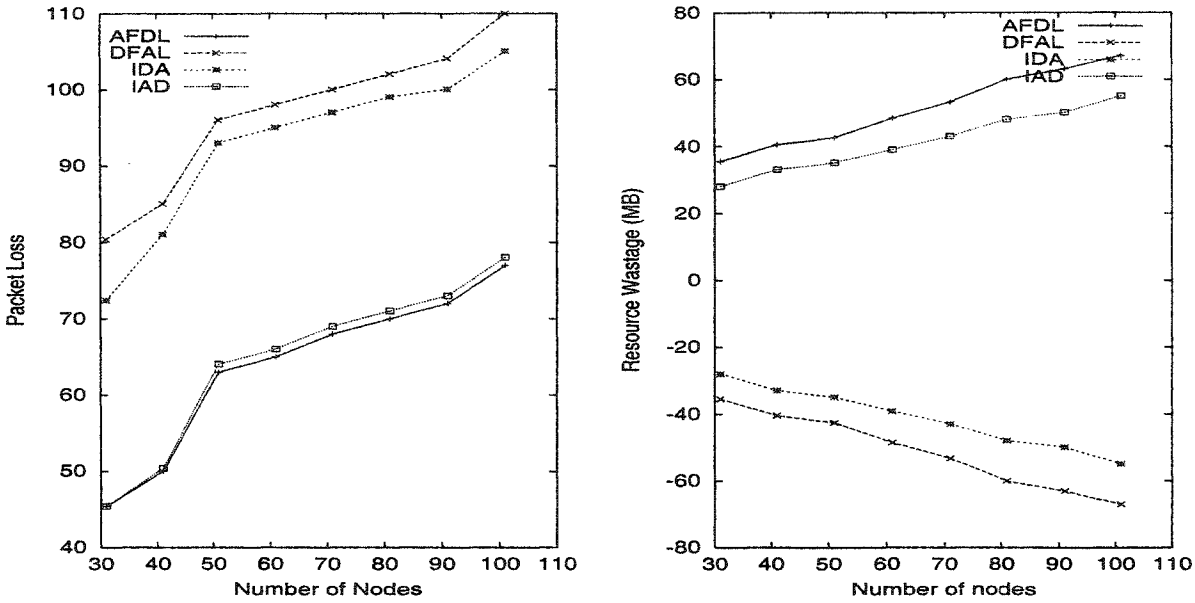


Figure 3.4 Effect of number of nodes on (a) packet loss and (b) bandwidth wastage

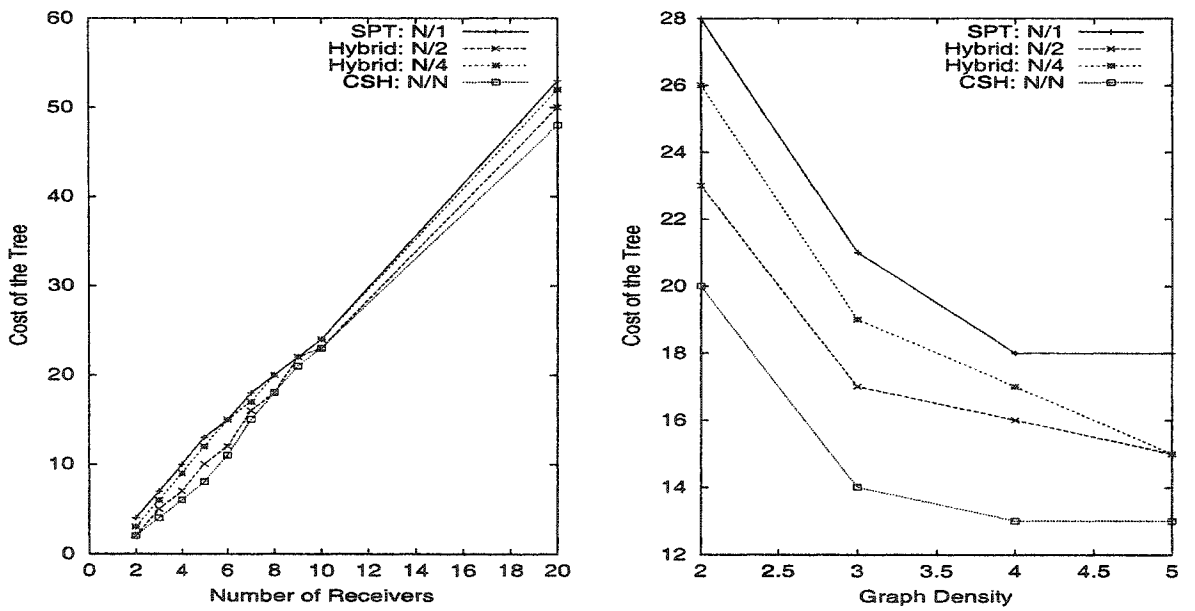


Figure 3.5 Effect of (a) number of receivers and (b) network connectivity on tree cost

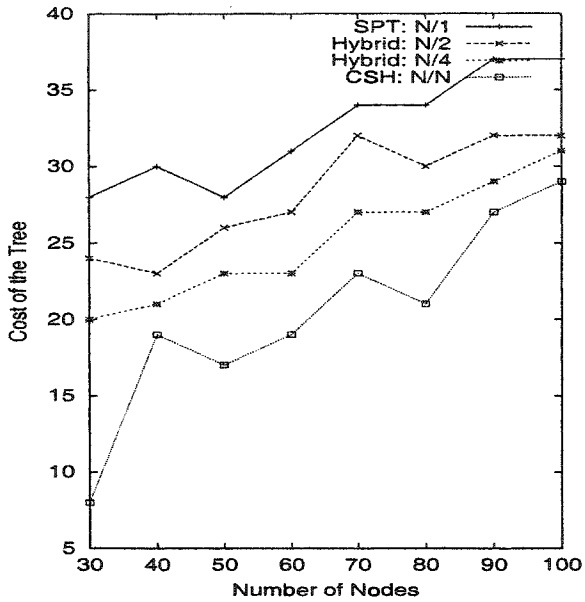


Figure 3.6 Effect of number of nodes on tree cost

other with $N-K$ receivers. In the 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 trade-off 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 the simulation studies, The following cases were considered: $K=1, K=N/4, K=N/2$, and $K=N$. The simulation results presented here are aimed at quantifying this trade-off between the approaches.

In Figure 3.5(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 3.5(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 3.6, 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 3.2.1. The relative tree costs among different cases of the algorithms exhibit similar behavior as Figures 3.5(a) and 3.5(b).

3.3 Summary

In this chapter, the importance of tree migration as a mechanism for handling membership and network dynamics in multicasting has been highlighted and heuristic algorithms and protocols has been developed. The proposed algorithms are evaluated under two performance metrics: service disruption and resource wastage. The 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. The 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.

3.4 Tree Migration and Beyond

Depending on the method used for tree migration, core migration can suffer from several key problems when groups are highly dynamic. If the core migration mechanism is continually invoked, members will be forced to migrate from one core to another when in actuality the multicast tree may be changing only slightly. This excessive migration also results in additional control message overhead as well as large amount of service disruption. Therefore , there is a need for developing a mechanism that does not aggressively migrate all the members from one core to another. In addition, the mechanism should be adaptable, so that the trade-off between cost and service disruption can be properly handled. This motivates the development of a more adaptable version of tree migration called "Tree Evolution", which is discussed in detail in Chapter 4.

CHAPTER 4 Tree Evolution

As the core of the multicast group is selected based on the initial group membership, therefore the core of the multicast group needs to be changed when the group is highly dynamic. To account for the core changes, the multicast group members need to be scalably moved from one tree to another without incurring enough service disruption. One way to move the members from one tree to another has been mentioned in Chapter 3 is called tree migration, where the members are moved from one tree to another together. However, there is a need for a more adaptive mechanism to handle group dynamics. Such an adaptive approach is named as "Tree Evolution", and is discussed in this chapter. In tree evolution, members are moved from one tree to another based on evolution timer. The rest of the chapter discusses the tree evolution concept and the implementation of the concept in the form of a protocol called Split Based Tree Evolution Protocol (STEP) in detail. Detailed simulation studies and analytical studies have also being discussed in this chapter.

4.1 Split-based Tree Evolution Protocol (STEP)

Tree evolution can be informally defined as *tree migration over a period of time.* Whereas tree migration occurs almost instantaneously, the migrations of group members under tree evolution are delayed until various trigger conditions are met. The goal of this delay is to reduce the effect of frequent core changes on the receivers. In order to achieve this goal, the join/leave behavior of the multicast group needs to be minimally modified. In addition to grafting/pruning paths through traditional join/leave messages, the multicast tree is evolved slowly. During evolution, 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. The evolution model

is based on a concept called *split*. A split in a multicast tree is said to have occurred at node X: "if X is part of the selected route to the new core and if there exists at least one link of X which is part of the old tree." Pruning the multicast tree under tree evolution occurs only when the path to the new core *splits* from the old tree. Pruning is done to remove possible cycles from the tree as well as unnecessary links from the multicast tree. In this model, splits play an important role. Hence, the proposed model is named STEP (Split-based Tree Evolution Protocol) due to its reliance on the concept of *splits* for pruning.

4.1.1 STEP Overview

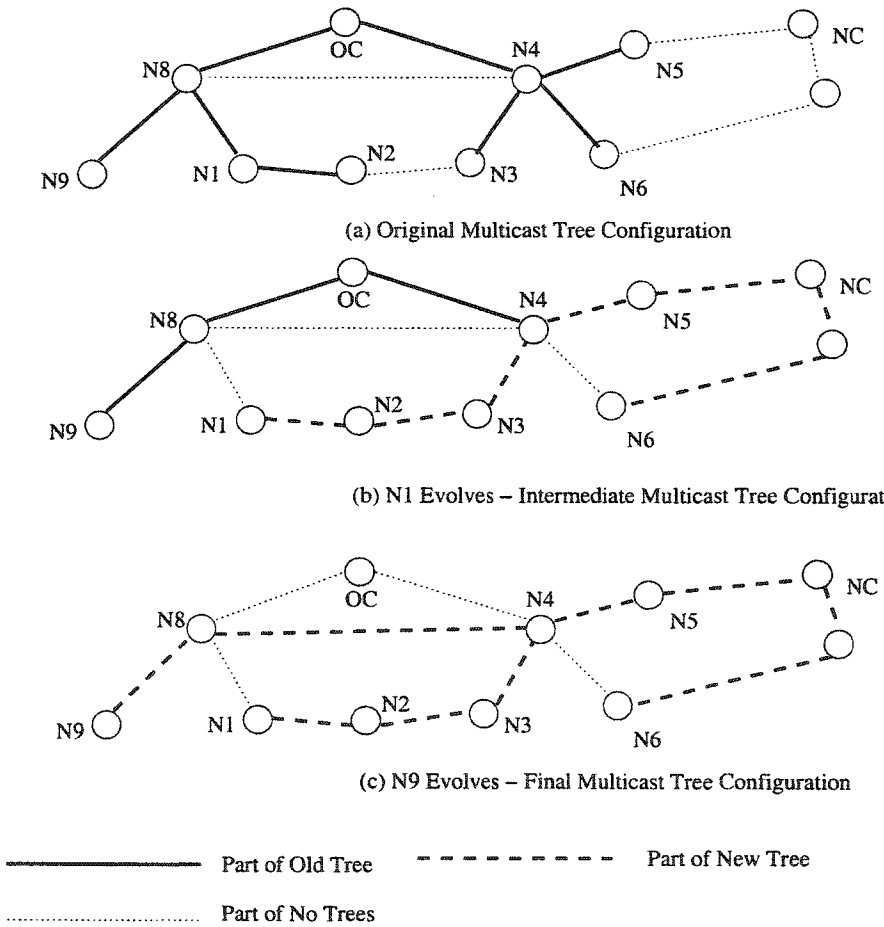


Figure 4.1 An illustration for tree evolution

In Figure 4.1, an overview of the STEP protocol is provided. Let OC be the old core of

the multicast tree, and NC be the selected new core. N_1, N_2, ... N_9 are the members of the multicast group. The goal of tree migration/evolution protocols is to migrate/evolve all members from a tree rooted at OC to a tree rooted at NC. Under migration, members N_1, N_2 ... N_9 send prune messages along the old tree towards OC and graft messages towards NC. Referring back to the Figure 4.1, under migration the final tree configuration is created in one abrupt step (from Figure 4.1(a) to (c)). Migration, thus involves 9 link additions, and 8 link deletions. Hence, it does not take advantage of the links that are shared between the two trees. In the example, let the cost of links, which are part of the old tree, be 2 and cost of all the other links be 1. Therefore, the cost of the old tree was 18 and the cost of the new tree is 13. Let the analysis of the average cost of the tree be calculated over a span of 3 time units. The migration process ends in time unit 2, so the the average cost of the tree is $(16 + 13 + 13)/3 = 14.0$.

Evolution, unlike migration, delays the process over a period of time. So, evolution may have different members attached to different cores. To ensure that every member is able to receive packets, NC joins to the multicast tree rooted at OC (at N_6 shown in the Figure 4.1). Let N_1 be the node which decides to migrate as its QoS is been violated (Details of the events that trigger evolution are discussed in Section 4.2). It is to be noted that links ($N_1 - N_2$ and $N_4 - N_3$) have already been reserved as part of the old tree. STEP protocol takes this observation into consideration by adding only those links which are unique to the new tree (links $N_5 - NC$ and $N_2 - N_3$) and prunes those links which would result in formation of loop (links $N_1 - N_8$ and $N_4 - N_6$). The configuration of the multicast tree after the evolution of node N_1 is illustrated in Figure 4.1(b). It is to be noted that nodes N_2, N_3, N_4 and N_5 fall along the path from N_1 to NC, so they also evolve along with N_1. However, nodes N_8 and N_9 still remain attached to the old core. Let the evolution of N_9 be triggered next. Figure 4.1(c) shows the final configuration. Evolution, because of selected grafting and pruning, has only 5 link additions and 4 link deletions. Service disruption is proportional to the changes made to the multicast tree (link deletion and addition), thus evolution protocol clearly provides less service disruption than migration. Taking the same link costs as above, the average cost of the

Table 4.1 STEP message list

Message	*Description*	*Parameters*
Evolve-Join	New join message (both for member and New Core)	Receiver, NewCore, GroupID
Evolve-Join-OK	Response - Evolve-Join Success	Receiver, NewCore, GroupID
Evolve-Join-Fail	Response - Evolve-Join Fail	Receiver, NewCore, GroupID, Reason
Evolve-Prune	Prune cycles and unnecessary links	GroupID
Evolve-NewCore	Announce new core to group	NewCore, GroupID
Evolve-QoS	Receiver $\rightarrow$ Sender - QoS problems with sender	GroupID, Core

tree produced by STEP is $(16 + 16 + 13)/3 = 15.0$. This simple example illustrates our claim that STEP is able to achieve a trade-off between service disruption (lower than migration) and tree cost (higher than migration).

Subsequently, in this section, the STEP protocol is explained in more detail with illustrations provided in Subsection 4.1.4.

4.1.2 STEP Behavior

In order to accommodate the different requirements of the evolution protocol, the join and leave processes of traditional multicasting are slightly modified. The reasons for modifying the join/leave behavior are three-fold. First, to allow members to correctly evolve to the new tree. Under tree evolution, a join message is propagated until it reaches the requested core 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 the old and the new cores. 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.

Table 4.1 lists the new messages introduced by the STEP model. STEP behavior can be classified as:

- *Core selection and notification:* In this phase, a new core is selected and the information is notified to all the members of the multicast group.
- *Member joining/evolving:* Both new member joining the multicast group, and member evolving to a new core from an old core is same under STEP as both uses *EvolveJoin* message.
- *Member Leaving:* Member leaving the multicast group is essentially the same as in traditional multicasting. In STEP *EvolvePrune* message is used for this purpose.

Core Selection and Notification

Core change notification is essentially done in STEP using *EvolveNewCore* message, which is sent by the node at which the new core attaches itself to the multicast tree. The nodes, upon receiving the *EvolveNewCore* message, not only store the information about the new core but also the direction from which the *EvolveNewCore* message arrives. The different steps of this phase are as follows:

1. A new core is selected using a core selection algorithm described in [20].
2. The new core joins to the old tree using *EvolveJoin*, and attaches itself to a node in the old tree called the *attachment point.*
3. The attachment point sends *EvolveNewCore* message to all the members of the multicast tree.
4. All members on receiving the *EvolveNewCore* message stores the new core information (NewCore) and the direction from which the *EvolveNewCore* message arrived (AttachDirection).

```
Evolve::ReceiveEvolveJoin(Member N, Node R, Core NewCore)
   If (R equals NewCore)
      SendEvolveOK(N);
      return;
   If (ResourceNotReserved())
      if (ReserveResources() equals FAILED)
         SendEvolveJoinFailed(N);
         return;
   forall (link of R)
      if (SplitExists(link))
         set Split(link)=1;
   SendEvolveJoin(NewCore);
   return;
```

(a)

```
Evolve:ReceiveEvolveJoinOK(Member N, Node R)
   if (R equals N)
      return;
   forall (link of R)
      if (Split(link) equals 1)
         SendEvolvePrune(AttachDirection);
         SendEvolveJoinOK(N);
   return;
```

(b)

Figure 4.2 (a) Algorithm ReceiveEvolveJoin (b) Algorithm ReceiveEvolve-JoinOK

Member Join/Evolve

The key difference between tree migration and STEP lies in the *Evolve-Join* message (details in Figure 4.2). Whereas tree migration uses the traditional join/leave messages to move between cores, evolution simply evolves the links from an older core to the new core. The main differences with the *Evolve-Join* message versus a traditional multicast join message is that the *Evolve-Join* message may not necessarily allocate resources on a link. An *Evolve-Join* message will only allocate bandwidth if the link is not currently on the multicast tree. If the link is on the multicast tree but belongs to a different core, the link will be evolved to belong to the new core without any resource allocation. However, unlike a traditional multicast join, an *Evolve-Join* message will continue to propagate towards the new core. In this process, it is possible to introduce cycles into the multicast tree unless proper precautions are taken. In the core evolution model these potential cycle locations are identified as *splits*. In Figure 4.2, *SplitExists* function identifies the splits.

A *split* in the multicast tree during an Evolve-Join signifies that a cycle may be formed during evolution of the member to the new core. It is also possible that multiple splits may occur when joining towards the new core. In order to remove these potential cycles from the tree, the questionable links are marked into the *split* state (identified as setting of variable Split in Figure 4.2(a)) during the forward pass of the *Evolve-Join* message. On the reverse pass, *Evolve-Join-OK* (algorithm in Figure 4.2(b)), the potential cycles are removed by sending an *Evolve-Prune* message towards the attachment point of the new core. As was stated earlier, the direction of the attachment point is saved when a node receives the *Evolve-NewCore* message. By sending a prune message towards the attachment point of the new core, the link immediately following the node is guaranteed to be pruned which ensures that the new multicast tree will be cycle-free.

Member Leave

Members leaving a multicast group is essentially same as in traditional multicasting. STEP uses *EvolvePrune* message for pruning the links in the multicast tree, which is identical to the traditional multicast leave message. However, the use of the *Evolve-Prune* message in

tree evolution is what distinguishes the message from the traditional leave message. As with the traditional leave message, an *Evolve-Prune* message is sent towards the core whenever a member wishes to leave the group. In addition, it is also used whenever a node evolves to the new core if there is a split. The steps are enumerated as follows:

1. Node N sends EvolvePrune message towards the core.
2. Node R receives EvolvePrune.
3. R releases the reserved resources.
4. If R has a link which is part of the new core, exit.
5. Forward EvolvePrune to the next node until current core is reached.

4.1.3 Link and Node Categorization

To incorporate the various features of the STEP model, the links and nodes are categorized based on which core the links/nodes are part of. Also, nodes require to store some additional split-related information.

Link Categorization

The links of the tree are categorized according to which core the link belongs to. Each link in the network can be classified into one of three states for each multicast group that is active on the network:

- *Not used:* The link is not used for the multicast group.
- *New core:* The link is part of the tree for the current core of the group.
- *Older core:* The link is part of the tree for one of the previous cores for the group.

Along with the *split* concept, the notion of a link membership provides the foundation for STEP. From this, the behavior of the join/leave messages can be appropriately modified to become *link/core-aware* rather than the traditional node/tree aware. This is necessitated by

the fact that different links may belong to different cores and the join/leave messages need to be aware of that.

Node Categorization

For each link that is part of the multicast tree for the group, the node maintains a 3-tuple in relation to which core the link was allocated towards as well as if the link is part of a possible split. This tuple consists of: $< Interface, CoreID, IsSplit >$, where CoreID is the core the link is attached to, and IsSplit indicates whether split has been identified in the link. This information is maintained relative to each group. Each node can be in one of the following states:

- *New core only:* The node has links that are part of the multicast tree for the group and all of those links are paths to the new core.
- *Old core only:* The node has links that are part of the multicast tree for the group and all of those links are paths to an old core.
- *No core:* The node does not have any links that are part of the multicast tree for the group.
- *Mixed cores:* The node has links that are part of the multicast tree for the group of which some are part of the paths to the new core and others are part of the tree for old cores.

In addition to the link information for each group, a node also maintains the appropriate evolution information for each group. This information includes the current group core, the evolution period and parameters mentioned in Section 4.2.

4.1.4 Examples of Member Evolution

Figure 4.3 demonstrates the behavior of one node evolving to the new core. This example describes in detail the evolution of node N_1, as mentioned in Figure 4.1. The new core (NC) has already joined to the old core (OC) and become part of the multicast tree. The evolution timer for node N_1 has expired and N_1 begins the evolution process by sending an *Evolve-Join*(G_x,NC,N_1) message towards NC, where G_x is the group to which N_1 belongs.

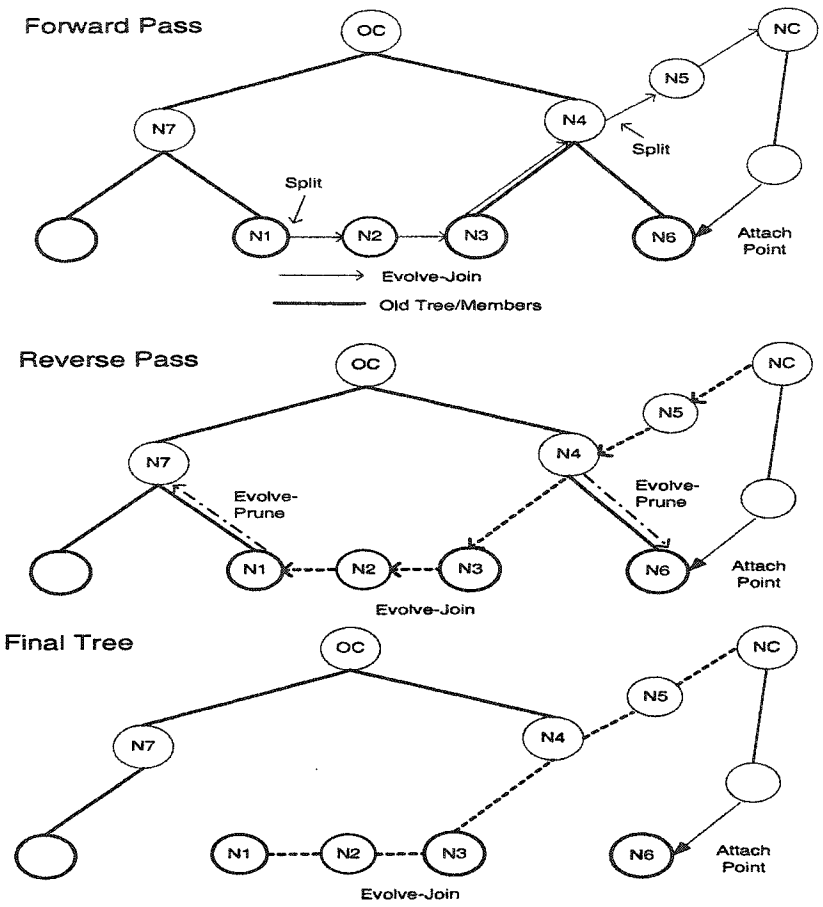


Figure 4.3 Tree evolution - a simple example

Forward Pass: The message is first processed by N_1 itself before being sent out onto the network. N_1 detects a split in the tree since the new path to NC is separate from any of the links for the old tree. N_1 then marks the outgoing link (for the new path) as being part of a *split* and sends the message onwards. As shown in the Figure 4.3, *Evolve-Join* message continues until N_3 where it meets back again with the old tree. However, unlike the traditional join, the *Evolve-Join* message has not yet met up with any part of the new tree and therefore continues to propagate towards the new core. The next path from N_3 continues along the old tree to N_4. Since the outgoing path is part of the existing tree, a *split* has not occurred. However, this is not the case at N_4 where the next hop towards NC *splits* off from the old tree. At N_4, another split occurs and the outgoing link towards NC is marked as being part

of a *split*. From N_4, the *Evolve-Join* message is propagated to NC where it meets up with the new tree. Referring to Figure 4.3, once the *Evolve-Join* message successfully reaches the new core NC, the new core sends an *Evolve-Join-OK* message back to the node. The contents of the *Evolve-Join-OK* message are the same as the contents of the *Evolve-Join* message.

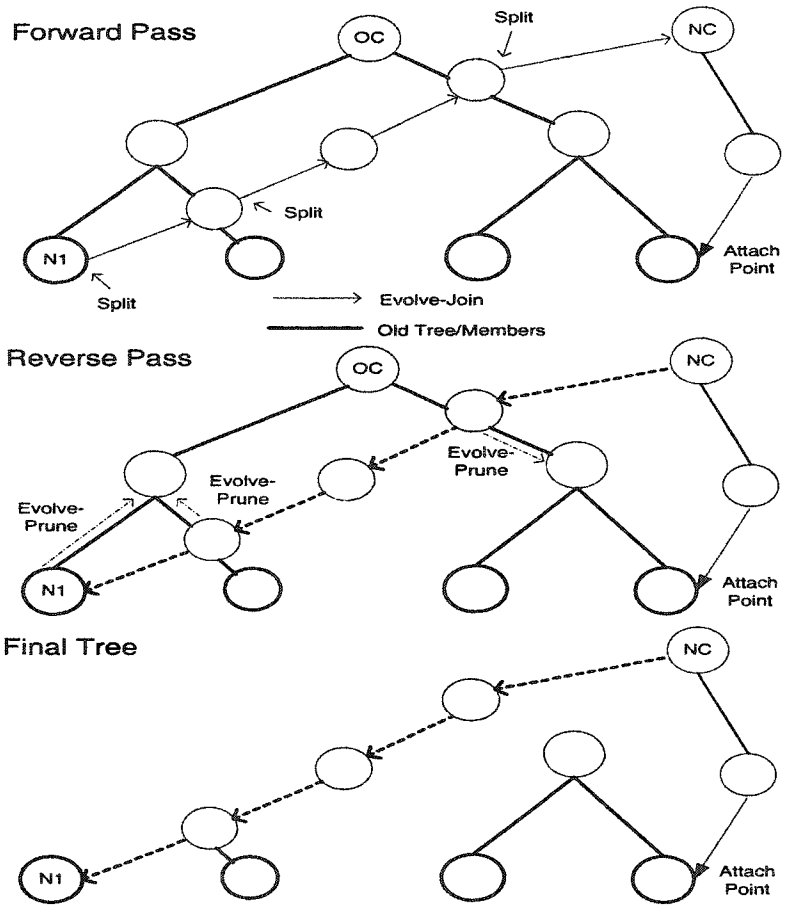


Figure 4.4 Tree evolution - a complex example

Reverse Pass: The *Evolve-Join-OK* message follows the same path back to N_1 from NC. When the message reaches N_4, the node acts on the possible cycle that may be formed by enabling the new link. Before any traffic may be accepted coming from the direction of N_5, the node must act on the potential cycle. Since the incoming link from N_5 to N_4 has already been marked in the *split* state for group G_x. Upon receiving the *Evolve-Join-OK* message, the node will remove the potential cycle by sending an *Evolve-Prune* message towards the attachment point of the

NC to the old tree. The attachment point can be saved simply as the direction from which the *Evolve-NewCore* message arrived. Following the arrival of the *Evolve-Join-OK* message, an *Evolve-Prune* message is sent towards N_6 with the G_x as the only parameter. As the *Evolve-Prune* message is propagating towards N_6, the *Evolve-Join-OK* message is propagated towards N_3. When the *Evolve-Prune* message reaches N_6, the message stops propagating after removing the link since N_6 is a member of the multicast group. Meanwhile, the *Evolve-Join-OK* message propagates back to N_1 since no *splits* had occurred along the path. Upon reaching N_1, the response message has now reached another *split* in the multicast tree. Once again, an *Evolve-Prune* message is spawned towards the attachment point of the new core to the multicast tree. For this *Evolve-Prune* message, the message is propagated to N_7 where it stops since N_7 has downstream children.

Figure 4.4 shows an alternative evolution tree. In this figure, the elegance of the evolution model is truly demonstrated as the *split* concept elegantly cleans up old links in the multicast tree. The concept of the *split* captures unnecessary links in the new multicast tree as well as avoiding cycling in the multicast tree.

4.2 Triggers for Evolution

Since the evolution of nodes to the new tree is not automatic as with traditional tree migration, there exists factors which trigger the evolution of members within the multicast tree. Under STEP, the two triggers for migration are QoS violations (violation of delay constraints) and time. If a node on the old tree experiences a QoS violation, the node evolves to the new tree. This evolution should occur immediately in order to minimize service disruption and begin receiving a better QoS. If a node is receiving adequate QoS while part of the old core, the node should still be encouraged to evolve to the new core. This should occur gradually in order to reduce potential service disruption.

QoS-based Evolution

All nodes suffering from QoS violation evolve immediately. The first cause of poor QoS arises from the location of the new core. Since there is an additional delay from the new core to the

old tree, non-member senders may violate the QoS of receivers because of this additional delay while transmitting to the new core. The second cause of poor QoS arises from the transitional nature of tree evolution. Since the tree is continually evolving, various regions of the tree may be entirely in the new tree and other regions entirely in the old tree, thus resulting in poor performance until the tree evolves further.

Thus, for all QoS violation cases, it is imperative that the node that has experienced a poor QoS takes action to improve its QoS. If the node is part of the old tree, it is the responsibility of the node to evolve to the new tree. However, once the node is part of the new tree, it is still possible to experience poor QoS due to the arrangement of the new/old tree. In this case, it is again the responsibility of the node to correct this poor QoS. The node then sends a unicast *Evolve-QoS* message to the sender who caused the QoS violation. For the sender, the receipt of an *Evolve-QoS* message means that the sender should migrate to the new core. For non-member senders, this simply means that messages should now be sent to the new core instead of the old core. Note that if a sender receives an *Evolve-QoS* message while belonging to the new core, this means that the multicast tree may not be adequate to suit all the members of the tree. In this case, an appropriate course of action should be taken such as immediately selecting a new core or other courses of action.

Timer-based Evolution

The second type of evolution that can occur is evolution over time. Although a member may be experiencing satisfactory QoS, the cost of the multicast tree may not be minimal. Thus, the member should be encouraged to evolve to the new core. In order to be considered for timer-based evolution, a member must be a leaf on the multicast tree. This is to prevent upstream members from adversely affecting downstream members on the multicast tree by joining to the new core. Consider the case where a parent node with K children evolves to the new core. However, the path from the parent to the new core is different than that of the children. As a result of evolving the parent node, all K children have a chance to have their service disrupted when the nodes themselves did nothing to change the multicast tree.

Thus, each leaf of the multicast tree should be encouraged to migrate to the new core

gradually. This is achieved through *evolution timers*. It is assumed that a node will be able to identify itself as a leaf on the multicast tree. Upon receiving the new core message, each leaf begins an evolution timer based on the evolution period contained within the *Evolve-NewCore* message. Each leaf sets its evolution timer based on the following equation:

$$TimerEvolve = Random(0, T) \tag{4.1}$$

where T is the period of the evolution timer, and *Random* returns a uniform random number between 0 and T.

Whenever activity is detected (a join or leave message for the group), the evolution timer will be reset to a new value. By selecting appropriate T value, the behavior of the timed evolution can be varied between migration ($T = 0$) and no migration ($T = \infty$).

4.2.1 Estimation of the Evolution Timer

As mentioned in the previous section, by varying the evolution timer the evolution process can be slowed down (by increasing the timer value), or hastened (by decreasing the timer value). Thus, selection of evolution timer forms a vital cog in the STEP mechanism. The reason for the tunability of evolution based on the evolution timer is due to the increase in the number of cores present in the system as the evolution becomes slower. Again, due to the increase in the number of cores in the system, the cost of the trees increases. Therefore, there exists a linear relationship between the cost of the multicast tree and the number of cores in the tree. In Section 4.4, simulation studies are carried out to justify this claim. In this subsection, the evolution timer is estimated based on the average number of cores in the system. In order to estimate the evolution timer, the following assumptions are made:

- Interval between successive core selection (core arrival interval) is exponentially distributed with a mean λ.
- Each node starts an evolution timer which is uniformly distributed in the interval $[0, T]$, and the node evolves as soon as the timer expires.
- A core gets deleted when all the nodes in that core evolve.

- Evolution process occurs instantaneously.

Infinitely large groups: In timer-based evolution, each node starts a uniform $[0, T]$ timer and evolves as soon as the timer expires. Therefore, a core in case of evolution gets deleted when all the nodes within the core evolve. This implies that a core gets deleted when the node having the maximum expiration time evolves. Hence, the core deletion time is distributed as the maximum of N uniform [0,T] distributions, where N is the number of nodes within a core. Such a distribution along with its mean and variance is derived in *Lemma A.1* (see Appendix A). For infinite value of N, deletion time becomes constant (variance = 0), and is equal to T (see *Corollary A.1* in Appendix A). When core deletion time becomes constant, tree evolution can be modeled as a Q_C queuing system mentioned in Appendix (see definition in Appendix A). Whenever a core arrival takes place, the number of cores in the system increases by one and the evolution timer T is reset. Therefore, core deletion occurs only if core arrival interval is less than T. At the end of interval T, all the cores get deleted except one because of the constant deletion time. The process is illustrated in Figure A.1 in the Appendix. Therefore, the average number of cores in evolution can be mapped to the average number of customers in a Q_C queuing system. From *Lemma A.2* (see Appendix A), the average number of cores (χ) is given by:

$$\chi = e^{\lambda T} \tag{4.2}$$

Solving Equation 4.2 for T, it follows

$$T = (1/\lambda) ln(\chi) \tag{4.3}$$

Finitely large groups: For groups with finite size, the number of members within a core in the multicast tree is also finite. Let N be the group size, and n be the average number of members in a core. Since each member starts an independent timer, n is equal to the mean number of successes in N independent Bernoulli trials, where the probability of successes is equal to the probability of a member moving into the new core. The mean number of successes in such an experiment in derived in *Lemma A.4* (see Appendix A). It is to be noted that $n \to \infty$ as

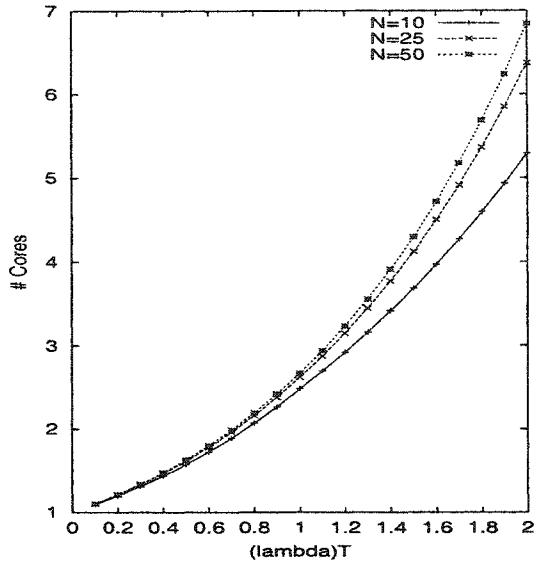


Figure 4.5 Variation of χ with λT

$N \rightarrow \infty$. Therefore, for higher value of N, Equation 4.2 can be applied with T being replaced by $\overline{T}$, where $\overline{T}$ is the mean value of T (see *Lemma A.1*). Therefore, the mean number of cores in the system (χ) becomes,

$$\chi = e^{\phi} \quad where \quad \phi = \frac{N(1-e^{-\lambda T})\lambda T}{N(1-e^{-\lambda T}) + \lambda T} \tag{4.4}$$

Variation of χ (i.e. #cores) with λT is plotted in Figure 4.5(b) according to Equation 4.4, for different values of N. As shown in the figure, the number of cores increases very rapidly compared to λT, therefore it is required to choose a lower value of T (i.e. evolution timer) so as to prevent having a large number of cores and thus high tree cost.

4.3 Comparison of Evolution versus Migration

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. The migration and evolution approaches are compared below:

- *Service Disruption:* Tree evolution is able to reduce service disruption experienced by

nodes in most cases. Service disruption can be categorized into mainly two types:(i) Service disruption due to the non-arrival of packets and (ii) Service disruption due to the packets failing to meet the QoS requirement. Since the new core joins to the existing multicast tree, all the packets that would have been lost in case of migration can be rerouted through the new core. This reduces service disruption of the first kind. Again, evolution incorporates QoS into the protocol unlike migration resulting in reduction of service disruption of the second type.

- *Group Dynamics:* The tree evolution model represents an excellent method for counteracting group dynamics in a multicast tree. The tree evolution acts as a buffer between the core-selection algorithm and the actual group members. By allowing members to stay with older cores for a limited time if their QoS is not violated, this inactive time buffers the node from the effect of the core selection. For even poor core selections, intra-group communication is relatively unaffected as the old tree will not change immediately. This is not the case in tree migration as all members would experience the effects of a poor core immediately. For the case of non-group senders, the effect is determined by the distance between the new core and the attachment point to the old tree.

- *Frequent core changes:* For highly dynamic networks where the core may switch frequently, the evolution model is ideal since it reduces the number of migrations required by a node. Consider an example where a core continually switches between two nodes. Under migration, all nodes would switch back and forth between each core as the core is continually migrated, thus resulting in potential "thrashing" of members among cores. However, under evolution, the number of nodes migrating may be dramatically decreased if the QoS of all members are being met thus removing unnecessary migrations.

- Tree Cost: Unlike migration, tree evolution may result in multiple cores to exist simultaneously. Hence, evolution has higher average tree cost than migration. In the simulation section (Section 4.4), the pros and cons of evolution are studied vis-a-vis migration and no-migration models.

4.4 Performance Studies

In order to evaluate the effectiveness of the tree evolution model, extensive simulation studies using NS-2 [42] were conducted. In the simulation studies, the comparison of the evolution model with migration as well as no-migration models was carried out. It is to be noted that the main objective of the simulation is to evaluate the relative performances of migration, evolution, and no-migration approaches, rather than to quantify the absolute performance offered by them. The various inputs for the simulation studies were generated as follows:

- Random network topologies were generated based on a given input parameter "graph density." This parameter determines the average node degree and hence the connectivity of the network. The higher the value, the denser the topology.
- The selections of source and receivers for a given multicast session were uniformly distributed from the node set.
- The initial and subsequent cores were selected randomly from a randomly selected from the group members and evaluated for QoS-constraints before being selected.
- An average of 10 random observations was considered as a single simulation point.
- *Default parameters:* (i) Total number of nodes = 100, (ii) Average Node degree = 4, (iii) Average number of groups = 10, (iv) Average link bandwidth = $15Mbps$, (v) Average link delay = $12.5ms$ (vi) Member join/leave inter-arrival time = $250ms$, (vii) Core change interval = $250ms$, (viii) Average QoS (delay) requirement of the nodes = $60ms$, (ix) Evolution Timer = $150ms$.

In order to compare the effectiveness of the various models, we evaluated the models according to the following performance metrics:

Service disruption: The percentage of packets that were either lost (dropped) or had violated QoS-constraints were monitored on a per-receiver basis and averaged. Service disruption due to QoS violation (referred to as QoS loss in the performance plots) was also measured along

with service disruption due to the non-arrival of packets (referred to as Packet Loss in the performance plots). Let the total number of receivers be n. Let a receiver i receive r_i packets out of total s_i packets it is supposed to receive. Let q_i be the number of packets violating QoS among r_i. Then, the total packet loss is defined as the percentage of packets lost due to non-arrival of packets and packets lost due to QoS violation. Mathematically, it follows

$$Total\ Packet\ Loss(\%) = \frac{\sum_1^n s_i - \sum_1^n r_i + \sum_1^n q_i}{\sum_1^n s_i} \times 100 \tag{4.5}$$

Packet Loss is defined as the percentage of packets lost due to non-arrival of packets. Mathematically, it follows

$$Packet\ Loss(\%) = \frac{\sum_1^n s_i - \sum_1^n r_i}{\sum_1^n s_i} \times 100 \tag{4.6}$$

QoS loss is defined as the percentage of packets lost due to the QoS violation among the packets arriving at the receivers. Therefore,

$$QoS\ loss(\%) = \frac{\sum_1^n q_i}{\sum_1^n r_i} \times 100 \tag{4.7}$$

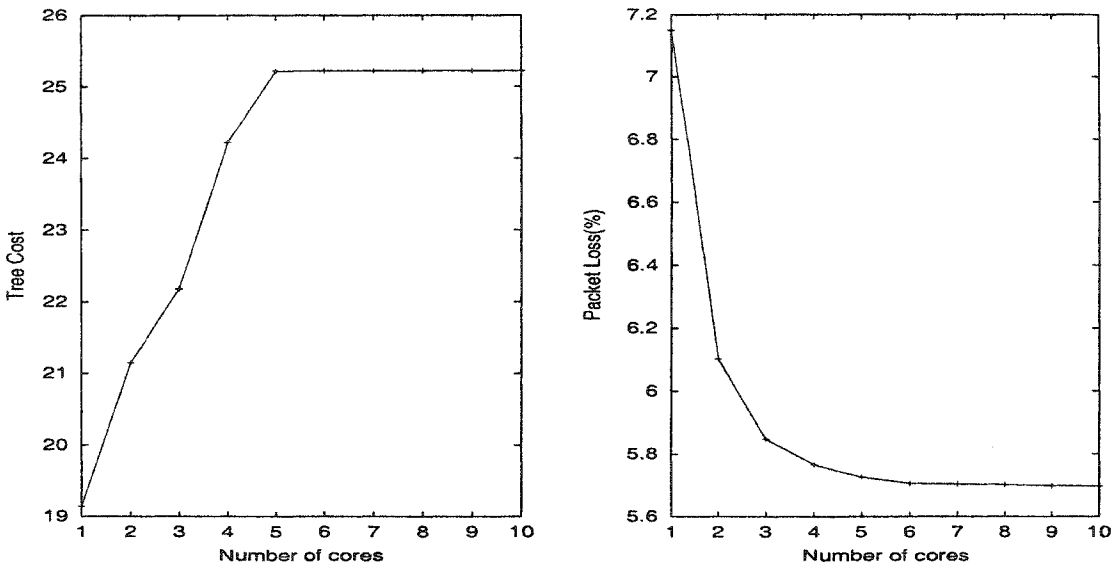
Tree cost: The average cost of multicast tree averaged out over time were compared to evaluate the effectiveness of the algorithms. Let the total simulation time be divided into n equal intervals. Let the cost of the multicast tree in the time interval $[j-1, j]$ be C_j. Then the tree cost is calculated as

$$Average\ Tree\ Cost\ =\ \frac{1}{n}\sum_{j=1}^{n} C_j \tag{4.8}$$

4.4.1 Simulation Experiments

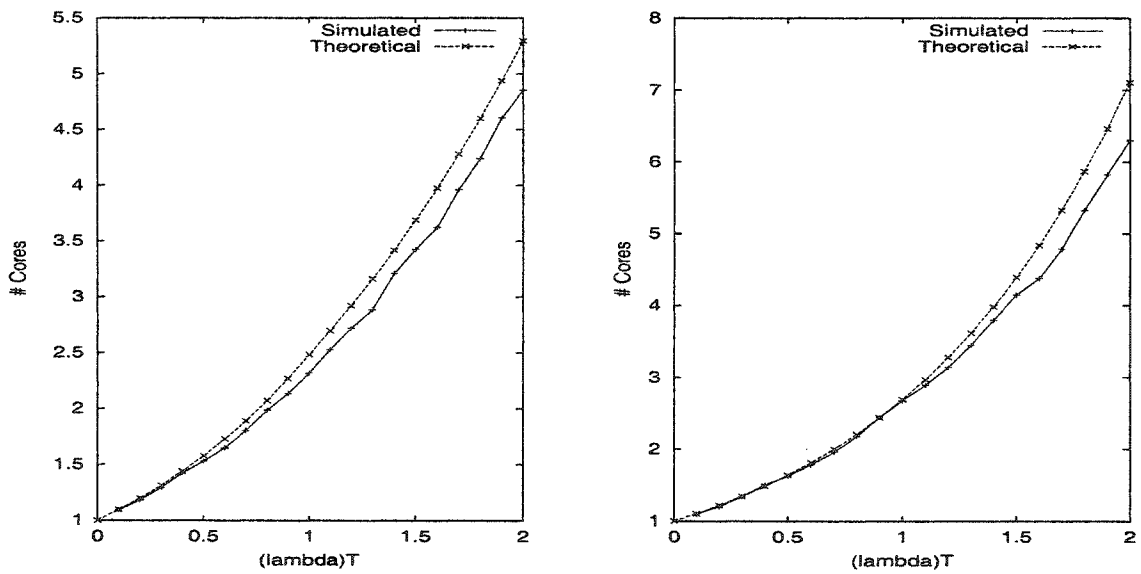
In order to adequately capture the differences between the evolution and migration algorithms, the effects of the following parameters were studied:

- *Estimation of Evolution Timer:* The evolution model was studied varying the number of cores. This experiment was performed to select a default core value, which was used to estimate the Evolution Timer.
- *Effect of Group Dynamics:* The rate of members joining and leaving a multicast group was varied keeping all the other parameters fixed at their default values.



(a) (b)

Figure 4.6 Effect of number of cores on (a) tree cost and (b) service disruption



(a) (b)

Figure 4.7 Variation of #cores with λT in a (a) 10 node and (b) 100 node network

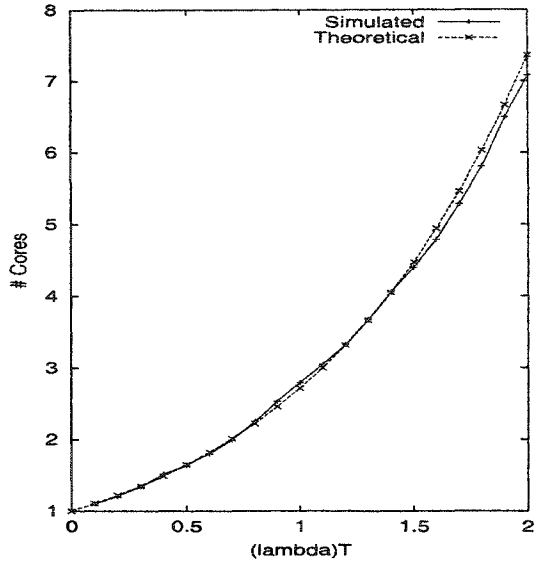


Figure 4.8 Variation of #cores with λT in a 1000 node network

- *Effect of Frequency of Core Selection:* The core migration rate was varied.
- *Effect of Graph Density:* Graphs can be made dense or sparse by changing the node degree. Effect of the node degree was studied keeping all the parameters fixed.
- *Effect of receiver QoS Requirement:* The effect of QoS requirement of the receivers was studied keeping all other parameters fixed.
- *Effect of Group Size:* The number of receivers in the group were varied.

4.4.2 Variation of number of cores and selection of Evolution Timer

Figure 4.6(a) shows the variation of tree cost under evolution model with number of successive cores that are allowed in the tree. There is an increase in tree cost as the number of cores increases. For number of cores ≥ 5, the cost remains more or less constant. When the number of cores is equal to 1, evolution performs like migration. Hence, the cost is the lowest. As the number of cores increases, the possibility of several cores existing at the same time increases, resulting in higher tree cost. It is to be noted that when the number of cores increases beyond 6, the cost of the multicast tree remains more or less constant. This phenomenon occurs be-

cause in this case the members remain spread out in different cores and hardly move to the new core, resulting in a constant cost. However, the cost of the tree in these cases are very high.

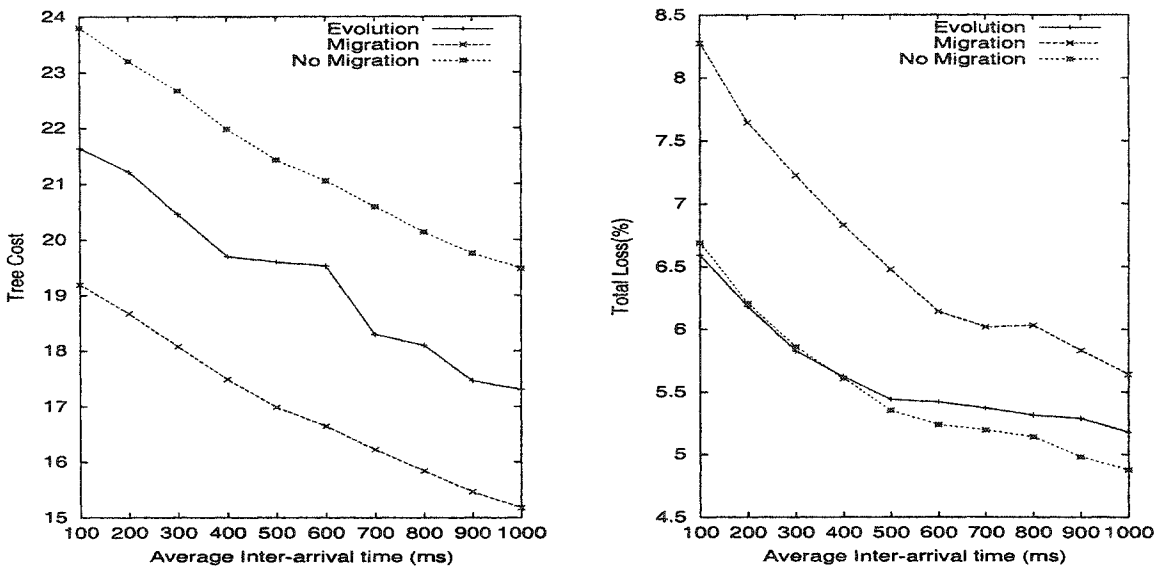
Figure 4.6(b) shows the variation of total packet loss under evolution with number of cores. The total packet loss decreases with increase in the number of cores. When the number of cores goes above 3, the packet loss remains more or less constant. This results help in choosing the value for the number of cores existing at the same time. Core value of 2 represents a balanced trade-off between tree cost and packet loss, and hence used subsequently.

These two graphs depict the trade-off between tree cost and service disruptions. From the graph it is evident that migration ($\#cores = 1$) results in better tree cost at the expense of service disruption. On the other hand, extremely slow evolution ($\#cores \geq 5$) is good in terms of service disruption but very bad in terms of tree cost. The evolution model essentially captures this trade-off.

In another set of experiments, the theoretical model of evolution is validated. After validation, Equation 4.4 is used to select the the value of evolution timer. As shown in the Figure 4.7 and 4.8, average number of cores are plotted against λT. The figure shows a comparison between the theoretical and simulation results for group size of 10, 100 and 1000. It is to be noted that, as the group size increases the theoretical and experimental results match closely, as has been predicted before. As has been argued before a system which restricts the number of cores to 2, provides a trade-off between service disruption and tree cost. Therefore, substituting the average number of cores (χ) as 2, $N = 10$, and $1/\lambda = 250ms$, in Equation 4.4, $T = 150ms$ is obtained. Therefore, node timer is selected as 150ms, which provides a balance between service disruption and tree cost.

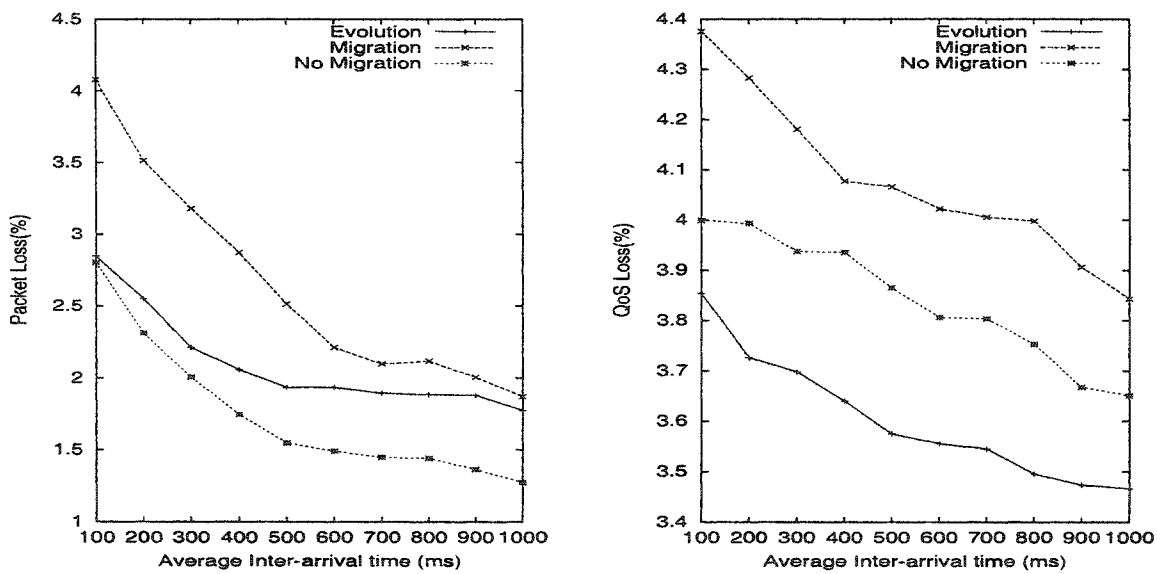
4.4.3 Effect of Member Join/Leave Inter-Arrival time

In Figure 4.9(a), tree cost is plotted against member join/leave inter-arrival time. Higher the inter-arrival time between the receivers, the groups become less dynamic. As the group dynamics increase, the activity within the group increases resulting in increase of tree cost.



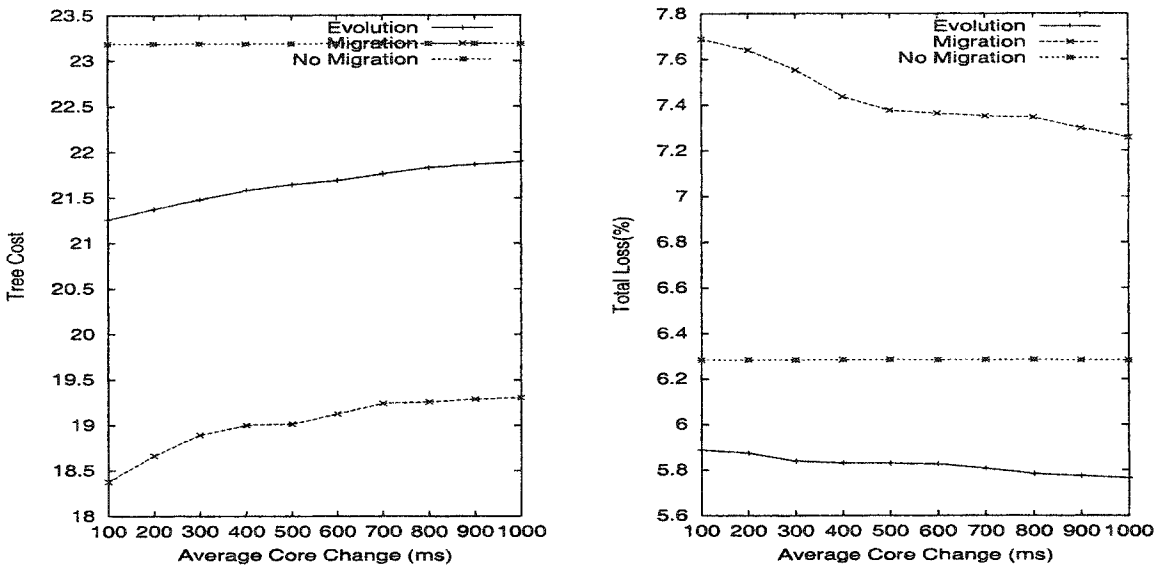
(a) (b)

Figure 4.9 Effect of inter-arrival time on (a) tree cost and (b) total packet loss



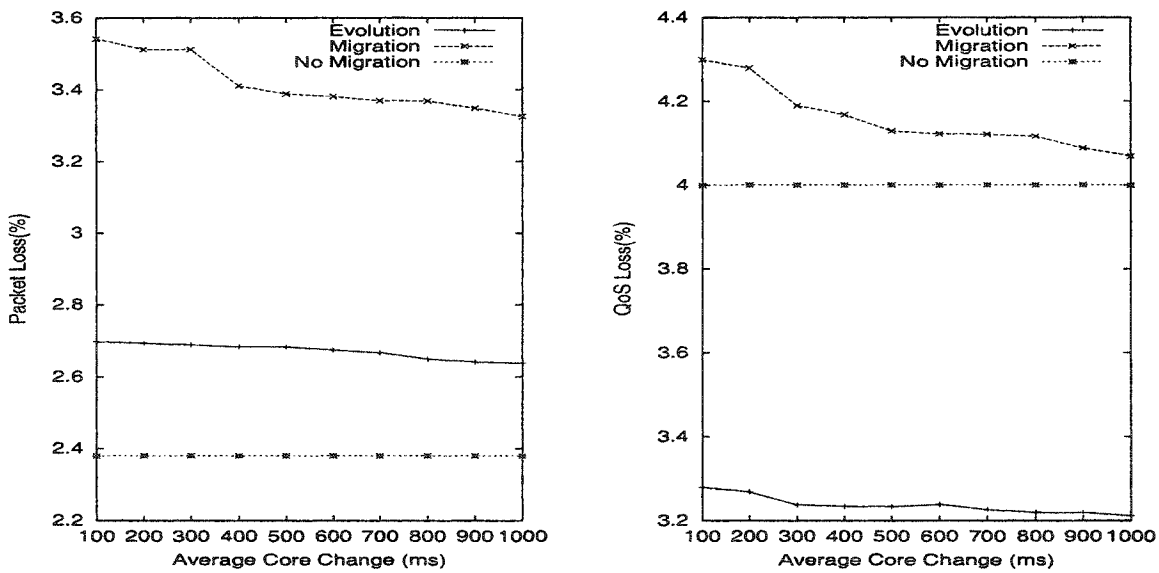
(a) (b)

Figure 4.10 Effect of inter-arrival time on (a) packet loss and (b) QoS loss



(a) (b)

Figure 4.11 Effect of core change time on (a) tree cost and (b) total packet loss



(a) (b)

Figure 4.12 Effect of core change time on (a) packet loss and (b) QoS loss

Comparing between the three models, it can be inferred that migration has the best tree cost and no migration has the worst, while trees created under evolution lie in between. The main motivation for migration and evolution is to reduce the tree cost by migrating/evolving to a new core. From the figure, it is evident that both the models are able to achieve this.

In the Figure 4.9(b), the total packet loss is plotted against inter-arrival time. Figures 4.10(a) and (b) show the packet loss due to non-arrival of packets and due to QoS violation, respectively. From the Figure 4.10(a) it is clear that the evolution model not only has the least loss, but evolution also has the highest tolerance to group dynamics. This point is reiterated by Figure 4.9(b). With increase in group dynamics, loss due to non-arrival of packets increases slower than the other two models. This justifies the claim that evolution is much less sensitive to group dynamics than others. Figure 4.10 (b) shows that, evolution has the least QoS loss as evolution takes QoS into consideration.

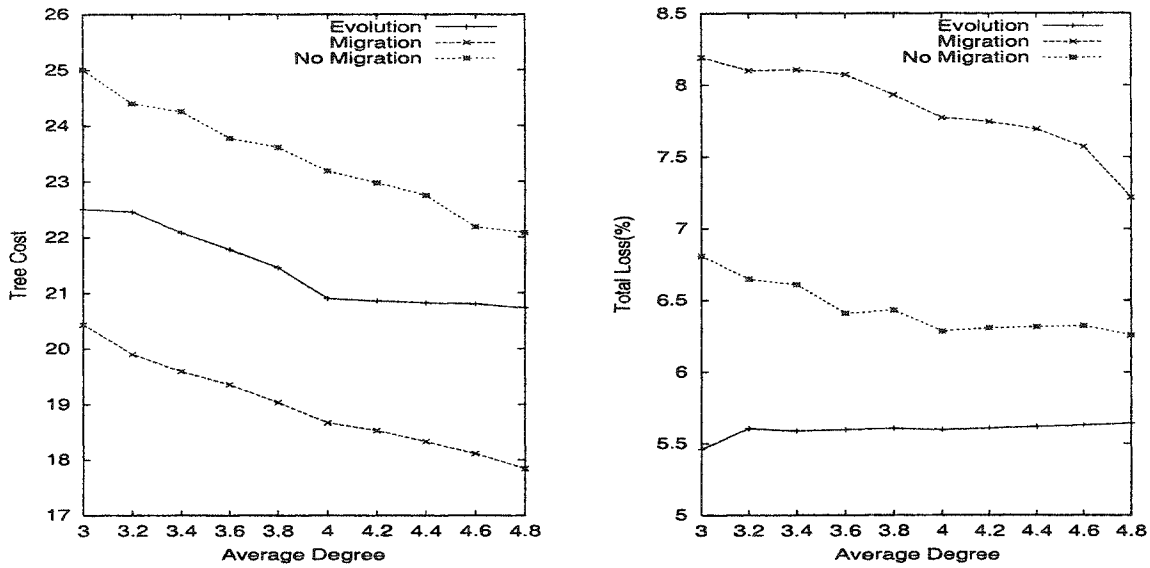
4.4.4 Effect of Core Change Frequency

In Figure 4.11(a), tree cost is plotted against core change frequency. With increase in core change frequency (*i.e.* decrease in core change time), the tree cost decreases. This shows that core change really helps in reducing the cost of the tree. Tree cost in case of migration is the least and tree cost in case of evolution is in between no migration and migration.

In Figure 4.11(b), the total packet loss is plotted against core change time. The total packet loss in case of evolution is much lower than that in case of no migration and migration. Also, evolution is less sensitive to core changes than migration. The total packet loss is broken down and plotted in Figures 4.12(a),(b).

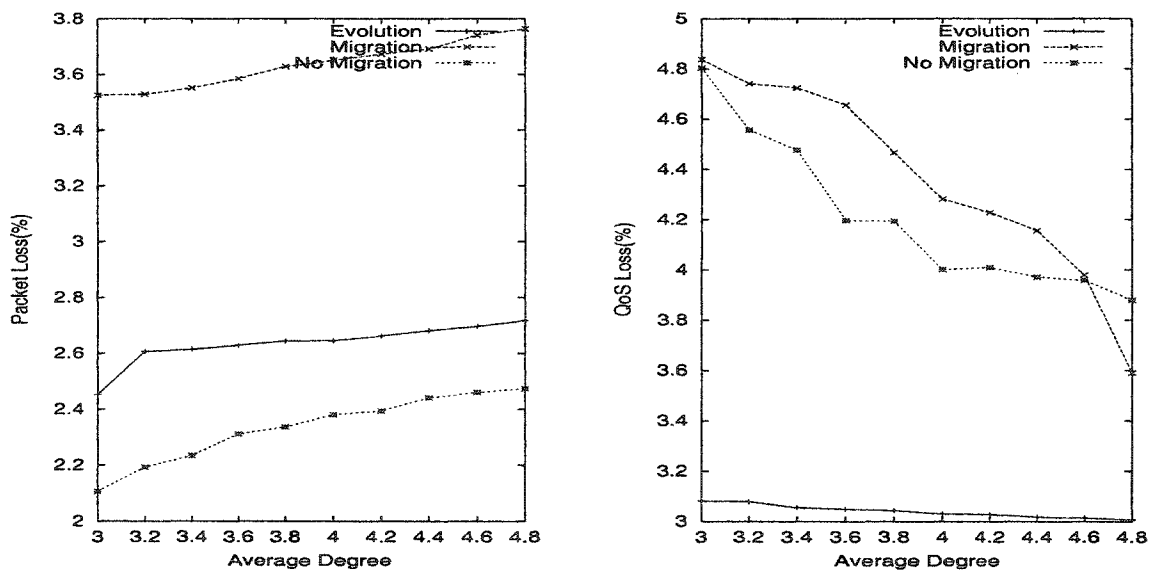
4.4.5 Effect of Node Degree

In Figure 4.13(a), tree cost is plotted against average node degree. With increase in node degree, tree cost decreases. This is because with increase in node degree, network becomes more dense resulting in better paths for the receivers to join the group, which in turn results in lower tree cost.



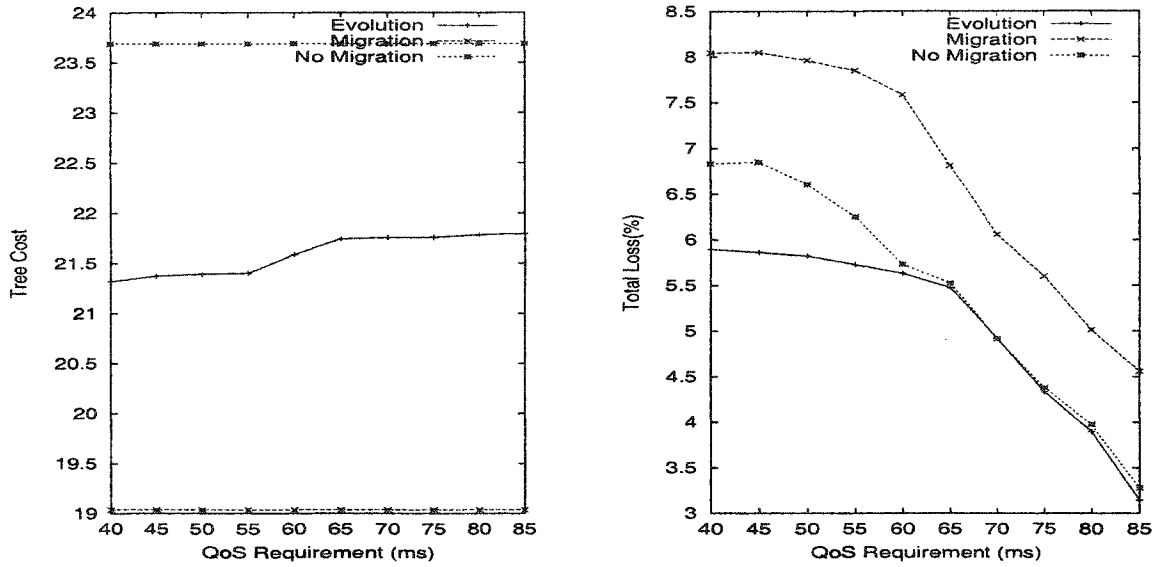
(a) (b)

Figure 4.13 Effect of node degree on (a) tree cost and (b) total packet loss



(a) (b)

Figure 4.14 Effect of node degree on (a) packet loss and (b) QoS loss



(a) (b)

Figure 4.15 Effect of QoS requirement on (a) tree cost and (b) total packet loss

The variation of total packet loss with node degree is plotted in Figure 4.13(b). The total packet loss decreases for migration and no migration but remains more or less constant for evolution. This is because as the node degree increases the possibility of two trees sharing common links reduces. Since evolution takes the advantage of tree sharing so it performs significantly better in sparse network. In Figures 4.14(a) and (b), the packet loss and QoS loss are respectively plotted with average node degree.

4.4.6 Effect of QoS Requirement

In the Figure 4.15(a), the tree cost is plotted against QoS requirements of the receivers. While migration and no migration models are indifferent to QoS changes, evolution cost increases as QoS requirements are relaxed. This is because for stringent QoS requirements, most of the receivers face QoS violations and evolve, so the resultant tree cost is similar to migration. For relaxed QoS requirement, less number of receivers evolve resulting in cost of the tree similar to no migration model.

In Figure 4.15(b), the total packet loss is plotted against QoS requirement. The packet

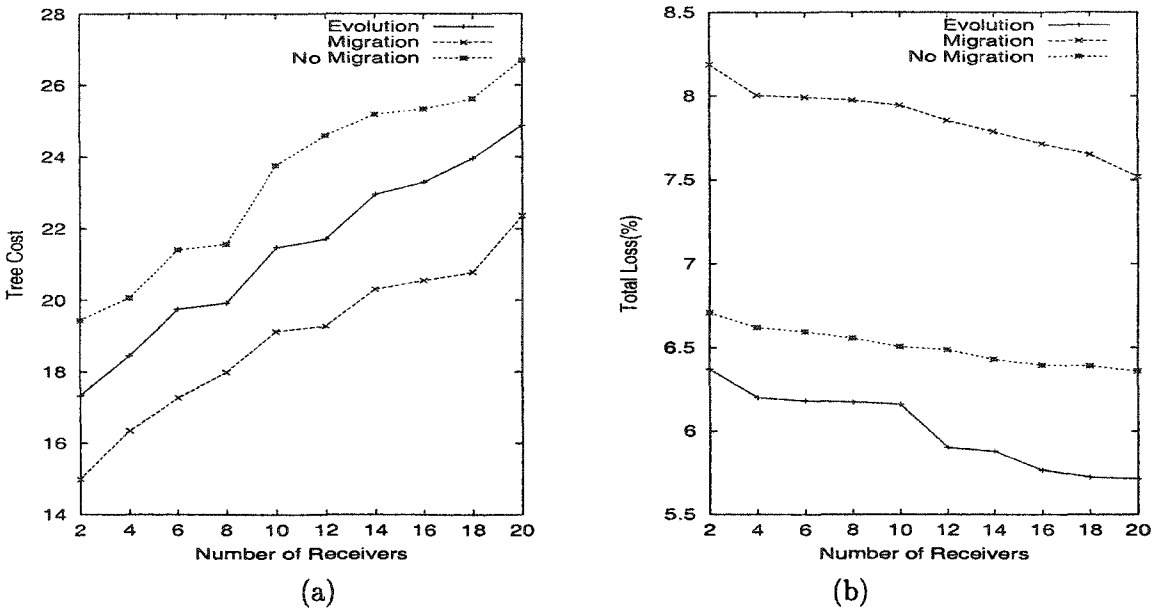


Figure 4.16 Effect of # receivers on (a) tree cost and (b) total packet loss

loss decreases as the QoS requirements of receivers becomes less stringent. This is because, as the QoS requirements are relaxed the receivers can tolerate worse paths to the multicast tree resulting in less packet loss. This graph also shows that for stringent QoS requirement ($\leq 60ms$) evolution model performs significantly better. It is to be noted that at very stringent QoS requirement ($\leq 50ms$), the packet loss is around $6 - 8\%$. This may or may not be acceptable depending on the application supported. Therefore, applications may choose QoS requirement based on the packet loss acceptable by the application.

4.4.7 Effect of number of receivers

In Figure 4.16(a), tree cost is plotted against number of receivers. As the number of receivers increases tree cost also increases. Again, migration has the least cost while evolution has slightly higher cost with no migration having the highest cost.

In Figure 4.16(b), total packet loss is plotted against the number of receivers. As the number of receivers increases, the total packet loss decreases as the tree becomes more and more dense. Evolution has the least service disruption among all the models.

4.5 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 [43], a global tree maintenance technique called *tree evolution* was proposed which provides a balance between service disruption and tree cost. The integrated framework developed in this dissertation, switches between tree migration and tree evolution depending on number of cores the system can support.

An integrated approach (shown in Figure 4.17) is proposed 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, this 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.

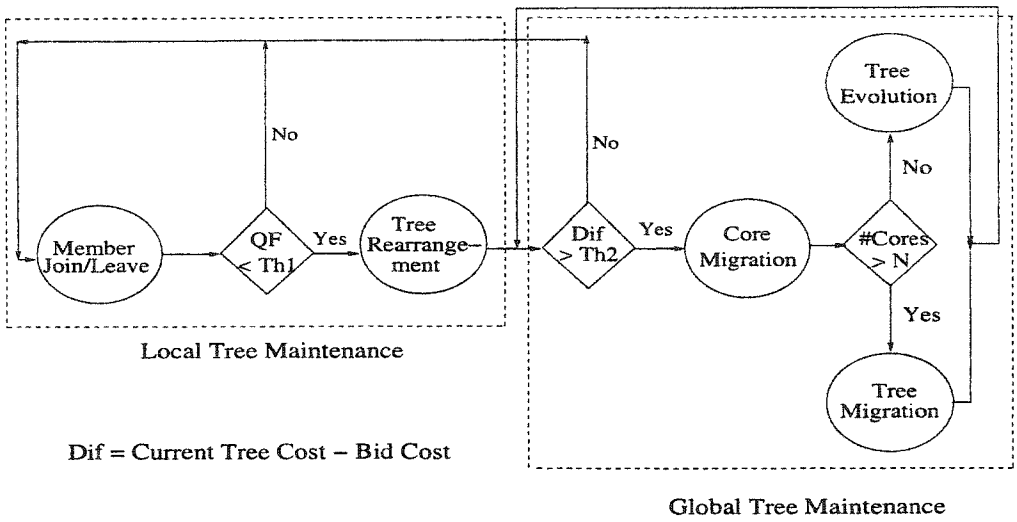


Figure 4.17 Integrated tree maintenance approach

- 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.

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, CRCDM protocol is considered, which is described in [27]. 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 a small 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 4.17, 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 [43] to justify the choice of N. As an example, let c_1, c_2 ... 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.

4.6 Summary

In this chapter, the case for tree evolution for managing group dynamics for QoS multicasting has been advocated. Tree evolution provides a mechanism for gradually evolving from one multicast tree to another over a period of time. The STEP protocol, based on the evolution model has also being proposed in this chapter. STEP strikes a balance between service disruption and tree cost and can be tuned between migration (low tree cost and high service disruption) and no migration (high tree cost and low service disruption) by changing the period. To quantify the cost and service disruption trade-off an analytical estimate of the evolution timer (which determines the number of cores in the multicast group) has been derived, and also extensive simulation studies has been carried out. The studies show that:

- *Service Disruption:* Evolution has much less service disruption than migration. Difference between them increases as the group becomes more dynamic and the network becomes sparser.

- *Tree Cost:* Trees produced under evolution model have higher cost than that produced

by migration, but has lesser cost than trees produced when no migration took place. Under stringent QoS requirements when nearly all the members evolve, trees produced by evolution have similar costs to that of migration. If the QoS requirements are relaxed, tree cost under evolution model nears to that of the no migration case. Thus evolution is able to achieve a balance between tree cost and service disruption.

- *Buffering from Frequent Core Changes:* Evolution is much less sensitive to frequent core changes than migration. This buffers evolution from bad core selection and possible "thrashing".

In this chapter, tree evolution, tree migration and tree rearrangement techniques are combined together into an integrated tree maintenance approach.

CHAPTER 5 Reliability Constrained Multicast Routing

The problem of handling network dynamics in the context of multimedia applications is an important issue due to the importance of the data involved and optimality considerations. An important aspect of network dynamics is failure handling, as link/node failure introduces service disruption. For unicasting, one type of failure handling approach is the *protection based* approach [44, 45, 46] wherein dedicated protection mechanisms, such as redundant (backup) channels operating in hot standby, are employed to cope with failures. The primary and backup channels are usually node and link disjoint. This approach is more suitable for hard real-time communication wherein every packet is critical. The other type is the *restoration based* approach [47] wherein a dedicated mechanism is used to detect node and link failures. On detecting a failure, attempts are made to reroute (restore) the channel around the faulty nodes/links with minimal service disruption. This approach is useful if occasional packet losses are tolerable (such as in multimedia applications), and restoration is initiated only when a permanent failure is detected. Creation of primary and backup paths have been studied in the context of ATM networks [47], real-time networks[44, 45, 46], and optical networks [48, 49]. With multicasting, the problem is much more complicated than with unicasting, as resource reservations are shared and group dynamics interact with network reconfigurations. Very little is known as to how to deal with such problems [8].

In this chapter, a protection based approach is proposed wherein protection is provided in a resource efficient manner. While joining the group, each receiver specifies its reliability requirement as a QoS parameter. The network is assumed to be divided into several domains and the network tries to satisfy the reliability requirement by providing backup paths within a selected set of domains, if necessary. If the reliability requirement is not satisfied even with

backup paths, then the join request is rejected. The problem tackled in this chapter falls under the area of inter-domain multi-constrained routing [50], where multiple constraints such as delay, bandwidth, jitter, and reliability need to be satisfied, with the objective of optimizing metrics such as path cost. In literature, constrained multicast routing problem has been studied in detail for centralized and distributed settings. One way to approach the QoS inter-domain routing problem is through QoS partitioning. Several optimal and heuristic algorithms for QoS partitioning have been studied in [51, 52, 53]. In these works, the authors assume that all nodes in the network have full topology and cost information, and then apply approximation algorithms to realize QoS partitioning. The problem of multi-constrained routing assuming full topology information has also been addressed in the literature, by proposing optimal and heuristic solutions for several instances of the problem [54, 55, 56]. Under distributed settings, several solutions to the QoS routing problem for unicast and multicast communications have been proposed [28, 29, 30, 57].

The main distinguishing feature of the work presented in this chapter from the multi-constrained QoS routing solutions (mentioned above) is that, this work treats reliability as a QoS parameter, and exploits the "parallel" path property of reliability in the QoS multicast routing algorithms developed in this chapter. The "parallel" path property (primary-backup paths) allows backup paths to be created for a primary path to increase the reliability of the combined path. In a multi-constrained QoS routing with reliability constraint, the creation of backup path(s) would increase the chances of a connection getting accepted which would otherwise have been rejected. Efficient algorithms for establishing real-time connections with reliability constraint have been studied recently in [58]. These algorithms, though very useful, can be used only in an intra-domain scenario with each node having full topology information, i.e., these are centralized intra-domain algorithms. In this chapter, distributed algorithms for the reliability and QoS constrained multicast routing problem are developed, assuming that the nodes have partial or imprecise topology information for intra-domain routing, and the border routers have only the hop count information for inter-domain routing. Therefore, the algorithms proposed in this chapter are based on a more realistic model of Internet routing,

and are clearly different from what is known in the literature.

In this chapter, firstly the Reliability Constrained least Cost Dynamic Multicast Routing (RCLCR) problem is described, and then the solution approach to the problem is discussed.

5.0.1 Internetwork Model and Assumptions

In this subsection, the different assumptions made throughout the chapter is listed.

- The network is composed of domains, where the nodes at the edge of the domains are called *edge routers* or *border routers.*
- Each edge router maintains intra-domain information as well as inter-domain information. The nodes inside a domain other than the edge routers maintain only the intra-domain information.
- An underlying distance vector protocol is assumed, by which nodes within a domain exchange information.
- Each link in the network has a non-zero cost associated with it. Cost of a link could be a function of bandwidth and/or delay.
- It is assumed that the reliability of the links are known, where the reliability of a link could be a function of the following: (i) the physical reliability of the link (e.g., fiber vs. copper), (ii) type of medium (e.g., wireline vs. wireless), (iii) the packet loss probability due to long-term congestion. It is recognized that computing link reliability based on these parameters is a research problem by itself and is beyond the scope of this dissertation.
- Each node maintains two tables as an intra-domain information: Primary Path Table and a Backup Path Table which are used to construct primary and backup paths respectively.
- In addition to the two intra-domain tables mentioned above, edge routers also maintain the inter-domain table. For a given domain D_i, the inter-domain table maintained at the edge router Y in domain D_s, contains the number of domains traversed in the shortest path between D_s and D_i. The inter-domain tables can be constructed by the edge routers

through exchange of path vector information. The idea is similar to BGP speakers exchanging information in case of Border Gateway Protocols. Reference [59] has a detailed discussion on creation of inter-domain distance table in hierarchical network settings.

5.1 Problem Definition and Solution Approach

In this section, firstly the Reliability Constrained Least Cost Dynamic Multicast Routing (RCLCR) problem is defined, and then the solution approach for the problem is discussed.

5.1.1 Problem Definition

A multicast join request is modeled in a network $N = (V, E)$ as a 4-tuple: $R =< S, G, B, r_r >$, where $S \in V$ is the receiver joining the multicast group G. B and r_r are respectively the bandwidth and reliability requirements of the receiver S. Let C be the core of the multicast group. RCLCR problem involves creating a path $P = (V_P, E_P)$ between S and C such that:

- $V_P \subseteq V$ and $E_P \subseteq E$
- $S \in V_P$
- $Rel_P(S,\ C) \geq r_r$ where $Rel_P(S,\ C)$ is the reliability of the join path between S and C.
- $AB(e) \geq B,\ \forall\ e \in E_P$, where $AB(e)$ is the available bandwidth in link e.
- Path P has the minimum cost among all the *feasible paths* between S and C.

In other words, RCLCR problem can be stated as: *Creation of path P between S and C, such that reliability and bandwidth constraints are satisfied and the cost of the path is minimized.* It has been proved that minimum cost routing having two path constraints is NP-Complete [3]. A well-known example of such problems is delay constrained least cost routing problem. Similar to the delay constrained least cost routing problem, RCLCR can also be proved to be NP-Complete (See Lemma B.1 in Appendix B).

5.1.2 Solution Approach

Since RCLCR problem is NP-Complete, development of a systematic heuristic approach is required which involves creation of end-to-end primary paths, and creation of backup paths for selected domains, if required. Solution to the RCLCR problem involves creation of backup paths in some domains such that the reliability requirement of the receiver is satisfied and the overall cost is minimized. Since backups are provided partially, this approach is known as Partial Protection Approach (PPA). PPA has two steps:

1. Primary path creation which minimizes cost and satisfies the reliability and bandwidth requirements. If such a path does not exist, then step 2 is carried out.

2. Creation of backup paths in selected domains to satisfy the reliability and bandwidth requirement and to minimize the overall cost. If the requirements are still not satisfied the request is rejected. The following illustration explains the proposed solution approach.

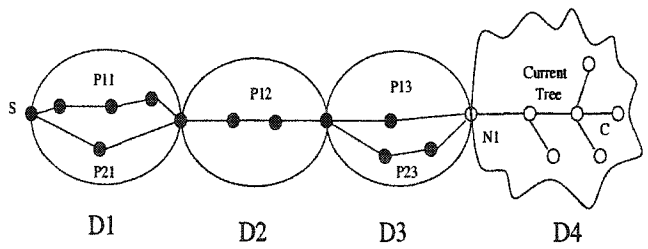


Figure 5.1 Illustration of inter-domain receiver join process

Example: In Figure 5.1, let $R =< S, G, B, r_r >$ be the join request. D_1, D_2, D_3 and D_4 are the domains in the join path. Let P_{11}, P_{12} and P_{13} be the primary paths in the domains D_1, D_2 and D_3 respectively. Let C be the core of the multicast group G. The current multicast tree exists solely in Domain D_4. Using the Partial Protection Approach, by providing backups for the primary paths (P_{21} and P_{23}) in domains D_1 and D_3, the reliability constraint of the receiver (S) is satisfied.

The basic idea used in PPA is as follows: *The reliability of the combined path (parallel combination) is more than the reliability of each of the individual paths.* For example, let the

reliability of the primary and backup paths be 0.9 each. Then the reliability of the combined path is 0.99. The PPA approach is formally defined as follows: Let r_{ij} be the reliability of the path P_{ij} i.e., i^{th} path in the j^{th} domain. *Weight*, w_i, of the path P_{ij} in a domain D_i in the join path is defined as the ratio of the reliability of the combined path (primary and backup together) to that of the primary path in the domain D_i and is given by

$$w_i = 1 + r_{2i}(\frac{1}{r_{1i}} - 1) \tag{5.1}$$

The total reliability of the end-to-end path is given as:

$$Rel = r_{11} \times r_{12} \times r_{13} \times w_1 \times w_2 \times w_3 \times (Rel_{Tree}) \tag{5.2}$$

In Equation 5.2, Rel_{Tree} is defined as the reliability of the path from the on-tree node (N_1) to the core (C) of the multicast tree. Note that $w_2 = 1$, since there is no backup path in domain D_2.

To generalize the above example, let D_1, D_2 ... D_n be n domains in the join path. Then the total reliability of the end-to-end path is:

$$Rel = (Rel_{Tree}) \times \prod_{i=1}^{n}(r_i \times w_i) \tag{5.3}$$

5.1.3 Schemes to implement Partial Protection

PPA can be implemented using single pass [60] or two-pass resource [61] reservation mechanism. In this chapter, for the implementation of PPA, the two-pass scheme is adopted as it is resource-efficient. In the two-pass scheme, resources are allocated in the forward pass and excess resources are relaxed in the reverse pass so that the resources are efficiently utilized. Three variations of two-pass schemes are proposed:

- *Conservative Scheme:* In this scheme, in the forward pass, both primary and backup paths are created in all domains along the path. In the reverse pass, backup paths in selected domains are relaxed.

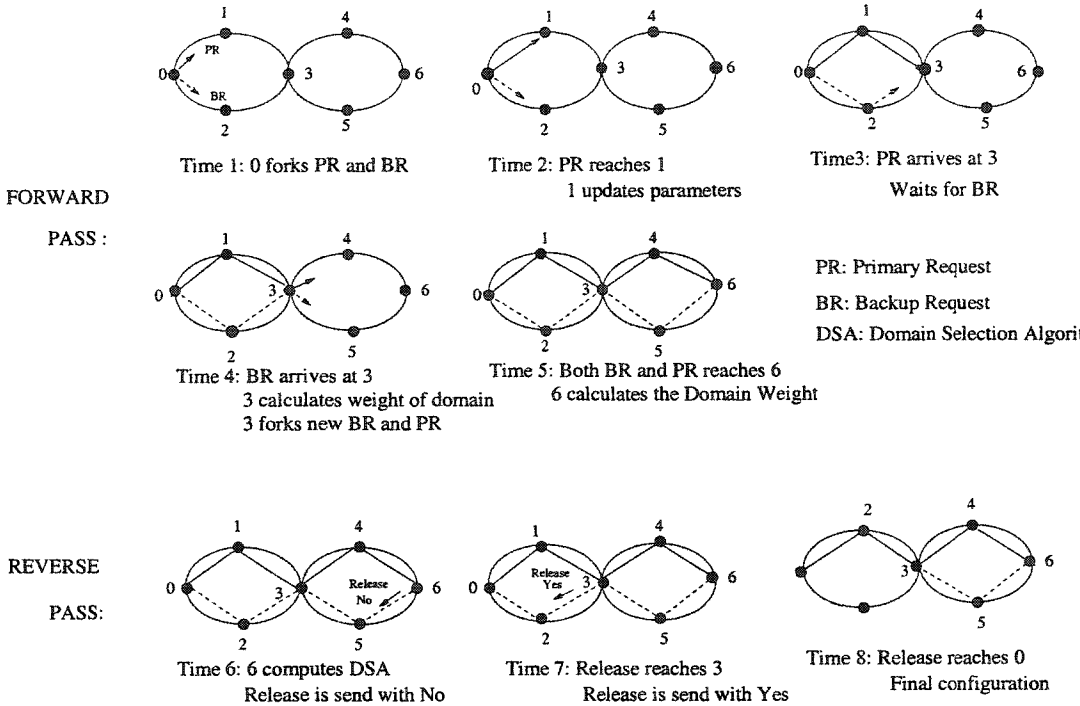


Figure 5.2 Illustration of the conservative protocol

- *Optimistic Scheme:* In this scheme, in the forward pass only the primary paths are created. Backup paths are created in the reverse pass, if necessary.

- *Hybrid Scheme:* This scheme combines the benefits of both Conservative Scheme and Optimistic Scheme. In the forward pass, backup paths are created based on a prediction of the domain reliability. In the reverse pass backup paths may be created or relaxed.

5.2 Conservative Partial Protection Scheme

In this scheme, during the forward pass both primary and backup paths are created and reserved for all domains in the end-to-end path. In the reverse pass, backup paths in some domains are released so that the reliability criterion remains satisfied and the cost is minimized. In the following sub-sections the forward and the reverse pass mechanisms of the scheme are described.

Figure 5.2 gives an overview of the Conservative Scheme. In the figure, let node 0 be the

source and node 6 be the core of the multicast tree. There are two domains and each domain has 4 nodes. The scheme is described at different instances of time. Detailed explanations follow in the next few subsections.

5.2.1 Forward Pass

The different stages of forward pass under Conservative Scheme is shown in Figure 5.2. A receiver willing to join a multicast group G (node 0 in Figure 5.2) forks a Primary Request (PR) and a Backup Request (BR) with destination as the edge router closest to the Core of the multicast group. At the egress router (node 3 in the Figure 5.2), weight of the path in the domain is calculated after both PR and BR arrives. The egress router (which is the ingress router to another domain) again forks PR and BR and the process continues. In the description of the scheme, whenever it is mentioned that a join request has reached an edge router, it is meant that both PR and BR have reached the edge router. Finally, the domain containing the Core node forks PR and BR with destination set as the Core node. If in any domain, either PR or BR reaches an on-tree node they are forwarded till the egress router of the domain or Core. The final node of the forward pass is known as the "Parting Node".

5.2.1.1 Primary Path Creation

The Primary Path Creation Problem involves creation of a feasible path between ingress and egress routers of a domain, so that the cost of the path is minimized and reliability and bandwidth constraints are satisfied. Since the problem is NP-Complete, two reliability-based heuristics are proposed to solve the problem. For the description of the heuristics, the following notations will be used:

- $MRel(x, d)$ = Reliability of the maximum reliable path from node x to d.
- $LCost(x, d)$ = Cost of the least cost path from node x to d.

Residual Reliability Maximization (RRM) Heuristic: The main intuition behind the RRM heuristic is to choose a path which simultaneously maximizes residual reliability and minimizes cost.

Let a join request $R =< S, G, B, r_r >$ reach the node X in domain D. Domain Reliability Parameter (DRP), r_d, is defined as the reliability required by the path in domain D, so that r_r is satisfied. DRP is calculated at the edge router and stored in the join packet. The calculation of DRP will be discussed later in this subsection. Let L_1, L_2 ... L_n be the links incident on node X. A link L_i is identified by the tuple $< X, Y_i >$, where Y_i is the node adjacent to node X along the link L_i. Let r be the reliability of the path till node X traversed by the join request packet. Let $RRM(L_i, d)$ be the value of the heuristic for the join request R along link L_i towards destination d. It is defined,

$$RRM(L_i, d) = \frac{LCost(X, Y_i)}{\rho_r} \tag{5.4}$$

$$where \quad \rho_r = \log(\frac{r \times MRel(X, Y_i) \times MRel(Y_i, d)}{r_d})$$

ρ_r in Equation 5.4 is defined as the *residual reliability*. Residual reliability contains logarithmic function, to convert reliability which is multiplicative in nature to additive form. A positive residual reliability indicates that the reliability criterion is satisfied for the current domain if the link is chosen. It can be seen that RRM tries to choose a path by simultaneously maximizing residual reliability and minimizing cost. Among all the links having positive RRM values the link having the least RRM value is selected. If none of the links has positive RRM values then the link having the minimum RRM value is selected. At each node, loop is taken care of by removing the links, which connect the current node to the already traversed nodes, from the list of possible outgoing links.

Cost Inverse Reliability Product (CIRP) Heuristic: The main intuition behind the CIRP heuristic is to choose a path which simultaneously minimizes the end-to-end cost and maximizes the end-to-end reliability. At node X, $CIRP(X, d)$ denotes the value of the heuristic between node X and node d.

$$CIRP(X, d) = LCost(X, d) \times \log(\frac{1}{MRel(X, d)}) \tag{5.5}$$

The main difference between RRM and CIRP heuristics is that, RRM heuristic tries to produce the least cost path satisfying the reliability requirement, while CIRP tries to optimize both cost and reliability without explicitly satisfying the reliability requirement.

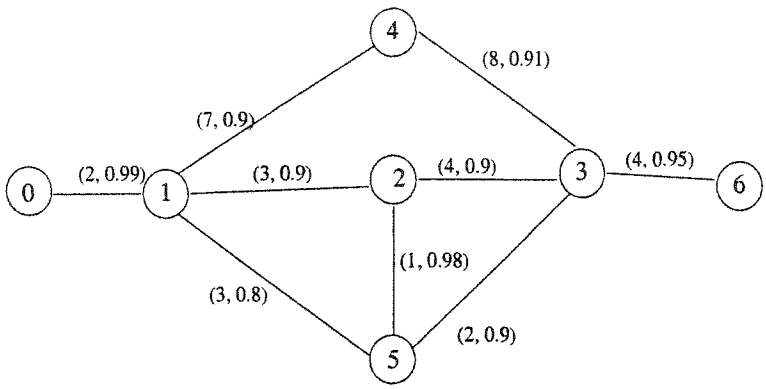


Figure 5.3 Example of primary path creation algorithms

Example of Path Creation under different Heuristics:

In Figure 5.3, the creation of paths between node 0 and node 6 is illustrated. Each link is shown as a tuple $< x, y >$, where x indicates the cost of the link and y indicates the reliability of the link. It is assumed that the DRP for this join request in the given domain is 0.75. The primary paths are now created using different heuristics mentioned above:

- *Most Reliable Path:* $0-1-4-3-6$ is the most reliable path between Node 0 and 6.
- *Least Cost Path:* $0-1-5-3-6$ is the least cost path between Node 0 and 6.
- *Least CIRP Path:* To compute the least CIRP path the CIRP values are computed for all the possible paths between 0 and 6. The least CIRP path is $0-1-2-5-3-6$.
- *Least RRM Path:* At node 0, join request is forwarded to node 1. Among all the links incident on 1, only link $(1-2)$ has a positive value and also does not create a loop. Therefore, the join request is forwarded to 2. Similarly, from node 2, the join request is forwarded to 3 and then to 6. Therefore, the least RRM path is $0-1-2-3-6$.

From the above example, it is clear the CIRP and RRM heuristics produce paths having similar cost and reliability. The difference between RRM and CIRP heuristics is evident from the paths returned by the two heuristics: The path returned by the RRM heuristic has a reliability of 0.762 and a cost of 13. The path returned by CIRP heuristic has reliability of 0.747 and cost of 12. Since the DRP value of the domain is 0.75, RRM is able to satisfy the

reliability requirement of the domain by incurring a little extra cost, CIRP optimized both cost and reliability without taking into account the reliability requirement.

Calculation of DRP: DRP, used for the calculation of primary path between ingress and egress routers, is computed in the following way:

Let join request $R =< S, G, B, r_r >$ reach the edge router B in domain D_1. The Core of the group G is C in domain D_2. Let the reliability of the path from S to B be r. Let r_s be the reliability of the path in domain D_1. If the reliability requirement r_r is to be satisfied then:

$$r_r \leq r_s \times r \times r(D_1, D_2) \times r(D_2)$$

$$\Rightarrow r_s \geq \frac{r_r}{r \times r(D_1, D_2) \times r(D_2)} \tag{5.6}$$

where $r(D_1, D_2)$ is the reliability of the path between domains D_1 and D_2 and $r(D_2)$ is the reliability of the path in domain D_2.

DRP, r_d, is defined as the minimum reliability of the domain such that the reliability requirement of the receiver is satisfied. Therefore, from Equation 5.6 it follows that

$$r_d = \frac{r_r}{r \times r(D_1, D_2) \times r(D_2)} \tag{5.7}$$

In Equation 5.7, $r(D_1, D_2)$ and $r(D_2)$ are not known before routing. Approximations are used to calculate the value of DRP. At edge router B, the minimum number of domains between the host and the destination is known from the Inter Domain Table stored at the edge routers. Let n be the number of domains between D_1 and D_2 in case of the shortest path. It is assumed that $r(D_2) = r_s$ and $r(D_1, D_2) = r_s^n$, which indicate that at the time of routing it is assumed that all domains in the path of the route have the same reliability as the current domain. Using the above approximation, the reliability of domain D_1 can be stated as:

$$r_s \geq (\frac{r_r}{r})^{\frac{1}{n+2}} \quad \text{if } D_1 \text{ has no on-tree nodes}$$

$$r_s \geq (\frac{r_r}{r}) \qquad \text{otherwise}$$

Since DRP is the minimum value of r_s such that the reliability requirement of the receiver is satisfied, it is obtained

$$r_d = (\frac{r_r}{r})^{\frac{1}{n+2}} \quad \text{if } D_1 \text{ has no on-tree nodes}$$
$$r_d = (\frac{r_r}{r}) \qquad \text{otherwise}$$

It is to be noted that, DRP is a scalable method of creating QoS partitions among the different domains. This simple method of calculating DRP is necessitated by the assumption that the inter-domain tables do not maintain any cost information, rather maintain only the shortest path information based on the number of domains. If the inter-domain tables contain cost specific information, then the complicated QoS partitioning algorithms mentioned in [51, 52, 53] can be applied. Simulation studies show that this simple method of DRP calculation provides good performance benefits.

5.2.1.2 Backup Path Creation

Backup paths are created between the ingress and the egress routers using the backup path table. The table contains the reliability of the most reliable path from any node X to all nodes within the domain. If any link is shared by both primary and the backup paths, resources are reserved along the link only once. In Figure 5.3, the primary path is created using the RRM heuristic and backup path is the maximum reliable path. Let Primary Join Request reach Node 1 earlier than the Backup Join Request. The Primary Join Request reserves link $0-1$, and the Backup Join Request is forwarded along link $1-4$ since link $0-1$ is already reserved.

5.2.1.3 Calculation of Weights

After creation of primary and backup paths, the egress router computes weight of the paths in the domain for the given join request. In this subsection, a simple online algorithm is described which gives approximate weight of a domain. This is illustrated in Figure 5.2, at time instance 5.

Steps of the online weight calculation algorithm described as follows:

1. In the online version of the weight calculation algorithm, the primary request packet stores two values r_{pe} and r_{pc} which are updated at each node within the domain. Similarly, r_{be} and r_{bc} are the values stored in the backup request packet. These are also updated at each node within the domain.

2. r_{pe}, r_{pc}, r_{be} and r_{bc} are initialized to 1.

3. Let primary request reach node n_i along link l_i. Let reliability of link l_i be $rel(l_i)$. Checks are carried out whether l_i has been reserved by the backup request or not. If the link l_i is not reserved by the backup request, r_{pe} is updated as $r_{pe} = r_{pe} \times rel(l_i)$. Otherwise, r_{pc} is updated as $r_{pc} = r_{pc} \times rel(l_i)$.

4. For the backup requests, r_{be} and r_{bc} are calculated similar to Step 3.

5. At the egress router, the reliability of the primary path is computed as $r_p = r_{pe} \times r_{pc}$ and reliability of the backup path as $r_b = r_{be} \times r_{bc}$. The reliability of the links common to both primary and backup paths is $r_c = r_{pc} \times r_{bc}$. Therefore, the reliability of the links exclusive to the primary path is $\frac{r_{pe} \times r_{pc}}{r_{pc} \times r_{bc}} = \frac{r_{pe}}{r_{bc}}$ Similarly, the reliability of the links exclusive to the backup path is $\frac{r_{be} \times r_{bc}}{r_{pc} \times r_{bc}} = \frac{r_{be}}{r_{pc}}$. Therefore, using Equation 5.1, weight of the of the paths in the domain for the given join request is given by

$$Weight = 1 + \frac{r_{be}}{r_{pc}} \times (\frac{r_{bc}}{r_{pe}} - 1) \tag{5.8}$$

It will be shown how weights can be calculated using the online algorithm for the example shown in Figure 5.3. The primary and backup paths are $0-1-4-3-6$ and $0-1-2-3-6$ respectively which are shown in Figure 5.4. It is to be noted that, in this case Equation 5.1 cannot be applied directly as the primary and backup paths have some common links. The weight can be calculated by separately considering link $0-1$ (part of both primary and backup), links $1-4-3$ and $1-2-3$ (primary and backup are disjoint in this segment) and link $3-6$ (part of both primary and backup). The weights of the common segments are 1 each, as the backups coincide with primary and the total reliability remains the same. The weight of the disjoint segment (using Equation 5.1) is 1.186. Therefore, the total weight is 1.186.

Table 5.1 A time chart showing the status of request shown in Figure 5.4

Time	0	1	2	3	4	5	6	7	8
PR at	0	1	-	-	4	-	3	-	6
BR at	0	-	1	2	-	3	-	6	6
r_{pe}	1	0.99	0.99	0.99	0.99 × 0.9	0.99 × 0.9	0.99 × 0.9 × 0.91	0.99 × 0.9 × 0.91	0.99 × 0.9 × 0.91
r_{pc}	1	1	1	1	1	1	1	1	0.95
r_{be}	1	1	1	0.9	0.9	0.9 × 0.9	0.9 × 0.9	0.9 × 0.9 × 0.95	0.9 × 0.9 × 0.95
r_{bc}	1	1	0.99	0.99	0.99	0.99	0.99	0.99	0.99

This complicated weight calculation is done by the online algorithm in a scalable, efficient and distributed manner.

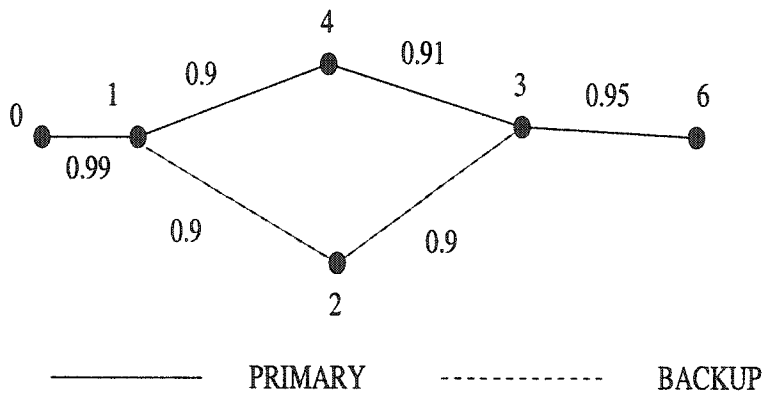


Figure 5.4 Calculation of domain weight for a given path

Table 5.1 indicates the status of the Primary and Backup Request at different instances of time. First row of the table indicates different instances of time. Second and third rows of the table indicate the node number the request has reached, at that instance of time. If the request is in transit, then it is indicated as '-'. At instance 0, both PR and BR are at node 0. This is the initial state, hence all the values r_{pe}, r_{pc}, r_{be} and r_{bc} are initialized to 1. At time 1, PR reaches node 1, while BR is still in transit. Therefore, r_{pe} is updated since the link $(0-1)$ is not reserved by the Backup Request. When the Backup Request reaches node 1, r_{bc} is updated since the link has already been reserved. At time instances $3, 4, 5$ and 6 r_{be} and

r_{pe} are updated as the requests traverse through no shared links. At time 7, BR reaches node 6. Since link $3-6$ is not reserved by the PR so, r_{be} is updated. At time 7, when PR reaches node 6 the link $3-6$ is already being reserved by BR. Therefore, r_{pc} is updated. The final values of the four parameters are shown in the last column. Using Equation 5.8 it is obtained $weight = 1 + (\frac{0.9\times0.9\times0.95}{0.95})(\frac{0.99}{0.99\times0.91\times0.9} - 1) = 1.186$.

It is to be noted that, the value returned by the online algorithm is less than the actual reliability. This is because, this algorithm approximates a serial connection of parallel paths to a parallel connection of serial paths and by Lemma B.2 (refer Appendix B), the reliability of the serial connection of parallel paths is always less than that of parallel connection of serial paths. Therefore, the connections accepted by the online algorithm will always satisfy the reliability requirements specified by the connection.

5.2.2 Reverse Pass

Let w_1, $w_2 ... w_n$ be the weights of the path in the domain D_1, D_2 $...D_n$ respectively for the join request
$R =< S, G, B, r_r >$. Let the reliability of the primary path be $r_p^\star$. Let the reliability of the end-to-end path from the Parting Node to the Core be Rel_{Tree}. Let $r_p = r_p^\star \times Rel_{Tree}$.

At the Parting Node the following checks are carried out:

1. If $(\prod_1^n w_i) < \frac{r_r}{r_p}$, the join request is forwarded to the next domain and the same check is carried out. If the Parting Node is the Core and the same condition holds, then the join request is rejected.

2. If $(\prod_1^n w_i) = \frac{r_r}{r_p}$, the join request exactly satisfies the reliability condition. Do nothing.

3. If $(\prod_1^n w_i) > \frac{r_r}{r_p}$, resources have been over-allocated. Release backup paths in some domains such that overall cost is minimized and the reliability constraint remains satisfied, using the Domain Selection Algorithm described later.

5.2.2.1 Domain Selection

Let $D = \{D_1, D_2...D_n\}$ be the domains where backup paths have been allocated with w_1, w_2 ... w_n and c_1, c_2 ... c_n are the weights and costs associated with the domains respectively, such that $w_i \geq 1 \ \forall i = 1, 2, ...n$. The domain selection problem is to find a subset of D such that the product of the weights of the paths in the domains in the subset is greater than W $(= \frac{r_r}{r_p})$ and the sum of the costs of the domains in the subset is minimized.

In Lemma B.3 (refer Appendix B), DSP has been proved to NP-Complete. Therefore, approximation algorithms need to be developed to solve DSP. Solutions for 0-1 Knapsack minimization problem can be used to solve DSP. Several well-known approximation algorithms exist for 0-1 Knapsack minimization problem. In [62], the authors proposed a greedy 2-approx solution (referred to as DSA) to the Knapsack problem. This algorithm has a time complexity of $O(nlog(n))$. Though the algorithm provides a bound, it is not amenable for distributed implementation. Therefore, it is better to use simple online heuristics such as First-Fit, which can be implemented in a distributed manner, to solve the DSP as part of the proposed Partial Protection Schemes. Note that, one can use DSA or any other algorithm (instead of First-Fit) as part of the proposed Partial Protection schemes. In the simulation studies, First-Fit heuristic has been used to solve the DSP and the results have been compared with the centralized heuristic implementing DSA.

5.2.2.2 Backup Relaxation

Let S_D be the solution returned by $DSA(D, log\frac{r_r}{r_p})$, for the join request $R = < S, G, B, r_r >$. Let D_s be the set of domains traversed by R. Let $L = D_s - S_D$. A Relax message is sent towards the joining receiver retracing the forward pass path. The message contains the set of domains included in L. When the ingress router of the domain receives the Relax message, it sets a bit in the Relax message indicating if the backup path in the domain needs to be relaxed or not. The ingress router sends the Relax Message along the backup path. All nodes in the backup path releases resources if the bit is set. The egress router on receiving the Relax message forwards it to the next domain. Before releasing resources, it is checked if the link

is part of the primary path also. If the link is part of the primary path, then the resources reserved along the link are not released.

5.3 Optimistic & Hybrid Partial Protection Schemes

In this section, different steps of the Optimistic and the Hybrid Partial Protection schemes are listed.

5.3.1 Forward Pass - Optimistic

In the forward pass of the Optimistic Scheme the backup paths are not created and reserved, but weights of the paths in the domains are calculated and they are put in the Join Request similar to the Conservative Scheme. To apply the online Weight Calculation Algorithm to this scheme each node needs to store the link from which the Backup Join Request arrives. This is reasonable to do as the backup paths are created per group basis.

5.3.2 Reverse Pass - Optimistic

As in Conservative Scheme, in the reverse pass of the Optimistic Scheme, the Domain Selection Problem is solved using the weights calculated in the forward pass. Let S_D be the solution returned by the DSA algorithm. An Add message, containing the set of domains included in S_D, is sent towards the receiver. On receiving the Add message, ingress router of domain D sets a bit in the message if D is present in S_D. The Add message is forked along primary and backup paths. Each node, which is part of the backup path, reserves the link from which the message arrives if the bit is set in the Add message. If the bit is not set, the incoming link is not reserved. The egress router on receiving both primary and backup Add messages passes them to the next domain after calculating weight using online weight calculation algorithm mentioned in Section 5.2.1. This process continues until the receiver is reached. If the calculated reliability is below the required reliability of the receiver, it cannot join the multicast group. In this scheme, the receiver may be given an option to join the multicast group with a lower reliability.

5.3.3 Forward Pass - Hybrid

In the forward pass of the Hybrid Scheme, decision is made whether to add the backup path for that domain at each ingress router. For this, the Domain Reliability Parameter (DRP) mentioned in Subsection 5.2.1.1, is used. Let a join request $R =< S, d, B, r_r >$ arrive at the ingress router. Let r_s denote the reliability value of the maximum reliable path from the ingress router to the egress router. If $r_s \geq \delta \times DRP$, backup path is created and reserved. Otherwise the backup path is not created. δ is defined as the *hybrid factor* whose value is in $[0, 1]$. When $\delta = 0$, the Hybrid Scheme reduces to the Conservative Scheme, when $\delta = 1$, the Hybrid Scheme reduces to the Optimistic Scheme. Unlike previous two schemes, a flag is maintained with each domain weight to indicate whether resources have been reserved along the backup path in the domain or not.

5.3.4 Reverse Pass - Hybrid

Let $D_r = \{D_{r_1}, D_{r_2}...D_{r_n}\}$ be the set of Domains where the backup paths are created and reserved. Let $D_u = \{D_{u_1}, D_{u_2}...D_{u_m}\}$ be the set of Domains where the backup paths are not created. Following checks are carried out at the Parting Node:

Case 1: $\prod_1^n weight(D_{r_i}) \times \prod_1^m weight(D_{u_i}) < \frac{r_r}{r_p}$: Case 1 of Conservative scheme is repeated.

Case 2: $\prod_1^n weight(D_{r_i}) \times \prod_1^m weight(D_{u_i}) \geq \frac{r_r}{r_p}$: This case has two sub-cases.

Case 2a: $\prod_1^n weight(D_{r_i}) \geq \frac{r_r}{r_p}$: The reliability requirement is already satisfied by the reserved domains. DSA$(D_r, \log \frac{r_r}{r_p})$ is called to optimize the domains already reserved. Let S_D be the set of domains returned by the DSA algorithm. Relax message is sent towards the receiver containing $(D_r - S_D)$ set of domains. The backup paths of the domains included in the $(D_r - S_D)$ set are released. The process of relaxing resources along all the domains is same as the Conservative scheme.

Case 2b: $\prod_1^n weight(D_{r_i}) < \frac{r_r}{r_p}$: The reliability requirement is still not satisfied. To select the domains resulting in cost minimization, DSA$(D_u, \log \frac{r_r}{r_p \times \prod_1^n w_{r_i}})$ is called. Let S_D be the set

returned by the DSA algorithm. If S_D is non-null, then an Add message is sent towards the receiver. The backup paths are created similar to the Optimistic scheme.

5.4 Complexity Analysis

The complexity associated with the Partial Protection Schemes is identified in terms of (a) packet overhead, (b) computational complexity, (c) storage requirements and (d) deployment.

- **Computational Complexity:** Due to the inherent nature of the schemes, most of the computation are carried out at the edge routers rather than at the core routers. Core routers update the parameters as mentioned in Section 5.2.1.3, and it can be done in constant time. Edge routers, in addition to the calculation of the weights (which can be done in constant time), runs the DSA. Centralized DSA runs in $O(nlog(n)$ time, where n is the number of domains traversed by the join request. This complexity is not high, since the number of domains traversed by the join request is typically very low. First-Fit algorithm can also be applied, where the complexity at the edge router is $O(n)$.

- **Packet Overhead:** The overhead introduced by the partial protection schemes in terms of packet size is minimal. The only overhead introduced is through the addition of weight information in the packet. Since, weight information is fractional in nature, 4 bytes of weight information (1 byte for integer and 3 bytes for fraction) per domain is sufficient to provide a precision in reliability in the order of 10^{-12}. Therefore, for 10 domains, 40 bytes of additional information in the setup packet is sufficient to provide extremely high accuracy in terms of weights of the paths in the domains. The packet size can be further reduced, if the accuracy of the weights can be sacrificed. For example, 2 bytes of weight information/domain will provide an accuracy of the order of 10^{-2}.

- **Storage Requirements:** As mentioned earlier, each core router needs to maintain primary and backup path tables and each edge router needs to maintain the inter-domain table. Since, primary path and inter-domain tables are currently stored in routers for

routing purposes, the only additional storage required by the schemes is the backup path table (reliability table) at each node.

- **Deployment:** The routers employing partial protection schemes (PPA-enabled routers) can be sparsely deployed and tunnels can be established between these routers so that the partial protection is transparent to other routers. The three schemes mentioned in this chapter can co-exist, *i.e.* different domains over the Internet can deploy different partial protection schemes.

5.5 Performance Studies

Extensive simulation studies, using NS [42], to evaluate the effectiveness of the proposed schemes for members joining the multicast group satisfying QoS and reliability constraints.

For the experiments, the various inputs were generated as follows.

- Random network topologies are generated based on the given input parameters: Number of nodes, number of links, and number of domains. Number of nodes and number of links are fixed at 40 and 80 per domain, respectively. The number of domains are fixed at 10.
- The other parameters that are kept fixed during the experimental studies are: (i) Average Link Bandwidth = 1.5 Mbps (ii) Average Link Reliability = 0.999 (iii) Average Link Cost = 1
- Multicast traffic represent, on an average, around 20% of the total traffic.
- Each simulation point is an average of 10 random observations.
- For the experimental purposes, the CIRP algorithm (discussed in Subsection 5.2.1.1) was used for the creation of the primary paths. Here the First-Fit algorithm was used for the Domain Selection Problem.
- All the ranges provided in the text (for the mean value plotted) are computed for a confidence interval of 95%.

5.5.1 Performance Metrics

Let R_1, R_2 ... R_N be the set of N join requests. Before defining the metrics two functions are defined, which will be used for defining the performance metrics.

- $accepted(R_i) = 1$ if R_i is accepted.
- $cost(R_j)$ = cost of the total reserved path for receiver R_j

For a join request R_k that is rejected, all functions return value 0. Following metrics are used for the analysis of the schemes.

- *Average Call Acceptance Rate (ACAR):* ACAR is defined as the average percentage of join requests that is accepted by the routing algorithm.

$$ACAR = \frac{\sum_1^N accepted(R_i)}{N} \times 100 \tag{5.9}$$

- *Average Cost Per Receiver (ACPR):* Let total time be divided into n equally spaced time intervals. Cost Per Receiver(CPR) at time j is defined as the average cost of reserved paths in the interval $[(j-1), j]$. Let N_j be the number of join requests processed in the interval $[(j-1), j]$. ACPR is defined as CPR averaged over n time intervals.

$$ACPR = \frac{1}{n}\sum_{j=1}^{n} \frac{\sum_{i=1}^{N_j} cost(R_i)}{\sum_{i=1}^{N_j} accepted(R_i)} \tag{5.10}$$

To evaluate the effectiveness of the algorithms, three sets of experiments were performed based on the performance metrics mentioned above.

- *Selection of the Hybrid Factor (δ):* In this set of experiments, the Hybrid Scheme is compared with Optimistic and Conservative schemes for different values of δ. This set of experiments helps us to select a suitable δ which provides a balance between ACAR and ACPR.
- *Comparison with the Centralized Scheme:* The online schemes mentioned in this chapter involve approximations in different levels. In this set of experiments, the effectiveness of the approximations are evaluated, by comparing the proposed online schemes with the

centralized scheme. In case of the Centralized Scheme, DSP is solved according to the 2-approx algorithm mentioned in Section 5.2.2.1 and resources are allocated accordingly in the forward pass itself, assuming that the full topology information is available at the joining node.

5.5.2 Selection of δ

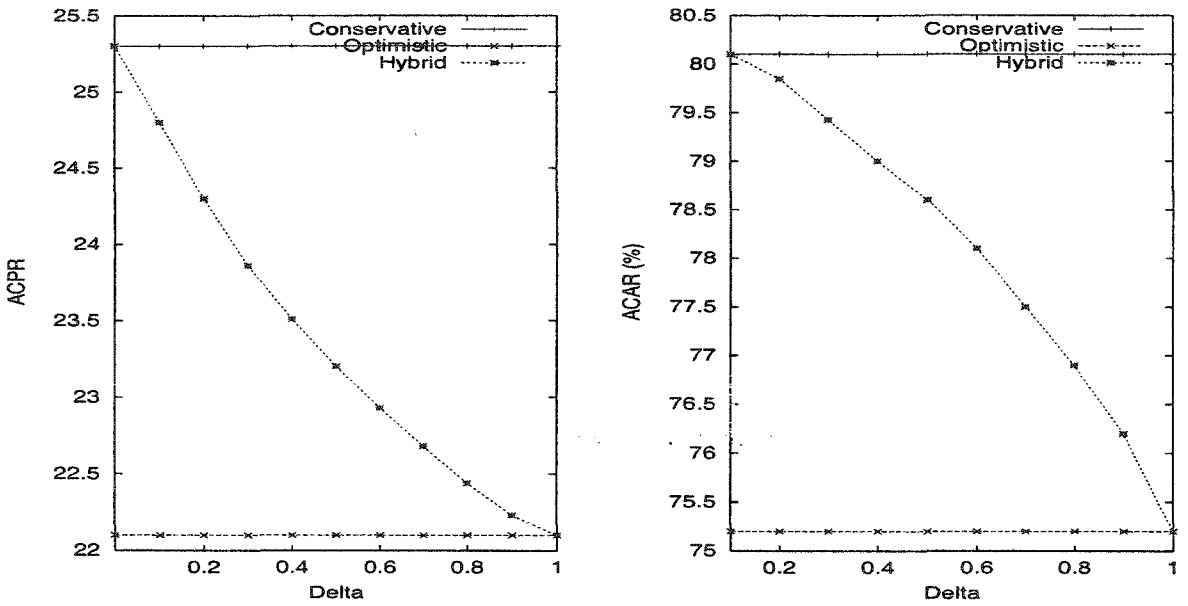


Figure 5.5 Variation of (a) ACPR and (b) ACAR with δ

In this set of experiments, the ACPR and ACAR values of the Hybrid Scheme are measured for different values of δ. In the Figure 5.5, experimental results are shown. For this set of experiments, the connections have reliability requirements of 0.99 and bandwidth requirements of $100KB$. Results shown in Figure 5.5(a) vary by around $\pm 3.3\%$ of the actual mean. Results shown in Figure 5.5(b) vary around $\pm 2.5\%$ of the actual mean. Both are calculated based on 95% confidence interval.

The Conservative Scheme offers higher cost than the Optimistic Scheme as shown in Figure 5.5(a). The reason for this is that, resources may be reserved in excess during the forward pass, which will have to be released later. Optimistic Scheme, on the other hand, does not

reserve resources in the forward pass and hence ACPR is less. Hybrid scheme, for different values of δ, performs in between the Optimistic and the Conservative schemes. Hybrid Scheme, with $\delta = 0$, behaves similar to the Conservative Scheme and with $\delta = 1$, behaves similar to the Optimistic Scheme. Hybrid Scheme, for other values of δ, performs in between. A closer look at the Figure 5.5(a) provides with the information that, ACPR of trees produced under Hybrid Scheme are within 5% of the Optimistic Scheme, when $\delta \leq 0.5$.

In Figure 5.5(b), the results of the variation of ACAR with δ are shown. For a high reliability requirement (0.99), the Conservative Scheme performs the best having a higher ACAR value than that of the Optimistic Scheme. This indicates that, it is a good idea to reserve resources in the forward pass itself, as resources may not be available in the reverse pass. Hybrid Scheme, as in the previous case, performs in between. Also similar to the previous case, Hybrid Scheme with $\delta = 0$ performs similar to the Optimistic Scheme and Hybrid Scheme with $\delta = 1$ performs similar to the Conservative Scheme. ACAR values of the connections under Hybrid Schemes are within 3% of the Conservative Scheme, when $\delta \geq 0.5$

From the graphs it is evident that Hybrid Scheme with a δ value of 0.5 provides a balance between call acceptance rate and tree cost. Therefore, $\delta = 0.5$ is selected for the Hybrid Scheme for the subsequent sets of experiments.

5.5.3 Comparison with Centralized Scheme

In the Figure 5.6, variation of ACPR and ACAR is shown with varying reliability requirements of the connections. In this set of experiments, the inter-arrival time between the receivers is set at $250ms$, and the bandwidth requirement of the connections is $100KB$. The results shown in the graph Figure 5.6(a) vary within $\pm 3.38\%$ and the results shown in Figure 5.6(b) vary within $\pm 2.21\%$ of the actual means. For low values of reliability requirement (< 0.98), Optimistic scheme performs better, both in terms of ACAR and ACPR. However, as the reliability requirements of the connections increase (≥ 0.98), Conservative Scheme performs better than the Optimistic Scheme in terms of ACAR. This happens because at higher reliability requirement (constrained case), Conservative Scheme gains as it reserves resources in the for-

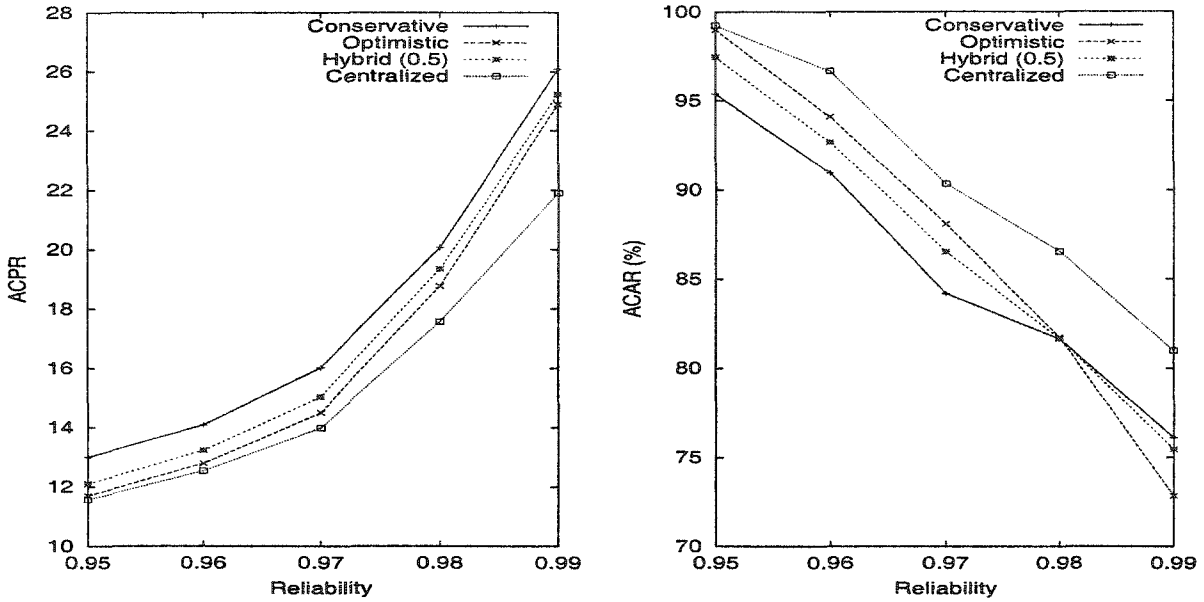


Figure 5.6 Variation of (a) ACPR, (b) ACAR with reliability

ward path. Also, the Centralized algorithm performs much better (both in terms of ACAR and ACPR) than the online algorithms at higher reliability requirements, though ACAR of all schemes decreases with increasing reliability. The reason for this is because, with increasing reliability requirements of the connections, the number of paths satisfying the reliability requirement decreases and hence there is a resource contention. This contention is resolved the best, if path allocation is done in a centralized fashion.

In another set of experiments, the effect of bandwidth (BW) requirements and node degree are studied, where reliability requirement is fixed at 0.99, and inter-arrival time between receivers is fixed at $250ms$. In Figure 5.7(a), the variation of ACAR is shown with BW requirements. The results shown in this figure vary within the range of around $\pm 2.83\%$ of the actual mean. With increase in BW requirements, the ACAR drops for all schemes. Among the online algorithm, Conservative Scheme produces the best performance followed by Hybrid and the Optimistic schemes. Centralized scheme offers better performance than all the on-line schemes under high BW requirements. In Figure 5.7(b), the effect of node degree is studied. In this figure, the results vary within the range of $\pm 2.75\%$ of the actual mean. With increase in

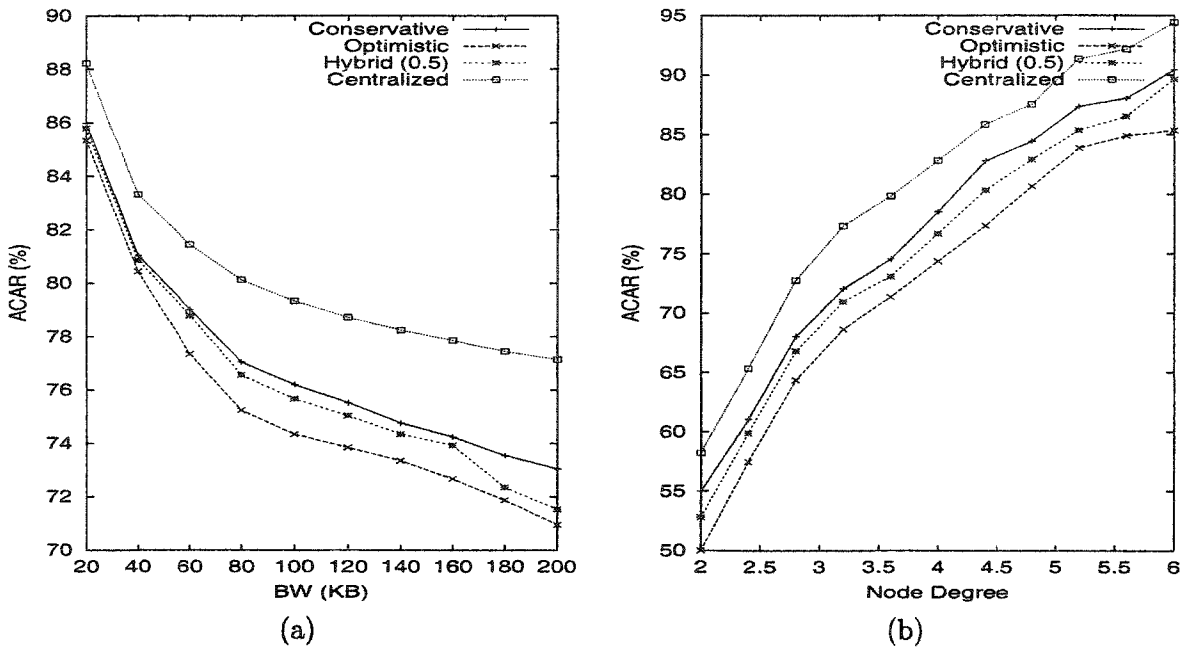


Figure 5.7 Variation of ACAR with (a) BW requirement and (b) node degree

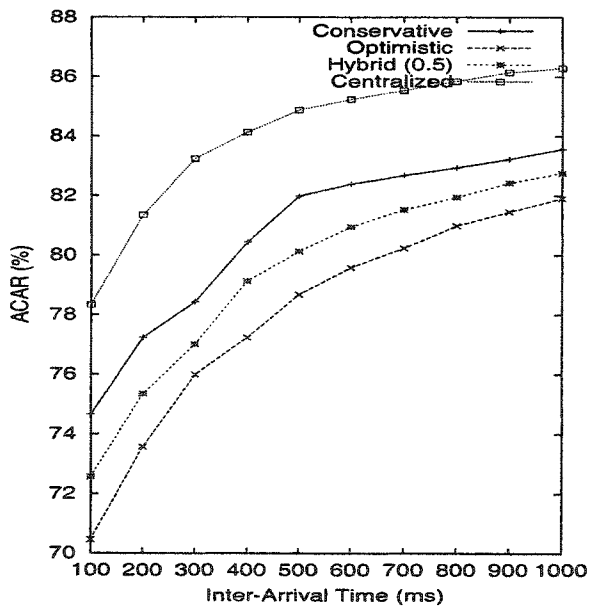


Figure 5.8 Variation of ACAR with inter-arrival time

node degree, the network offers more paths satisfying the reliability constraints and hence the ACAR decreases. When the network is sparse, the ACAR offered by the centralized scheme is around 15% higher than that offered by Optimistic scheme (the lowest among the online algorithms). When the network is dense, the ACAR offered by the centralized scheme is around $5-7\%$ higher than the Optimistic scheme.

In Figure 5.8, the variation of ACAR is studied with receiver Inter-Arrival time, where the bandwidth and reliability requirements are fixed at $100KB$ and 0.99 respectively. In this figure, the results show a variation of around $\pm 2.91\%$ of the actual mean. As the inter-arrival time increases (i.e. the group becomes less dynamic), the ACAR increases for all the schemes. For highly dynamic groups (low Inter-arrival time), Centralized Scheme performs nearly 15% better than the Optimistic Scheme (which has the lowest ACAR). When the group becomes less dynamic (inter-arrival time increases), Centralized Scheme performs $7-8\%$ better than the Optimistic scheme. The Conservative Scheme also performs around $6-7\%$ better than the Optimistic Scheme for low inter-arrival times, and around $3-4\%$ better for higher inter-arrival times.

5.6 Summary

In this chapter, a scalable approach, called the Partial Protection Approach (PPA), has been proposed to solve the reliability constrained least cost dynamic multicast routing (RCLCR) problem. Three two-pass schemes have also being proposed to implement the solution approach of the above problem. The schemes are scalable and have little extra overhead in terms of packet overhead, message complexity and computational complexity. Detailed simulation studies have been carried out to evaluate the performance of the three schemes. The simulation studies have shown that:

- The Centralized scheme performs $5-7\%$ better than the online schemes under moderate user requirements (BW and reliability), and denser network. Under more constrained user requirements and sparser networks the Centralized scheme performs $15-20\%$ better than the online schemes. Average node degree in the Internet is on the higher side

(3.3 – 3.5). Therefore, in the Internet, for moderate user requirements, simple online schemes based on First-Fit performs close to the Centralized scheme.

- Optimistic Scheme produces lesser cost trees than the Conservative scheme as resources are reserved in an Optimistic manner.

- ACAR offered by the Optimistic Scheme is higher than that offered by the Conservative Scheme under low reliability requirements. However, there is a switch at reliability requirement of 0.98 where from the Conservative Scheme starts performing better. It is to be noted that the results are obtained for a $80 - 20\%$ (unicast-multicast) traffic mix. The absolute values may change depending on the traffic mix.

- Hybrid Scheme performs closer to the Conservative or Optimistic Scheme depending on the value of the δ (hybrid factor). Hybrid Scheme with high δ value, performs closer to the Optimistic Scheme, while Hybrid Scheme with low δ value performs closer to the Conservative Scheme. A δ of 0.5 offers a good balance, as its performance remain close to the Centralized scheme under all simulated conditions.

To summarize, the four approaches viz., Centralized, Conservative, Optimistic and Hybrid schemes should be used depending on the nature of the requirements. If the requirements are really stringent for all cases and the resources are scarce (sparser networks), it is better to go with the Centralized scheme. However, in this case more computation is needed. If the computational capabilities at the routers are less, but the reliability requirement is very high, Conservative Scheme is better. In other cases, Optimistic scheme can be used. Hybrid scheme can be used if a trade-off between Optimistic and Conservative is desired.

CHAPTER 6 Tree Maintenance in End System Multicasting

With the advent of overlay networks [33, 34], where the nodes arrange themselves in an overlay, researchers are examining the possibilities of having network functionalities like packet forwarding [35], multicasting [36, 37] etc. at the application layer. As an alternative to IP multicasting, researchers have proposed the overlay multicasting approach [36, 37, 38, 39], wherein the complex multicasting features like replication, group membership management and multicast routing are implemented at the application layer, assuming only the end-systems or hosts are responsible for multicasting. End System Multicasting (ESM) is an example of overlay multicasting. In ESM, the end-systems organize themselves in an overlay spanning tree for data delivery. Each link in the ESM spanning tree corresponds to the unicast path in the actual physical network. As all the complexities are handled at the hosts rather than at the routers, it offers some distinct advantages over its IP counterpart. (i) ESM is easier to implement, as there is no complexity required at the routers. Therefore, it is also scalable. (ii) Complex functionalities like congestion control, reliable data transfer are handled separately at the unicast level, and therefore manageable. (iii) Adding security features to multicasting is easier as routers are not involved.

In spite of these advantages, ESM has some issues which need future research attention. (i) The quality of the multicast tree produced using ESM is worse than that produced using IP multicasting. In multicasting quality of a multicast tree may refer to the cost of the multicast tree, or the average delay to all the receivers, or some other optimization metric. (ii) Since each node in the ESM tree is a host, therefore the nodes have limited capability in terms of bandwidth and processor capabilities. This is abstracted as *fanout constraint* which identifies the number of outgoing links that the multicast tree can support. (iii) The multicast sessions

are unreliable as they depend on the hosts for data transmission.

Therefore, IP multicasting offers performance benefits while ESM offers simplicity in implementation. ESM is still in its infancy and many issues need to be resolved before it can become a serious alternative to IP multicasting. Recently, efforts to combine IP and ESM has also taken place [40]. In this dissertation, this new trend is recognized in multicasting research and propose solutions for building efficient and scalable multicast trees for ESM.

6.1 End System Multicasting Approaches

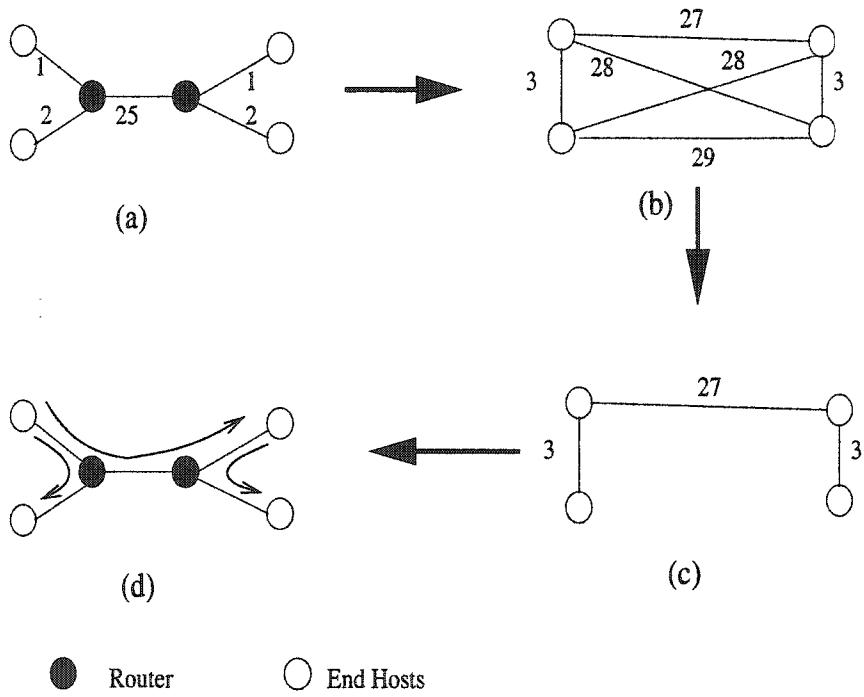


Figure 6.1 Creation of mesh-first ESM tree

Multicast trees in ESM can be constructed using two approaches: (a) Tree-first approaches and (b) Mesh-first approaches.

- *Tree-first Approach:* In this type of approach, members directly select their upstream neighbors from among the known members. Therefore ESM tree is constructed on the physical network directly. Example protocols for Tree-first techniques are Yoid [63],

Overcast [64], etc. Tree-first protocols produce good quality multicast trees. However, these protocols are complex and require specific loop prevention mechanisms.

- *Mesh-first Approach:* In this type of approach, a mesh is constructed on the physical network. A mesh is a subset of a fully connected virtual graph. A multicast tree is constructed on the mesh. The group management becomes much more easier using this approach as group management is abstracted at the mesh level rather than at the tree level, standard tree creation approaches can be employed, and loop management becomes a non-issue on a mesh. On the flip side, mesh management becomes a critical issue, and influences the construction of the tree to a great extent. Narada [36] and NICE [37] are examples of this approach.

  - **Narada:** Narada is an example of the mesh-first protocols. In the Narada protocol a mesh is created based on the members of the multicast group. Whenever a new member joins the multicast group, the information is distributed throughout the group through a distance vector like protocol running on the application layer. A multicast tree is created on the constructed mesh. When a node leaves the multicast group, the mesh is reconstructed.
  - **NICE:** NICE protocol extends Narada by creating a mesh in a hierarchical manner. The resultant mesh helps in reducing the service distribution when a member leaves the multicast group. The protocol therefore, increases the scalability of the Narada protocol.

An example of tree creation using Mesh-first approach is shown in Figure 6.1. In the example, the mesh is the fully connected virtual graph. In Figure 6.1(a), the physical network is shown, where the dark circles indicate the routers and the light circles indicate the end hosts.In Figure 6.1(b), the mesh is shown. In Figure 6.1(c), a spanning tree is constructed on the constructed mesh. In Figure 6.1(d), the actual data transfer is shown. The cost of multicast tree thus created is 33. For the same example, IP multicasting would have produced a data

delivery tree having cost of 31. This clearly identifies the trade-off between IP multicasting and ESM.

6.2 Problem Statement and Motivation

In this section the ESM tree management problem is first formally defined, and then motivation is provided for the approach taken in this chapter to solve the problem.

6.2.1 Problem Definition

Given an undirected network $N = (V, E)$, where V is the set of vertices or nodes, and E is the set of edges or links. Let e_{ij} be an edge between nodes i and j, such that $e_{ij} \in E \; \forall i, j \in V$. $D_{i,j}$ be the delay associated with the edge e_{ij}. Let S be the set of all shortest paths in N, and s_{ij} is the shortest path between nodes i and j, such that $s_{ij} \in S \; \forall i, j \in V$. Let M be the set of members in a multicast session, such that $M \subseteq V$. Let F_i be the fanout constraint of each member i. The problem is to construct a multicast tree $T = (V_M, S_M)$, $S_M \subseteq S$ spanning all members so that the average delay to all members is minimized such that $d_i \leq F_i \;\; \forall \, i \in M$, where d_i is the degree at node i.

The above problem is known as ESM tree management problem. In this chapter, whenever the quality of a multicast tree is mentioned, average delay is used as the metric. The ESM tree management problem can be tackled using two methods. The first method creates a degree constrained spanning tree on a fully connected virtual graph. As shown in [65, 66], the problem is NP-Complete. Tree-first techniques use this approach. The second approach, is through construction of a degree-constrained K-spanner on the fully connected virtual graph. A degree-constrained K-spanner is a subset of the fully connected virtual graph such that, each node satisfies the degree constraint and the shortest path between any two node in the K-spanner is not more than K times the shortest path in the fully connected virtual graph. As shown in [67], this problem is also NP-Complete. In this approach, after the construction of the K-spanner, a spanning tree is constructed on the K-spanner. Mesh-first techniques use this approach. In this chapter, a technique called Mesh Tree Interaction (MTI)is proposed, which

combines the management ability of the mesh-first approaches, and the performance benefits of the tree-first approaches.

6.2.2 Motivation

As mentioned earlier, both Tree-first and Mesh-first approaches are based on NP-Complete problems. Therefore, both the approaches use approximations to construct spanning tree and K-spanner respectively. While the first approach constructs multicast tree directly; the second approach constructs a subset of the complete virtual graph and then constructs the multicast tree from the graph using standard tree construction techniques. While the former approach is difficult to manage (as mentioned earlier), the latter is simple to implement as mesh and tree construction are made independent from one another. Independent mesh and tree, though result is simplicity in mesh management, result in creation of low-quality tree as mesh construction is done without taking the actual tree construction into account. Therefore, mesh construction may result in creation of mesh links which do not contribute to the improvement of the quality of the multicast tree. It is to be noted that mesh provides redundancy, however it is the multicast tree which is used for actual data dissemination. Therefore, quality of multicast tree is absolutely critical for group communication. Quality of a multicast tree can either be the cost of the tree or the average delay to all the members of the tree. In this chapter, average delay is referred as the 'quality' of the multicast tree. The above-mentioned point is illustrated with an example shown in Figure 6.2.

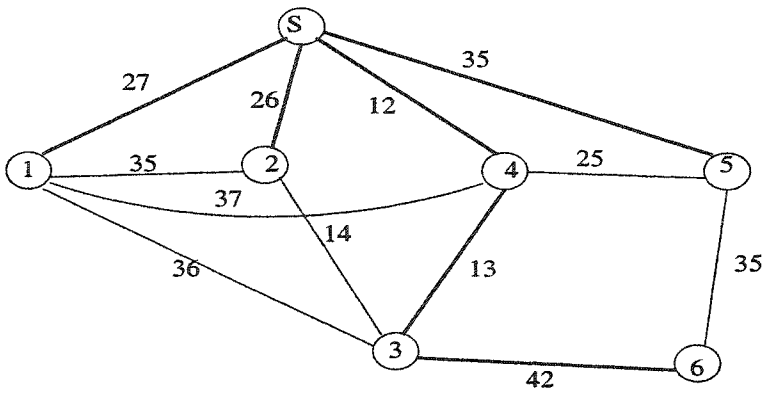


Figure 6.2 An example mesh

In this figure, an example mesh is shown which is a subset of a fully connected virtual graph. Each link in the figure is the shortest path between the nodes. The number shown with each link indicates the delay of the shortest path between the two nodes. The links which are part of the multicast tree are indicated in the figure using darker lines. Let S be the source of the multicast communication; shortest path from node 6 to node 4 has a delay of 12. However, link $(6-4)$ is not part of the mesh, as shown in the figure. Let the fanout limit for each node in this example be 4. Therefore, node 6, if connected to node 4 can provide a better delay path for itself. However, node 4 has reached its fanout limit (in this case 4). It is to be noted that in mesh-first approaches, each node independently tries to satisfy the fanout constraint and find the best neighbor at the mesh level. Since the Mesh-first protocols have no way to identify that link $(6-4)$ if added to the mesh, it will result in a better tree, will not add link $6-4$ to the mesh.

This example shows that there is a need for interaction among the constructed mesh and tree so that "unimportant" mesh links can be removed to eventually produce better quality tree. Referring back to the above example, if node 4 had somehow realized, during link addition itself, that links $(1-4)$and $(4-5)$ are "unimportant" links, or links which will not result in a better quality tree, then one of these links could be removed in this case and the link $(6-4)$ can be accommodated such that the quality of the overall multicast tree is improved. In other words, a continuous interaction between mesh and tree is needed to construct a better quality mesh, which eventually leads to the construction of better quality tree. In this chapter a mesh-tree interaction approach is proposed which achieves the above. MTI identifies whether a link is important (part of the tree) or not (part of the non-tree mesh), and takes action based on the information. MTI technique achieves the following objectives:

- MTI achieves a better "quality" tree than other Mesh-first protocols.
- MTI is easily deployable, as group management is still controlled at the mesh level, instead of tree level in case of Tree-first protocols.
- MTI can be used in isolation, as well as in conjunction with any of the existing Mesh-first

protocols like Narada and NICE.

In addition, a restricted version of MTI called R-MTI is also developed, which improves the scalability of the protocol. The trade-off between performance and scalability of MTI and R-MTI are studied as part of the simulation studies in Section 6.6.

6.2.3 Assumption and Model

Here, the assumptions and the model used throughout the chapter, to solve the ESM tree management problem, are listed as follows:

- A dynamic multicast session is assumed, i.e., members can leave and join the multicast session at any time.
- The ESM protocols does not have any information of the underlying network topology.
- A distance vector like protocol is assumed to run in the application layer, with messages exchanged over the mesh.
- A flat mesh structure is assumed.

The rest of the chapter is organized as follows: In Section 6.3, an overview of the MTI approach is provided with important definitions to be used for the rest of the chapter. In Section 6.4, the different steps of MTI are described in detail. A Restricted MTI (R-MTI) approach is outlined in Section 6.5. Finally, in Sections 6.6, the simulation results are provided.

6.3 Mesh-Tree Interaction (MTI) Overview

Mesh-tree Interaction (MTI) is a mechanism to create "good" quality multicast tree through the improvement of the mesh in an iterative manner. While all the mesh-first protocols create the tree from the mesh, MTI improves the quality of the mesh based on the constructed tree, which in turn improves the quality of the tree. On an abstract level, the main difference between a standard mesh-first approach and MTI lies in the inherent understanding of the nature of the multicast tree, which is used to construct a better mesh. The basic difference is illustrated

in Figure 6.3. While in Mesh-first approaches quality of the tree depends enormously on the quality of the underlying mesh, MTI uses an iterative process as tree structure influences the mesh, which in turn influences the tree structure.

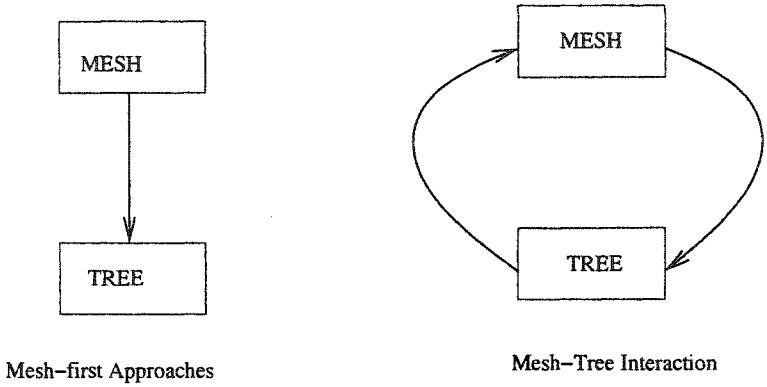


Figure 6.3 Mesh-first approaches vs. MTI

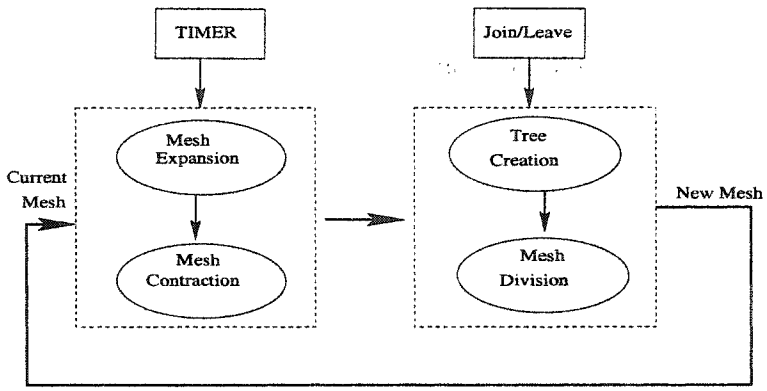


Figure 6.4 Interaction between the different steps of MTI

To implement MTI, the mesh is divided into Primary Mesh (PM) and Secondary Mesh (SM), where PM contains all the links that are part of the multicast tree and SM contains all the links that are not part of the multicast tree. The second difference between Mesh-first approaches and MTI is the selection of the "best" neighbors for mesh optimization. Goodness of neighbors are identified by a parameter called the Upstream Correlation Factor (ρ), which determines how "good" the upstream neighbor of a node is.

Mesh-tree Interaction has three main steps which differentiate the approach from the traditional mesh-first and tree-first approaches:

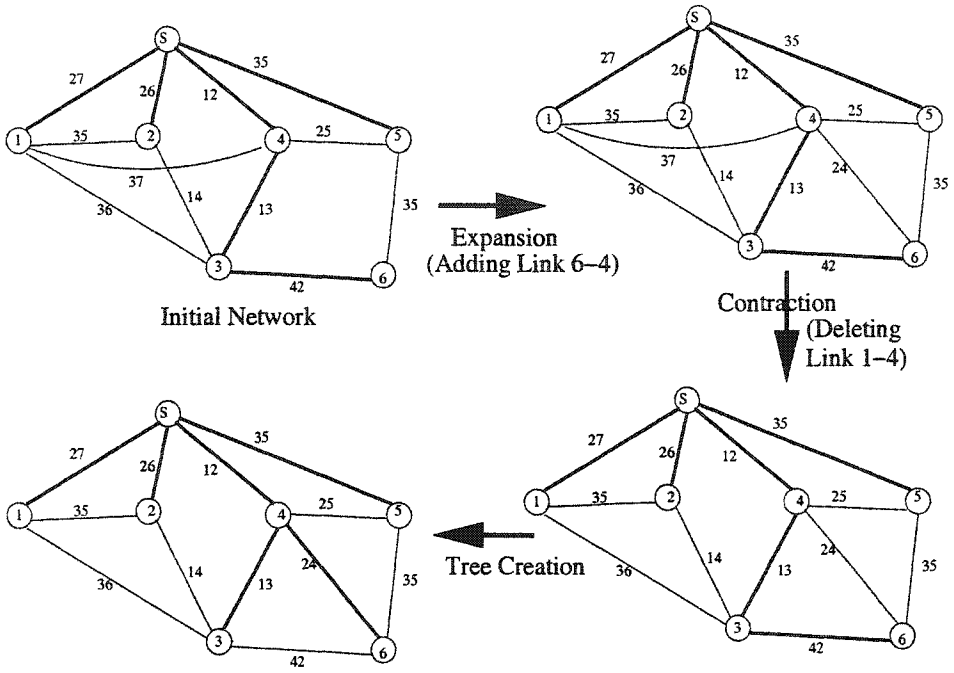


Figure 6.5 An MTI example

- *Mesh Division:* This is the first step where the mesh is divided into Primary Mesh and Secondary Mesh. The intuition behind this is to differentiate between tree links and non-tree links within a mesh. Mesh division also helps MTI to keep track of the important links which constitute the Primary Mesh, and unimportant links which constitutes the Secondary Mesh. This prioritization helps MTI to accommodate links which eventually results in the construction of a better multicast tree.

- *Mesh Expansion:* In this step, links are added to the secondary mesh which eventually leads to the improvement of the quality of the mesh.

- *Mesh Contraction:* Mesh contraction takes place when certain mesh links are deleted to make way for mesh expansion. It is to be noted that Mesh Contraction is a by-product of Mesh Expansion, and is only called when Mesh Expansion leads to the violation of fanout constraints. The interaction between the different steps conceptualized in a figure illustrated in Figure 6.4.

- *Tree Creation:* Tree is created using shortest path on the total mesh. Tree creation step

leads to the mesh division. Tree creation step is not much different from the tree creation protocol described in [36]. However, MTI does not restrict the implementation of other multicasting tree building protocols.

In Figure 6.5, the different steps of MTI are illustrated with the help of the example shown in Figure 6.2. The figure shows the initial mesh, on which mesh expansion leads to the addition of link $(6-4)$ to the mesh. This leads to the violation of fanout constraint at node 4. This leads to mesh contraction, and link $(1-2)$ is deleted from the mesh. Finally, a tree is constructed on the modified mesh. This modification leads to a decrease of average delay from 32.17 in the original case to 27.33 in the final case.

Property 1: $-1 \leq \rho \leq 1$

Property 2: If $\rho^s_{ij} = \delta$, then $\rho^s_{ji} = -\delta$.

Property 3: $\rho^s_{is} = 1$.

Property 4: Let i, j and k are three nodes and source is s, $\rho^s_{ij} > 0$ and $\rho^s_{jk} > 0$, then $\rho_{ik} \geq \frac{\rho^s_{ij}.\Delta_{ij}+\rho^s_{jk}.\Delta_{jk}}{\Delta_{ij}+\Delta_{jk}}$.

Property 5: If i_j is the best upstream neighbor of i_{j-1} $\forall j = 1, 2...n$, and $\rho^s_{i_{j-1},i_j} > 0$, then i_1, i_2 ... i_n cannot form a loop.

Figure 6.6 Properties of ρ

6.3.1 Upstream Correlation Factor (ρ)

To understand Upstream Correlation Factor (ρ), the following terms are defined:

- *Shortest Path Delay (Δ_{ij}):* Δ_{ij} determines the delay of the shortest path between nodes i and j.
- *Upstream Neighbor (η^s_i):* η^s_i indicates the upstream neighbor of node i with respect to source s.
- *Upstream Correlation Factor (ρ^s_{ij}):*ρ^s_{ij} determines the quality of the upstream neighbor j of node i, where s is the source of the multicast tree. Mathematically,

$$\rho^s_{ij} = \frac{\Delta_{is} - \Delta_{js}}{\Delta_{ij}} \tag{6.1}$$

Properties of ρ are shown in Figure 6.6. Refer to Appendix C for the proofs.

ρ or the Upstream Correlation Factor is an interesting and important metric to determines the quality of the upstream neighbor. $\rho^s_{ij} = 1$ indicates that the shortest path of i goes through j, and $\rho^s_{ij} = -1$ indicates that the shortest path of j goes through i. Higher the value of ρ, lower is the delay of the path through the upstream node. The reason ρ is such an important metric in ESM context is that it determines the quality of the upstream neighbor without actually knowing anything about the actual path. Therefore, the metric can be measured by sending ICMP packets to different nodes and measuring the delay experienced by the packets. Therefore, the metric does not violate the basic premises of the ESM architectures.

Now, to illustrate the usefulness of ρ with the help of an example. Let node A and node B have shortest delay path to these source as 10 and 8 respectively. The current delay offered by the two nodes are 10 and 12 respectively. This is a possible scenario, as the current delay path depends on the quality of the underlying mesh. Now, a node C has shortest delay path to nodes A and B as 2 and 3 respectively. Therefore, the current delay offered to node C, if node A is C's upstream neighbor is 12, while that offered if node B is the upstream neighbor is 15. However, node B has the capability of offering a delay path of 11 to node C. Therefore, it is a "better" upstream neighbor. ρ value reflects this as $\rho_{CA} = 0.5$ and $\rho_{CB} = 1$. From Property 1, ρ_{CB} is the maximum ρ value possible for any neighbor of C. Therefore, B is the best upstream neighbor.

6.4 Different Steps of MTI

As mentioned earlier, MTI consists of three steps: (a) Mesh Division, (b) Mesh Expansion, (c) Mesh Contraction and (d) Tree Construction. The steps are described in detail, in the following subsections. In this chapter, only the first three steps are discussed as MTI is flexible, and any tree construction algorithm can be employed.

6.4.1 Mesh Division

In this step, the mesh is divided into Primary Mesh (PM) and Secondary Mesh (SM). PM consists of the links which are part of the multicast tree, and SM consists of all the links which are not part of the multicast tree. Each mesh consists of a two lists. SM consists of two lists SM^+ and SM^-, while PM consists of PM^+ and PM^-. List SM^+ consists of all links having positive ρ value among the SM links, sorted in descending order such that the head of the list contains the link having the maximum ρ value. On the other hand, SM^- consists of all links having negative ρ values among the SM links, arranged in ascending order such that the head of the list contains the link having minimum ρ value. The head and tail of SM^+ are called SM Maximum+ (Γ^+_{SM}), SM Minimum+ (γ^+_{SM}) respectively. The head and tail of SM^- are called SM Minimum- (γ^-_{SM})and SM Maximum- (Γ^-_{SM}) respectively. It is to be noted that, the Minimum and Maximum of SM^+ and SM^- are reversed. The reason behind this is that, the "importance" of a link is higher if the ρ value is lower, if the link has negative ρ value.

The links in PM are also arranged in PM^+ and PM^- in a similar way.

Table 6.1 Mesh division for node 4

Types	ρ	Links
PM^+	1	$(4-S)$
PM^-	-0.85	$(4-3)$
SM^+	-	None
SM^-	-0.41	$(4-1)$
SM^-	-0.52	$(4-5)$

Table 6.1 illustrates the mesh division concept based on the example mesh shown in the Figure 6.2. The values are for Node 4. PM^+ has only one link having ρ value of 1.0. PM^- also has only one link $(4-3)$, having ρ value of -0.85. SM^+ is empty. SM^- has two links $(4-1)$ and $(4-5)$ having ρ values of -0.41 and -0.52 respectively. $\Gamma^+_{PM} = \gamma^+_{PM}$, and $\Gamma^-_{PM} = \gamma^-_{PM}$, as there is only one link each in PM^+ and PM^-. Link $(4-1)$ is γ^-_{SM} and link $(4-5)$ is Γ^-_{SM}.

6.4.2 Mesh Expansion and Contraction

Mesh expansion forms the second step of MTI which may or may not lead to mesh contraction. Under mesh expansion each node proactively tries to expand the mesh by adding a neighbor to its mesh, which is better than at least one of its current neighbors. The "goodness" of a neighbor is measured by the ρ value mentioned earlier. Higher is the ρ value of the neighbor, better is the neighbor. Mesh expansion may lead to the violation of fanout constraint in some node, then mesh is contracted by deleting some links so that a "good" neighbor can be accommodated.

The main principle behind mesh expansion is that each node (say i) attempts to find its Best Upstream Neighbor ($\mu+$). To identify $\mu+$ each node searches for the neighbor having the maximum ρ. Each searching node (i) searches for a set of candidate neighbors and calculates the ρ value for each of them. The candidate neighbor having the highest ρ value is selected as the best candidate neighbor (say n). If both i and n have enough resources (fanout is less than the limit), the link is accommodated. In this case, mesh contraction is not called. Otherwise, mesh contraction is called.

In case of mesh contraction, some candidate links are found connected to both i (mentioned as $ReplaceLink_i$) and n (mentioned as $ReplaceLink_n$). If the resource constraint at node i is violated (the fanout at node i exceeds the limit) because of the addition of the link $(i-n)$, $ReplaceLink_i$ is deleted from the mesh to accommodate for link $(i-n)$. Similarly, if the resource constraint at node n is violated because of the addition of the link $(i-n)$, $ReplaceLink_n$ is deleted from the mesh. This step is part of the mesh contraction, as mentioned before. To search for $ReplaceLink_i$, firstly SM^+ of i is searched. The reason behind this is that, all links which are present in SM^+ are not part of the tree.

If a link is found (say j) which has lower ρ than ρ_{in} (ρ value of link $i-n$), then j is identified as the $ReplaceLink_i$. If SM^+ is empty, γ^-_{SM} is identified as the $ReplaceLink_i$. If both SM^+ and SM^- are empty, then $ReplaceLink_i$ is identified from PM^+ if ρ value of the link is PM^+ is less than ρ_{in}. To identify $ReplaceLink_n$, if n is not the source same sequence is followed, only this time first SM^- is searched, then SM^+ and then PM^- as ρ changes sign

in n (Property 2). However, if n is a source then $ReplaceLink_n$ is identified as a link (say j) in the PM^+ of n, if $\Delta_j > \Delta_{in}$. The reason for using Δ instead of ρ is because $\rho_{in} = 1$, in this case as n is the source (Property 3). The pseudo-code of the mesh contraction algorithm is described in the Appendix D.

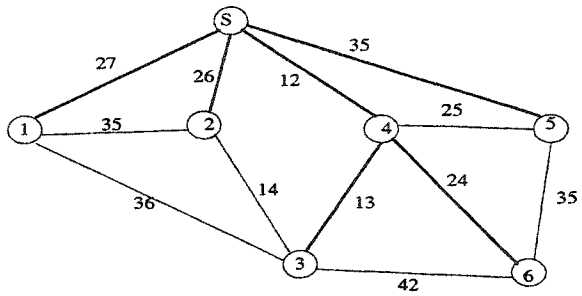


Figure 6.7 New mesh after expansion/contraction

To illustrate the above algorithm with the help of an example, refer back to the Figure 6.2. Let $\Delta_{6S} = 36$ and $\Delta_{64} = 27$. This means that the shortest distance from node 6 to the source is 36, and the shortest delay from node 6 to node 4 is 27. From the figure, the shortest distance from source to node 4 is 12 *i.e.* $\Delta_{4S} = 12$. Therefore, $\rho^S_{64} = 0.9$ and $\rho^S_{46} = -0.9$. Referring back to the Table 6.1, $\gamma^-_{SM} = (4-1)$, and $\rho^S_{41} = -0.41$. Node 6 does not violate its fanout constraint by accommodating the link. However, node 4 does. Therefore, mesh contraction algorithm needs to be called at node 4. There exists at least one link in the SM^- of node 4 having lesser importance than link $(4-6)$. Therefore, γ^+_{SM} *i.e.* link $(4-1)$ is removed from the mesh to accommodate link $(4-6)$. Hence, link $(4-1)$ is removed from the SM and link $(4-6)$ is added to the SM^- of node 4. After tree creation, this link will be added to the tree. After the expansion/contraction of the mesh, the mesh looks like Figure 6.7. After the mesh addition and tree creation, the average delay improves from 32.17, in the first case to 27.33.

6.5 Restricted MTI (R-MTI)

Comparing between MTI and any other flat mesh based ESM protocol (Narada for example), the following points need to be considered:

- *Message Complexity:* Number of messages exchanged is an important parameter which

measures the scalability of the protocol. Message complexity of MTI and any mesh-first protocol is comparable. In both the cases each node need to send $O(n)$ messages, where n is the number of nodes in the mesh. Therefore, for very large n, a huge number of messages need to be exchanged which reduce the scalability of the protocols.

- *Message size:* In case of standard mesh-first approaches (like Narada), probe messages calculate the distance of the probing node to the potential neighbors. In case of MTI, the probe message should also include the distance of the potential neighbor to the source in addition to the distance information between the two nodes. This requires 4 bytes of extra information in the probing message.

- *Computational Complexity:* In case of ESM, since all multicasting activities are handled at the end systems, therefore computational complexity assumes important proportions. This is because of the computational limitations at the host. The computational complexity in case of any flat mesh-first protocol is $O(n)$, where n is the number of nodes in the mesh. This is because, each node can make a decision to include or exclude a potential neighbor in constant time. MTI increases the computational complexity to $O(flogf + fn)$, where f is the fanout limit of the nodes in the mesh. The first term comes because of the sorting of the mesh links, and the second term comes since each decision requires $O(f)$ time.

Since the message and computational complexity of MTI increases linearly, then the scalability of the protocol suffers for high number of nodes in the mesh. A message complexity of $O(n)$ has the potential of message explosion, if the number of nodes in the mesh increases. Therefore, there is a need to device means to reduce or restrict the number of messages transmitted. Reduction of message complexity motivates the development of Restricted MTI (R-MTI). In R-MTI, the potential neighbor search is only restricted to the neighbors in either SM^- and PM^-. The ρ value of neighbors of SM^+ and PM^+ are calculated based on Property 4. R-MTI helps to restrict the worst-case message complexity from $O(n)$ under normal MTI, to $O(f^2)$, where f is the fanout limit of the nodes. Under normal case, it will be still less

because search will be restricted to only neighbors having negative ρ. Similarly, the computational complexity is reduced from $O(flogf + fn)$ to $O(flogf + f^2)$. Though R-MTI has low computational and message complexity, it produces lower "quality" tree as the search space is restricted. Therefore, R-MTI introduces a trade-off between message and computational complexity with the "quality" of the multicast tree. In the simulation section R-MTI is studied vis-a-vis MTI and Narada to quantify this trade-off.

6.6 Performance Studies

In order to evaluate the effectiveness of the MTI model, extensive simulation studies using were conducted using ns [42]. In the simulation studies, the MTI model is compared with Narada as well as several Centralized algorithms. It is to be noted that the main objective of the simulation is to evaluate the relative performances of MTI, Narada, and centralized approaches, rather than to quantify the absolute performance offered by them. The various inputs for the simulation studies were generated as follows:

- Random network topologies were generated based on a given input parameter "graph density." This parameter determines the average node degree and hence the connectivity of the network. The higher the value, the denser the topology.
- The selection of receivers for a given multicast session were uniformly distributed from the node set.
- Members join and leave the multicast group, and the mesh is reorganized assuming a Distance Vector protocol running on the mesh level.
- For each point in the graph, an average of 10 simulation runs were conducted.
- The variation of the results from the actual mean is computed based on 95% confidence interval.
- *Default parameters:* (i) Total number of nodes = 1000, (ii) 20% of all nodes are end hosts, (iii) Average Node degree = 4, (iv) Average number of members = 100, (v) Average link

bandwidth = $15Mbps$, (vi) Average link delay = $12.5ms$ (vii) Member join/leave inter-arrival time = $100ms$, (viii) Average fanout of the nodes = 4.0, (viii) Time interval between heartbeat messages in the mesh = $20ms$.

In order to compare the effectiveness of the various models, we evaluated the models according to the following performance metrics:

- *Average Relative Delay Penalty (ARDP):* Relative Delay Penalty (RDP) is defined as the ratio of the delay provided by the current multicast tree to that provided by unicast averaged for all the members. Let the number of members in time interval $[j-1, j]$ is m_j. Let the current delay experienced by the member i is D_i and the delay experienced if the connection had been unicast is D_i^U. Let n be the total number of time intervals. Then ARDP is defined as:

$$ARDP = \frac{1}{n}\sum_{j=1}^{n}\frac{1}{m_j}(\sum_{i=1}^{m_j}\frac{D_i}{D_i^U}) \tag{6.2}$$

- *Average Maximum Relative Delay Penalty (AMRDP):* Maximum Relative Delay Penalty (MRDP) is defined as the maximum RDP suffered by a node, among all the nodes currently in session. Let the number of members in time interval $[j-1, j]$ is m_j. Let the current delay experienced by the member i is D_i and the delay experienced if the connection had been unicast is D_i^U. Then AMRDP is defined as:

$$AMRDP = \frac{1}{n}\sum_{j=1}^{n} Max_{1\leq i\leq m_j}\{\frac{D_i}{D_i^U}\} \tag{6.3}$$

- *Average Stress:* Stress is defined as the average number of unicast flows per tree link. Let the total number of tree links in time interval $[j-1, j]$ be l_j. Let c_i be the number of unicast connections in link i. Then the Average Stress is defined as

$$Average\ Stress = \frac{1}{n}\sum_{j=1}^{n}\frac{1}{l_j}(\sum_{i=1}^{l_j} c_i) \tag{6.4}$$

- *Average Tree Cost:* The average cost of multicast tree averaged out over time were compared to evaluate the effectiveness of the algorithms. Let the total simulation time

be divided into n equal intervals. Let the cost of the multicast tree in the time interval $[j-1, j]$ be C_j. Then the tree cost is calculated as

$$Average\ Tree\ Cost\ =\ \frac{1}{n}\sum_{j=1}^{n} C_j \tag{6.5}$$

6.6.1 Performance Studies

In order to evaluate the effectiveness of different approaches, the effects of the following parameters were studied:

- *Effect of Fanout Constraint:* Fanout constraint of each member in the multicast group was varied and its effect was studied.
- *Effect of Group Dynamics:* The rate of members joining and leaving a multicast group was varied keeping all the other parameters fixed at their default values.
- *Effect of Network Density:* Graphs (routers & end hosts) can be made dense or sparse by changing the node degree. Effect of the node degree was studied keeping all the parameters fixed.
- *Effect of Group Size:* The number of receivers in the group were varied.

6.6.2 Effect of Fanout Limit

In this set of experiments, the fanout limit of each node participating in the ESM session was varied and its effect was studied on different performance metrics for different approaches. The results shown in Figures 6.8(a) and (b) vary from the actual mean by around ±4.2% and ±4.4% respectively. While, the results shown in Figures 6.9(a) and (b) vary from the actual mean by around ±2.7% and ±4.3% respectively. In Figure 6.8(a), the average RDP (ARDP) is studied with varying fanout limit. With the increase of fanout limit, each node in the multicast group can accommodate more mesh links, and therefore reduces the average delay of the overall tree. This trend can be observed for all the approaches. Comparing the relative performances of different approaches, the ARDP of the trees constructed using MTI

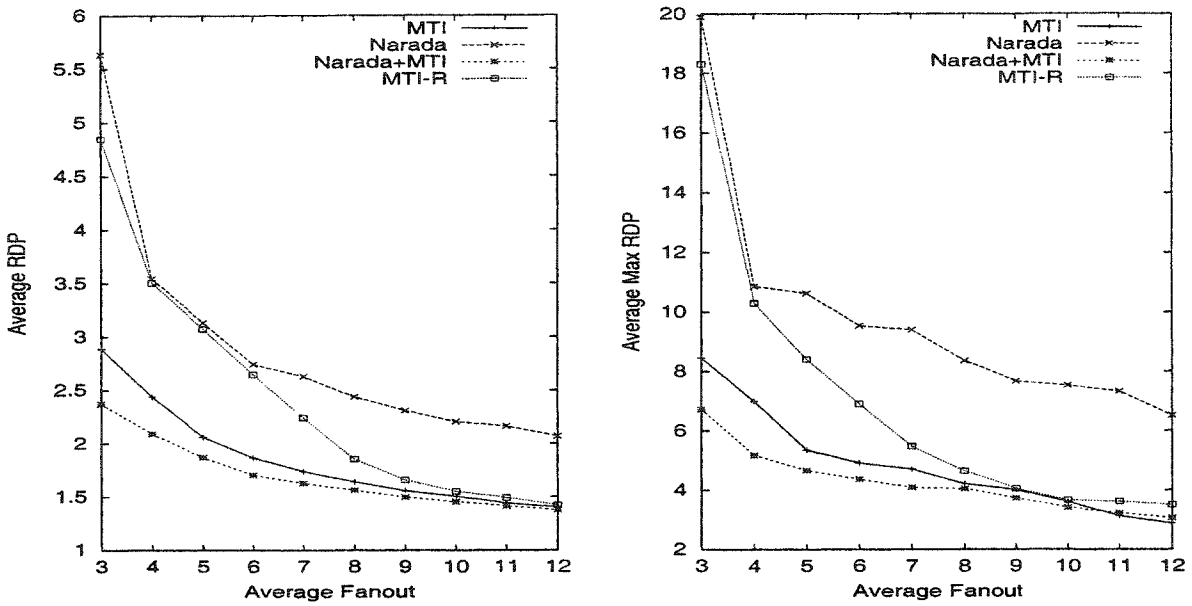


Figure 6.8 Variation of (a) ARDP and (b) AMRDP with varying fanout limit

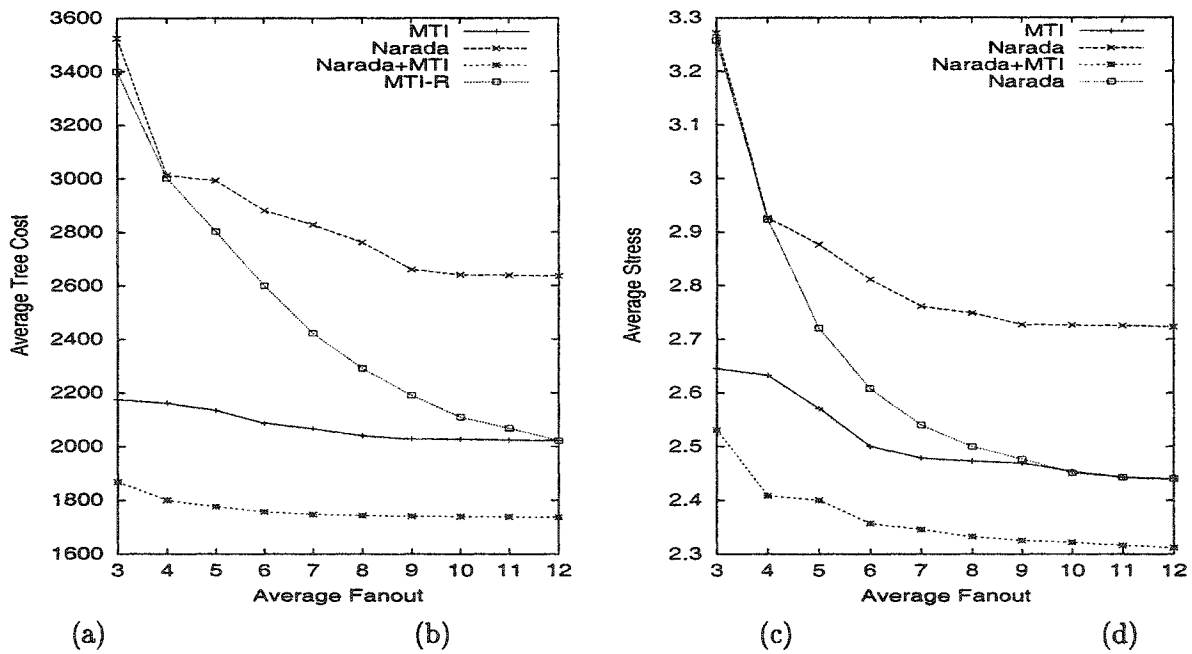
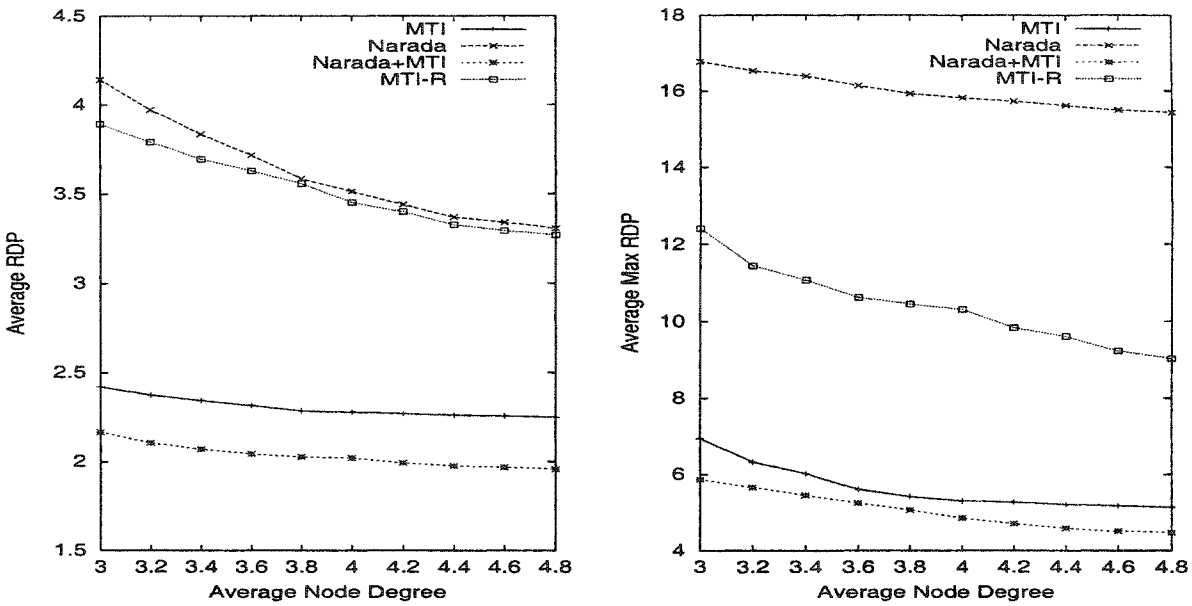
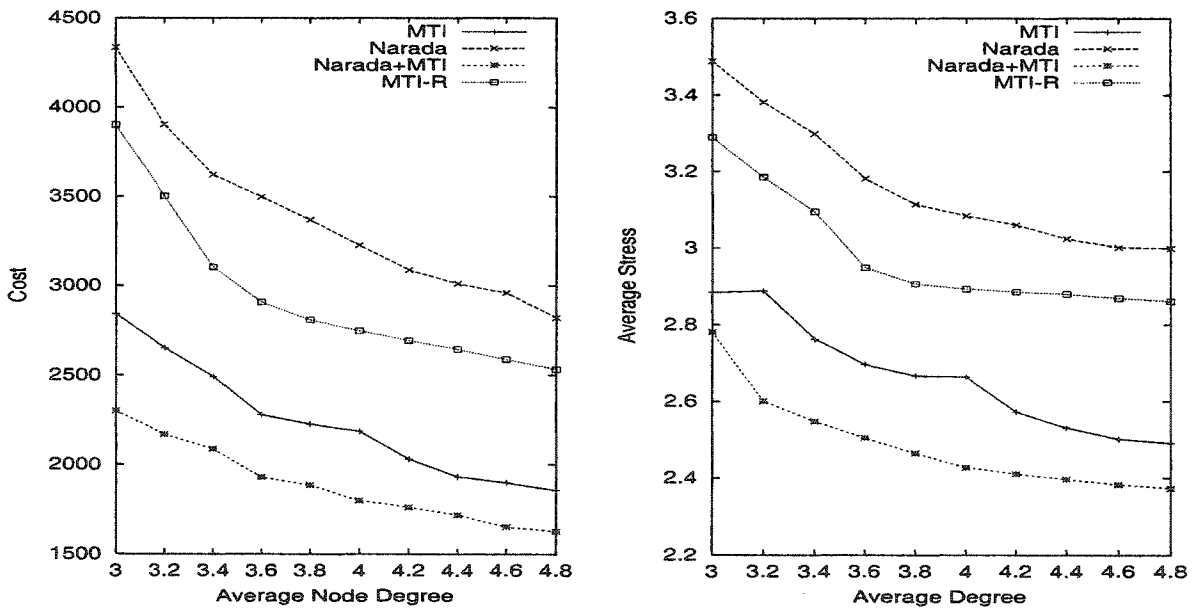


Figure 6.9 Variation of (a) average tree cost and (b) average stress with varying fanout limit



(a) (b)

Figure 6.10 Variation of (a) ARDP and (b) AMRDP with varying average network density



(a) (b)

Figure 6.11 Variation of (a) average tree cost and (b) average stress with varying average network density

is 50 – 60% of that of the Narada trees. Narada and MTI combination reduces the ARDP value further by 10 – 15%, justifying that MTI and Narada can be combined in practice to get a better quality without losing the inherent easy maintenance of the mesh-first approaches. ARDP of the MTI-R approach, is lower (better) than that of Narada and gets closer to MTI with increase in fanout limit. The reason behind this is that, with increase in fanout limit, more links nodes are searched for the identification of better upstream neighbor.

Though ARDP is the primary metric used in MTI, the unique method of selecting best neighbors using ρ value reduces AMRDP, Average Tree Cost and Stress also. This point is justified in Figures 6.9(a), and 6.9(b). The trends exhibited by each of these metrics is similar to that exhibited by the ARDP metric. Narada has the highest Average Tree Cost, MTI and Narada combination has the lowest. MTI and MTI-R lies somewhere in between. Trees produced by the MTI-R approach is similar to the MTI approach with increase in fanout limit.

6.6.3 Effect of Network Density

In this set of experiments the node degree of the physical network is varied and its effect is studied on the four different performance metrics mentioned above. Higher the average node degree, denser is the physical topology. Results shown in Figures 6.10(a), 6.10(b), 6.11(a) 6.11(b) vary from the actual means by ±4.1%, ±4.6%, ±2.9% and ±4.2% respectively. Increasing the density of the nodes in the physical network has significant effect on the trees created on the overlay. Figures 6.10 and 6.11 illustrate the effect. As the network becomes denser, the chances of two paths on the overlay going through the same physical links gets substantially reduced. This effect gets reflected the Average Stress metric (shown in Figure 6.11(b)), and in turn on all the other metrics used for performance analysis. Average Stress decreases with increasing node degree.

Figure 6.10(a) shows the variation of ARDP with average node degree of the physical network. As the network becomes denser, then there are more options to go from one node to another, as a result the chances of getting a better delay path increases. This phenomenon is reflected in Figure 6.10(a). Among the different approaches, combination of MTI and Narada

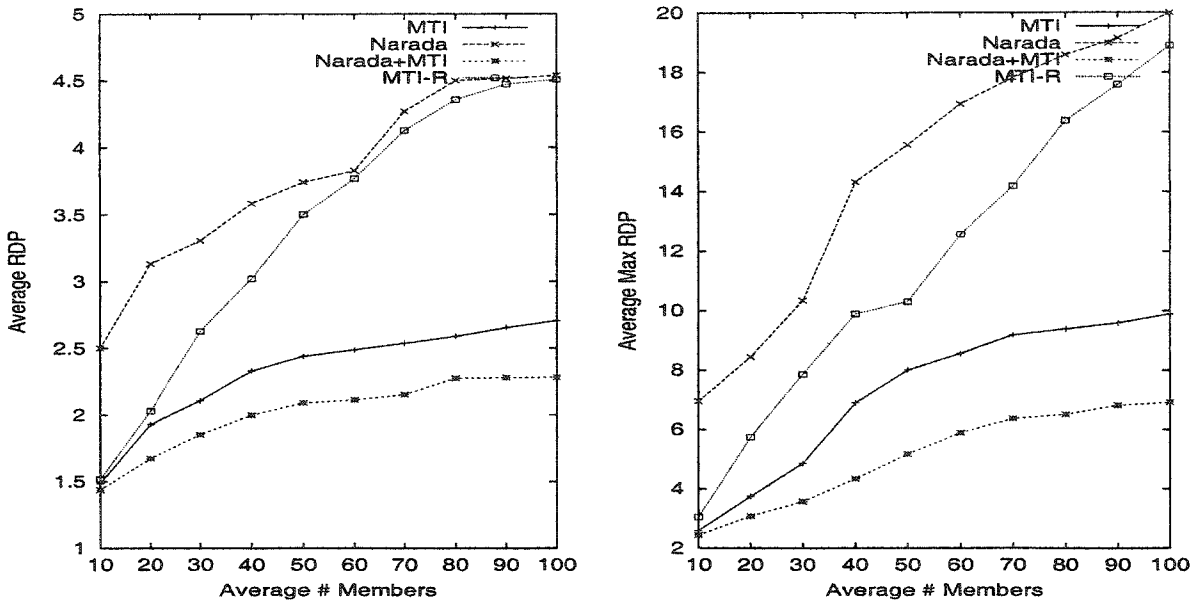


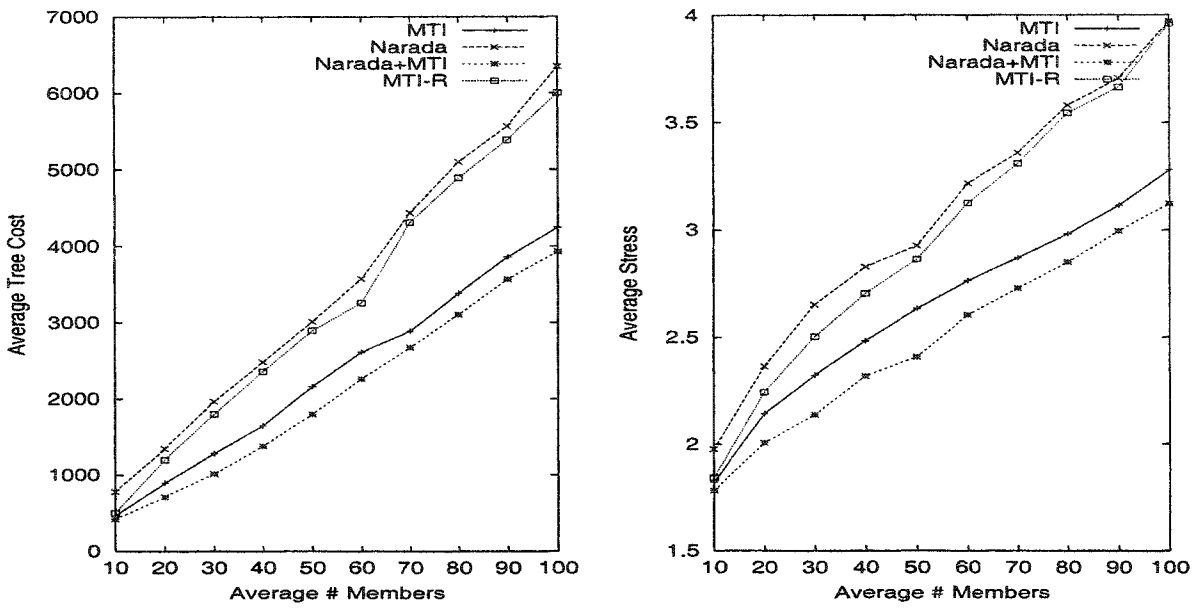
Figure 6.12 Variation of (a) ARDP and (b) AMRDP with varying average group size

has the lowest ARDP value. ARDP value in case of MTI is nearly 50% less than that of Narada. R-MTI is in between Narada and MTI and ARDP of most R-MTI trees are around 10% lower than that of Narada, while 20 – 30% more than MTI. Similar trends are also witnessed for AMRDP, Average tree cost and Average Stress metrics.

6.6.4 Effect of Group Size

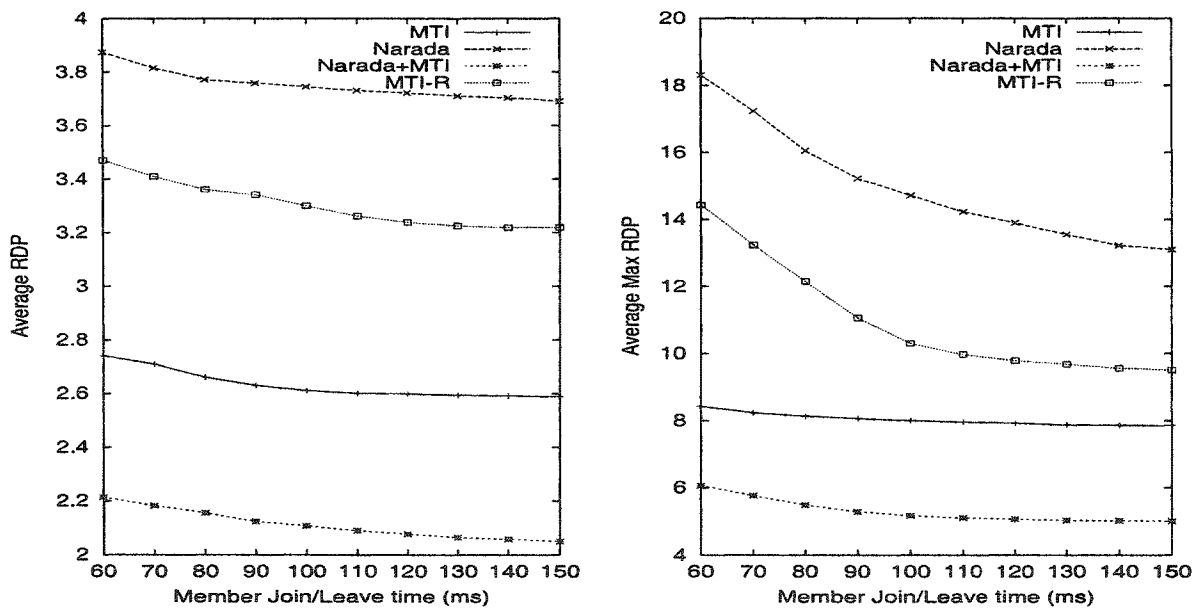
In this set of experiments, the average group size of a multicast session is increased. The results are shown in Figures 6.12 and 6.13. Results shown in Figures 6.12(a), 6.12(b), 6.13(a) 6.13(b) vary from the actual means by ±4.1%, ±4.4%, ±2.6% and ±3.9% respectively. As the group size increases the size of the overlay increases. Therefore, the average cost of the multicast tree increases, as more members are part of the tree. Delay and stress metrics also show an increase, as more members join the group which may be farther away from the rest of the tree, resulting in increase in these performance metrics.

In Figure 6.12(a), the variation of ARDP metrics is shown with average group size. At



(a) (b)

Figure 6.13 Variation of (a) average tree cost and (b) average stress with varying average group size



(a) (b)

Figure 6.14 Variation of (a) ARDP and (b) AMRDP with varying average join/leave time

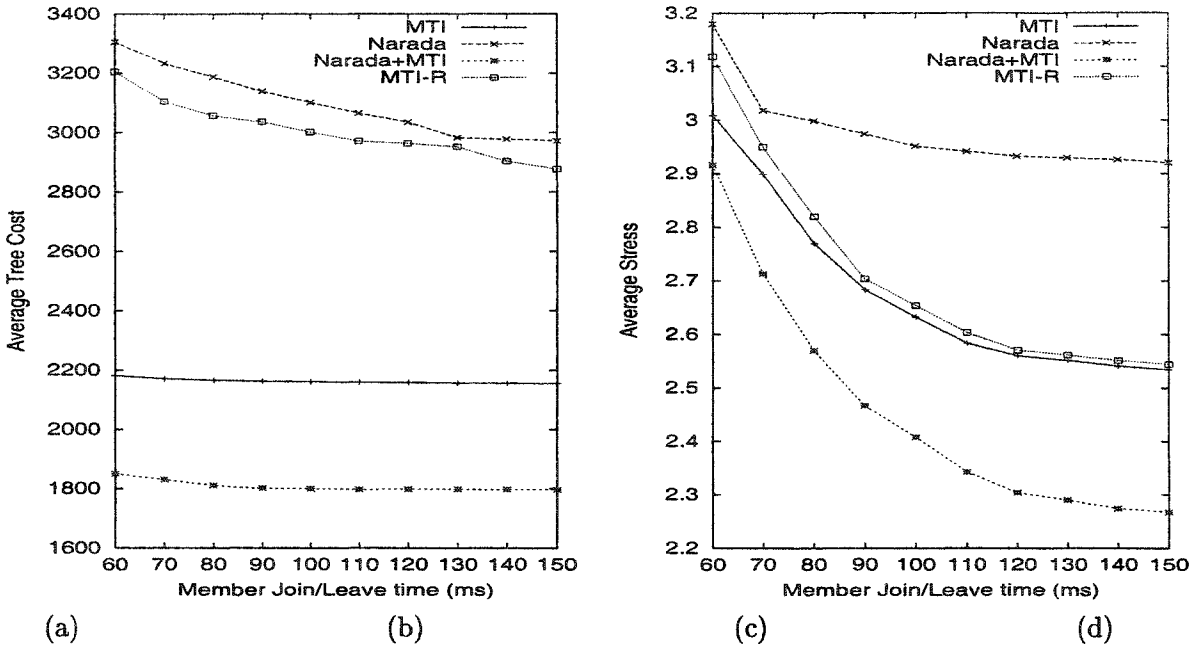


Figure 6.15 Variation of (a) average tree cost and (b) average stress with varying average join/leave time

low group size, ($\leq$ 60), R-MTI performs similar to MTI and its ARDP is nearly 50% less than that of Narada. The combination is 10% less than MTI. With increase in group size, the ARDP value increases for all approaches as the distance between nodes increases. The increase of R-MTI is maximum, as the ARDP value is nearly equal to that of Narada for group size $\geq$ 160. Increase of Narada is approximately linear with group size, while that of MTI and the combination is sub-linear. Therefore, ARDP of MTI is approximately 80% less than Narada at higher values of group size ($\geq$ 160). Variation of AMRDP is shown in Figure 6.12(b), which is similar to RDP.

In Figure 6.13(a) and (b), the variation of Average Tree Cost and Stress is shown with average group size. Both the parameters increase approximately linearly with average group size. The relative performance of the approaches match that of ARDP.

6.6.5 Effect of Group Dynamics

In Figures 6.14 and 6.15 variations of group dynamics are shown on the performance metrics. Results shown in Figures 6.14(a), 6.14(b), 6.15(a) 6.15(b) vary from the actual means by ±4.2%, ±4.3%, ±2.5% and ±4% respectively. Group dynamics is measured by the join/leave time. Higher the join/leave time interval, lesser dynamics is the group. All the performance metrics increase with the increase in the group dynamics. The reason for this is that, the multicast tree and the mesh size gets bigger at a faster rate than the optimization when the group dynamics is very high.

MTI is more immune to group dynamics, as the ARDP, Average Cost and AMRDP increase approximately $3 - 5\%$ for MTI, while $5 - 7\%$ for Narada.

6.7 Summary

In this chapter, a mesh-tree interaction (MTI) approach was proposed which does not compromise on the inherent simplicity in management of the mesh first approaches, however builds a much better quality multicast tree than the mesh-first approaches. The main principle behind the MTI approach is that the "quality" of the mesh is improved based on the underlying multicast tree, which in turn improves the quality of the tree itself. Thus, by keeping the mesh structure, the management simplicity of mesh-first approaches is maintained, and the iterative tree-building mechanism improves the quality of the tree dramatically. In addition, extensive simulation studies were carried out illustrating the MTI approach. In comparison to other mesh-first approaches like the Narada, MTI improves the ARDP metric by nearly $40 - 50\%$ and cost of the multicast tree improves by $20 - 30\%$ for lower fanout constraints (4-5). MTI can also be applied in conjunction with other mesh management techniques like Narada, which further improves ARDP by $10 - 15\%$.

CHAPTER 7 Conclusions and Future Work

The proliferation and increasing importance of QoS-aware group applications coupled with the advancement in high-speed networking are driving the need for scalable and deployable group communication architectures, algorithms, and protocols over the Internet. Multicasting has been the most popular mechanism for supporting group communication. During a life-cycle of a multicast session, three important events can occur: membership dynamics, network dynamics, and traffic dynamics. Membership dynamics is concerned with maintaining a good quality (cost) multicast tree taking into account dynamic join/leave of members, network dynamics is concerned with link/node failures/additions, and traffic dynamics is concerned with flow, congestion and error control. Though all the three issues are equally important, the first two are the least researched and hence require research attention. The focus of this dissertation is to develop comprehensive set of techniques that manage group dynamics in QoS multicasting. Specifically, the dissertation makes the following key contributions:

1. *Protocols for tree maintenance:* In a highly dynamic group multicast trees 'degenerate' over time. Tree rearrangement techniques maintain a part of the multicast tree. Another way of tree maintenance involves reconstructing the multicast tree and moving the members from the old tree to the new one. This type of tree maintenance reduces the cost of the multicast tree by incurring huge amounts of service disruptions to the members. Therefore there exists a trade-off between tree cost and service disruption. The first part of the research develops scalable and efficient protocols for tree maintenance that achieve a balance between tree cost and service disruption. While tree migration is more abrupt, tree evolution is an adaptive protocol where the adaptivity is tuned based on "evolution timer". Simulation and analytical studies have been carried out to compare the effec-

tiveness of tree migration, tree evolution and no-migration approaches. The simulation studies show that evolution protocol achieves a balance between service disruption and tree cost.

2. *QoS and Reliability Constrained Multicast Routing:* Development of efficient protocols which establish multicast sessions based on QoS constraints and can handle node/link failures is an important problem. Robustness of a path to the user can be measured by the reliability of the path. Since service disruption depends on the reliability of the path, therefore it is important to take reliability into account during multicast member join. As part of the second step, a reliability constrained multicast routing protocol is developed. An approach called Partial Protection Approach (PPA) is developed which provides protection partially to certain domains. In addition, three schemes have been proposed to implement the PPA approach. Extensive simulation studies on the proposed schemes show that the schemes are scalable and the relative performance of the schemes depend on the network and group parameters.

3. *Tree maintenance in End System Multicasting:* End System Multicasting (ESM) is fast becoming an alternative to QoS multicasting. ESM tries to reduce some of the disadvantages of the IP multicasting architecture. Hence, it provides a trade-off between performance and deployability. End system multicasting offers different sets of challenges to handle group dynamics. Therefore the protocols developed for IP multicasting cannot be trivially extended to ESM. As part of the third contribution of the dissertation, a tree maintenance technique called the Mesh Tree Interaction (MTI) have been proposed for ESM. The performance of the proposed technique is compared with existing solutions like Narada, and the results show that MTI offers better performance than Narada without sacrificing the scalability.

As part of the future work, the following research directions can be followed:

- More studies need to be performed for the integrated tree maintenance approach (proposed in Chapter 4). The integrated framework can be compared with the migration and

evolution models, and performance of the framework can be made tunable to different network and group parameters.

- Reliability constrained multicast routing has been studied in this dissertation taking only one backup path into consideration. However, the reliability constraint of the request may not be satisfied by one backup path only. The problem will then be modified to optimizing the number of paths in each domain so that the overall reliability constraint is satisfied. In this case, the Domain Selection Problem will be more difficult, as the solution should not only select the domains but also should select the number of backup paths in that domain.

- The partial protection approach proposed in Chapter 5 can be extended as a mechanism for network provisioning where reliability is used as a network design parameter. Network provisioning typically works offline and the methods can be developed based on the concepts outlined in this dissertation to develop such schemes.

- In End System Multicasting, if the receivers join and leave the group continuously, the mesh will be reconstructed every time, resulting in service disruption to the existing members. This problem can be used as a tool for Denial-of-Service attack on the multicast group, by malicious users. Therefore, efforts must be undertaken to prevent this type of attacks. One way to tackle the problem is to allow the members to slowly move up the multicast tree based on their longevity in the group. Another interesting approach to solve the problem would be to divide the multicast groups into Virtual Sources (VS) or trusted nodes, and create the tree so that the path to the VS for any node is never greater than some threshold.

- In End System Multicasting framework, management of group dynamics in a hierarchical setting offers new and interesting challenges. The hierarchical setup can be used because of maintenance ease [37] or to address the problem mentioned in the previous point through the introduction of Virtual Sources. An extension of MTI can also be construed in such a setup.

- Tree maintenance in DiffServ multicasting can also be an interesting future problem. In a DiffServ network, only the edge routers have state information and all the core routers are only used for forwarding. Therefore, it would be challenging problem to maintain the multicast group without introducing any complexity inside the core routers.

APPENDIX A Lemmas of Chapter 4

Lemma A.1: If X_1, X_2, ... X_n be n independent uniform random variables ranging between 0 and T and X_{max} is their maximum, then $\overline{X}_{max} = \frac{nT}{(n+1)}$ and $\sigma^2_{X_{max}} = \frac{nT^2}{(n+1)^2(n+2)}$

Proof: Let $f_x(X_i)$ and $F_x(X_i)$ be the pdf and cdf of the random variable X_i respectively. Since, they are iids, therefore,

$$f_x(X_i) = f_x(X) \;\; \forall i \in 1, 2, ...n; \quad F_x(X_i) = F_x(X) \;\; \forall i \in 1, 2, ...n$$

$$F_x(X_{max}) = Prob(X_{max} < x) = F(X_1 < x, X_2 < x ... X_n < x)$$

$$= F_x(X_1).F_x(X_2)....F_x(X_n) = (F_x(X))^n$$

$$\Rightarrow \;\; f_x(X_{max}) = nF_x(X)^{(n-1)} f_x(X) = \frac{nx^{n-1}}{T^n} \tag{A.1}$$

$$\Rightarrow \;\; \overline{X}_{max} = E(X_{max}) = \int_0^T x.f_x(X_{max})dx = \frac{nT}{n+1} \tag{A.2}$$

$$\Rightarrow \;\; \sigma^2_{X_{max}} = E(X^2_{max}) - E^2(X_{max}) = \int_0^T x^2.f_x(X_{max})dx \; - \; \frac{n^2T^2}{(n+1)^2} = \frac{nT^2}{(n+1)^2(n+2)} \tag{A.3}$$

Equations A.2 and A.3 prove *Lemma A.1.*

Corollary A.1: $\lim_{n\to\infty} \sigma^2_{X_{max}} = 0$

Proof: This can be proved by using limit on Equation A.3.

Definition: A queuing system is defined as a Core Queuing system (Q_C), if and only if:

- The inter-arrival time of customers is exponentially distributed with mean λ.
- There are infinite servers.

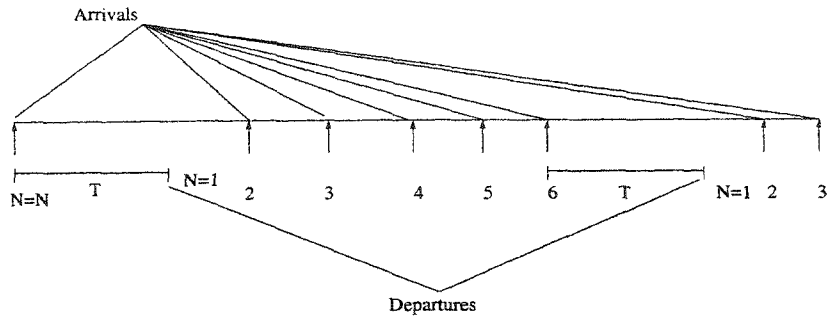


Figure A.1 A core queuing system

- The servers wait for a time T as soon as an arrival occurs.
- If there are N customers in the beginning of interval $[0, T]$, then exactly $(N-1)$ customers will get served at time T.

A Q_C queuing system is shown in the Figure A.1. Arrivals occur at instances shown by arrows. At every arrival, the number of customers increases by 1, and at every departure, the number of customers reduces to 1.

Lemma A.2: Under Q_C queuing system, the mean number of customers $(\overline{N})$ in the queue is given by $\overline{N} = e^{\lambda T}$.

Proof: Since the inter-arrival times (t) are exponentially distributed, therefore,

$$P(t > T) = 1 - P(t \leq T) = 1 - (1 - e^{-\lambda T}) = e^{-\lambda T} = p \tag{A.4}$$

Let $N(i)$ denote the number of customers at the end of interval i. Therefore,

$$N(i) = N(i) + 1 \quad \text{if } t(i) < T$$
$$N(i) = 1 \qquad\qquad \text{otherwise}$$

Let $\overline{N}$ be the expected number of customers in the system. Therefore,

$$\overline{N} = \sum_{n=1}^{\infty} np.(1-p)^{n-1} = 1/p = e^{\lambda T} \tag{A.5}$$

Equation A.5 prove the lemma.

Lemma A.3: Let X and Y are independent random variables. X follows a uniform $(0,\ T)$ distribution and Y follows an exponential (λ) distribution, then $P(X < Y) = \frac{(1-e^{-\lambda T})}{\lambda T}$.

Proof: X follows a uniform $(0, T)$ distribution. Therefore,

$$f_X(x) = 1/T \quad 0 < x < T \tag{A.6}$$

Y follows an exponential (λ) distribution. Therefore,

$$f_Y(y) = \lambda e^{-\lambda T} \quad y > 0 \tag{A.7}$$

Using the above equations and the property of independence,

$$P(Y > X) = \int_0^T \int_x^\infty f_{XY}(xy)dydx = \int_0^T \int_x^\infty f_X(x).f_Y(y)dydx = \frac{(1 - e^{-\lambda T})}{\lambda T} \tag{A.8}$$

Lemma A.4: Let X and Y be similar to *Lemma 3.* If N independent Bernoulli trials are performed such that $Prob(Success) = P(X < Y)$, then expected number of successes in N trials equal $\frac{N(1-e^{-\lambda T})}{\lambda T}$.

Proof: Expected number of successes in case of N independent Bernoulli experiment is Np, where p is the probability of success. Lemma can be proved by substituting the value of p from Equation A.8.

Lemma A.5: Let $Y = \frac{N(1-e^{-\rho})\rho}{N(1-e^{-\rho})+\rho}$, $Y_1 = \frac{N\rho}{N+1}$, and $Y_2 = \frac{N\rho}{N+\rho}$. Y can be approximated by Y_1 as $\rho \to 0$, and by Y_2 as $\rho \to \infty$, respectively.

Proof (A):

$$Y = Y = \frac{N(1 - e^{-\rho})\rho}{N(1 - e^{-\rho}) + \rho} = \frac{N}{N + \frac{\rho}{1-e^{-\rho}}} \tag{A.9}$$

In the above Equation A.9, $\frac{\rho}{1-e^{-\rho}} \to 1$ as $\rho \to 0$. Therefore, Y can be approximated by Y_1 as $\rho \to 0$.

Proof (B): $(1 - e^{-\rho}) \to 1$ as $\rho \to \infty$. Therefore, Y can be approximated by Y_2 as $\rho \to \infty$.

APPENDIX B Lemmas of Chapter 5

Lemma B.1: RCLCR is NP-hard.

Proof: Let $N = (V, E)$ be a network. Each link $e \in E$ has a three-tuple $< c_i, d_i, p_i >$, where $c_i \geq 0$, $d_i \geq 0$, $0 \leq p_i \leq 1$. RCLCR is defined as:

$$\text{minimize } \sum_{\forall i \in V} c_i$$

$$\text{subject to } \prod_{\forall i \in V} p_i \geq P$$

$$\text{where } 0 \leq p_i \leq 1 \text{ and } c_i \geq 0$$

Delay Constrained Least Cost Routing problem (DCLCR), which is known to be NP-hard, can be stated mathematically as follows:

$$\text{minimize } \sum_{\forall i \in V} c_i$$

$$\text{subject to } \sum_{\forall i \in V} d_i \leq D$$

$$\text{where } d_i \geq 0 \text{ and } c_i \geq 0$$

An instance of DCLCR can be reduced to an instance of RCLCR by making the following transformation: $p_i = e^{-d_i}$ and $P = e^{-D}$. The transformation can be made in polynomial time. An instance of RCLCR can also be reduced to DCLCR in polynomial time by setting $d_i = log(1/p_i)$ and $D = log(1/P)$. Since DCLCR is NP-hard. Therefore, RCLCR is also NP-hard.

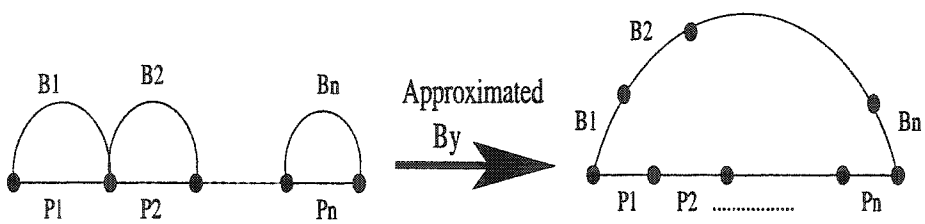


Figure B.1 Approximation due to the online algorithm

Lemma B.2: Reliability of a serial connection of parallel paths is always greater than that of a parallel connection of serial paths, for a homogeneous network.

Proof: In the Figure B.1, serial connections of parallel paths and parallel connections of serial paths are shown. It is to be proved that the approximation is valid under the conditions mentioned in the Lemma. Let the Reliability of the path P_j be r_{p_j} and the reliability of path B_i be r_{b_i}. Since it is a homogeneous network, $r_{b_i} = r_{p_j} = R \ \ \forall i = 1, 2, 3...n, \ j = 1, 2, ...n.$ Then the reliability of the serial connection of parallel paths is given by: $\prod_1^n (r_{b_i} || r_{p_i})$, where $x||y = x + y \times (1 - x)$. Therefore, the reliability becomes $R_{sp} = (2R - R^2)^n = R^n(2 - R)^n$. The reliability of the parallel connection of the serial paths is given by $(\prod_1^n r_{b_i})||(\prod_1^n r_{p_i})$. Therefore, the reliability becomes $R_{ps} = (2R^n - R^{2n}) = R^n(2 - R^n)$.

Therefore,

$$\alpha = \frac{R_{sp}}{R_{ps}} = \frac{(2 - R)^n}{2 - R^n} \tag{B.1}$$

Differentiating Equation B.1, it follows

$$\frac{d\alpha}{dR} = \frac{2n(2 - R)^{n-1}(R^{n-1} - 1)}{(2 - R^n)^2} \tag{B.2}$$

Extremum of α is obtained at $\frac{d\alpha}{dR} = 0$, *i.e.* $R = 1$. Differentiating Equation B.2 at $R = 1$, it follows

$$\frac{d^2\alpha}{dR^2}|_{R=1} = 2n(n - 1) > 0, \ \ when \ \ n > 1 \tag{B.3}$$

Equation B.3 proves the Lemma.

Lemma B.3: Domain Selection Problem (DSP) is NP-hard.

Proof: To prove that DSP is NP-hard DSP s formally defined as follows:

$$\text{minimize } \sum_1^n c_i \times x_i$$

$$\text{subject to } \prod_1^n w_i^{x_i} \leq W \text{ and } x_i \in \{0, 1\}$$

$$\text{where } w_i \geq 1 \text{ and } c_i > 0$$

0-1 Knapsack problem, which is a well known NP-hard problem, can be stated formally as

follows:

$$\text{minimize } \sum_1^n c_i \times x_i$$

$$\text{subject to } \sum_1^n a_i \times x_i \leq B \text{ and } x_i \in \{0,1\}$$

$$\text{where } a_i \geq 0 \text{ and } c_i > 0$$

An instance of 0-1 Knapsack problem can be reduced to an instance of DSP by setting: $w_i = e^{a_i}$ and $W = e^B$. This transformation can take place in polynomial time. Similarly, an instance of DSP can be reduced to an instance of 0-1 Knapsack problem in polynomial time by setting $a_i = log(w_i)$ and $B = log(W)$. Since, 0-1 Knapsack optimization problem is known to be NP-hard, therefore DSP is also NP-hard.

APPENDIX C Properties of Chapter 6

Property 1: $-1 \leq \rho \leq 1$

Proof: It is assumed that $\rho_{ij}^{s} > 0$. Then, from Equation 6.1 it is obtained,

$$\Delta_{is} > \Delta_{ij} + \Delta_{js} \tag{C.1}$$

Equation C.1 shows that there is an alternate path through j which is shorter than the shortest path $i - s$. This leads to contradiction, therefore

$$\rho_{ij}^{s} \leq 1 \tag{C.2}$$

To prove the lower bound, it is assumed that $\rho_{ij}^{s} < -1$. Therefore, from Equation 6.1, it is obtained

$$\Delta_{js} > \Delta_{ij} + \Delta_{is} \tag{C.3}$$

Since the network is undirected, $\Delta_{ij} = \Delta_{ji}$. Therefore, Equation C.3 leads to a contradiction as there exists a shorter path than the shortest from j to s through i. Therefore,

$$\rho_{ij}^{s} \geq -1 \tag{C.4}$$

Equations C.2 and C.4 prove the property.

Property 2: If $\rho_{ij}^{s} = \delta$, then $\rho_{ji}^{s} = -\delta$.

Proof: From Equation 6.1,

$$\rho_{ij}^{s} = \frac{\Delta_{js} - \Delta_{is}}{\Delta_{ji}} = -(\frac{\Delta_{is} - \Delta_{js}}{\Delta_{ij}}) = -\rho_{ij}^{s} \tag{C.5}$$

Equation C.5 proves the Property.

Property 3: $\rho_{is}^{s} = 1$.

Proof: The property can be proved by substituting $\Delta_{ss} = 0$ in Equation 6.1.

Property 4: Let i, j and k are three nodes and source is s, $\rho_{ij}^s > 0$ and $\rho_{jk}^s > 0$, then $\rho_{ik} \geq \frac{\rho_{ij}^s \times \Delta_{ij} + \rho_{jk}^s \times \Delta_{jk}}{\Delta_{ij} + \Delta_{jk}}$.

Proof: From Equation 6.1,

$$\rho_{ik} = \frac{\Delta_{is} - \Delta_{ks}}{\Delta_{ik}} \geq \frac{\Delta_{is} - \Delta_{ks}}{\Delta_{ij} + \Delta_{jk}} \tag{C.6}$$

The Property can be proved by substituting the values of Δ_{is} and Δ_{js} from Equation 6.1 to Equation C.6.

Property 5: If $i_j = \eta_{i_{j-1}}^s \ \forall j = 1, 2...n$, and $\rho_{i_{j-1}, i_j}^s > 0$, then i_1, i_2 ... i_n cannot form a loop.

Proof: It is assumed that i, j and k are nodes such that $j = \eta_i^s$, $k = \eta_j^s$ and $i = \eta_k^s$ *i.e.*, i,j and k form a loop. Also, $\rho_{ij}^s, \rho_{jk}^s, \rho_{ki}^s > 0$. From Equation 1, it is obtained that $\Delta_{is} > \Delta_{js}$,$\Delta_{js} > \Delta_{ks}$ and $\Delta_{ks} > \Delta_{is}$. This leads to contradiction, therefore such a loop cannot occur. This argument can be extended to prove the Property $\forall \ k = i_1, i_2...i_n$

APPENDIX D Details of Algorithm for Mesh Expansion & Contraction

1. If Current Fanout of i is less than the fanout limit of i goto 9. Therefore, if i can accommodate the link it will as long as n can also accommodate the link.

2. If SM^+ of i is empty goto 4. To accommodate the link, some link of i need to be removed, since SM^+ is empty, links from SM^- are searched.

3. If $\rho_{in} > \rho_{\gamma^+_{SM}}$

   (a) This means that link $(i-n)$ is "better" than at least one link in, SM^+

   (b) Set $ReplaceLink_i = \gamma_{SM+}$ *i.e.* γ_{SM+} is chosen as the likely candidate for removal from the mesh.

   (c) Goto 9

4. Otherwise Goto 18 *i.e* current link is "good" enough, therefore the link is not added to the mesh.

5. If SM^- of i is empty Goto 7 *i.e.* SM is empty, therefore PM is searched.

6. If SM^- of i is non-empty

   (a) set $ReplaceLink_i = \gamma_{SM-}$ *i.e.* γ_{SM-} is a likely candidate for removal.

   (b) Goto 9 *i.e.* check whether n can accommodate the link.

7. If $\rho_{in} > \rho_{\gamma^+_{PM}}$

   (a) Current Link is better than the link in PM^+, therefore PM^+ is the likely candidate for removal.

(b) Set $ReplaceLink_i = \gamma_{PM+}$ *i.e.* γ_{PM+} is a likely candidate of removal from the mesh.

(c) Goto 9

8. Otherwise Goto 18, *i.e.* no candidate can be found. Therefore, the link is not "good" enough.

9. If current fanout of n is less than the fanout limit of n Goto 17 *i.e.* the link can be added without removing any current link in n.

10. If n is a source node Goto 17.

11. If SM^- of n is empty Goto 13 *i.e.* n is not a source and SM^+ for candidates.

12. If $\rho_{in} < \rho_{\gamma_{SM}^-}$

(a) Current link is "better" than at least one link in SM^- of n.

(b) set $ReplaceLink_n = \gamma_{SM-}$ *i.e.* γ_{SM-} is the likely candidate for removal.

(c) Goto 18

13. If SM^+ of n is empty Goto 15

14. If SM^+ of n is non-empty

(a) set $ReplaceLink_n = \gamma_{SM+}$ *i.e.* γ_{SM+} is a likely candidate of removal from the mesh.

(b) Goto 18

15. If $\rho_{in} < \rho_{\gamma_{PM}^-}$

(a) Current link is "better" than at least one link in PM^- of n, therefore γ_{PM}^- is the likely candidate candidate for removal.

(b) set $ReplaceLink_n = \gamma_{PM-}$ *i.e.* γ_{PM-} is a likely candidate of removal from the mesh.

(c) Goto 18

16. Otherwise Goto 19

17. If $\Delta_{in} < \Delta_{jn}$, $j \in PM^+$

(a) set $ReplaceLink_n = j$ *i.e.* j is a likely candidate of removal from the mesh. Here Δ is used as a parameter instead of ρ because, in this case $\rho_{in} = 1$ (Property 3).

(b) Goto 18

18. The current link is added

(a) Link $(i - n)$ is added to the secondary mesh

(b) Links $ReplaceLink_i$ and $ReplaceLink_n$ are removed from the mesh of i and n respectively.

19. Exit

Bibliography

[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] C. Diot, B.N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for the IP multicast service and architecture," *IEEE Network,* pp.78-88, Jan./Feb. 2000.

[3] J. Hou and B. Wang, "Multicast routing and its QoS extension: Problems, algorithms and Protocols," *IEEE Networks,* Jan./Feb. 2000.

[4] L. Sahasrabuddhe and B. Mukherjee, "Multicast routing algorithms and protocols: A tutorial," *IEEE Network,* Jan./Feb. 2000.

[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] D.R. Cheriton and S. Deering, "Host groups: A multicast extension for datagram internetworks," in *Proc. Data Communications Symposium,* pp.172-179, 1985.

[7] A. Striegel and G. Manimaran, "A survey of QoS multicasting issues," in *IEEE Communications,* vol.40, no.6, pp.82-87, June 2002.

[8] J.C. Pasquale, G.C. Polyzos, and G. Xylomenos, "The multimedia multicasting problem," *Multimedia Systems,* vol.6, no.1, pp.43-59, 1998.

[9] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," *Acta Informatica,* vol.15, no.2, pp.141-145, 1981.

[10] H. Takashami and A. Matsuyama, "An approximate solution for the Steiner tree problem in graphs," in *Intl. J. Math Educ. in Sci. and Technol.*, vol.14, no.1, pp.15-23, 1983.

[11] M. Handley, H. Schulzrinne, E. Schooler, J.Rosenberg, "Session Initiation Protocol (SIP)", *Proposed Internet Standard, RFC 2543, Internet Engineering Task Force (IETF) Multi-party Multimedia Session Control (MMusic) Working Group,* Mar. 1999.

[12] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A New Resource Reservation Protocol,", in *IEEE Network*, September 1993.

[13] S. Deering, C. Partridge, and D. Waitzmann, "Distance vector multicast routing protocol," *RFC 1075,* Nov. 1988.

[14] J. Moy, "Multicast Extensions to OSPF", *RFC 1584*, Mar. 1994.

[15] 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.

[16] Hugh. W. Holbrook, and David.R. Cheriton, "IP Multicast Channels: Express Support for Large Scale Single Source", *Proc. ACM SIGCOMM,* Sept. 1999.

[17] 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.

[18] D.G. Thaler and C.V. Ravishankar, "Distributed center location algorithms," *IEEE JSAC,* vol.15, no.3, pp.291-303, Apr. 1997.

[19] M. Parsa and J.J. Garcia-Luna-Aceves, "A protocol for scalable loop-free multicasting," in *IEEE JSAC,* vol.15, no.3, April 1997.

[20] C. Donahoo and Zegura, "Core selection methods for multicast routing," in *Proc. ICCCN,* 1995.

[21] 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.

[22] C. Shields and J.J. Garica-Luna-Aceves, "The Ordered Core Based Tree Protocol," in *Proc. INFOCOM,* Apr. 1997.

[23] L. Wei and D. Estrin, "The trade-offs of multicast trees and algorithms," in *Proc. ICCCN,* pp. 12-14, Sept. 1994.

[24] C. Donahoo and Zegura, "Core migration for dynamic multicast routing," in *Proc. ICCCN,* 1996.

[25] B. Waxman, "Routing of multipoint connections," *IEEE JSAC,* vol. 6, pp.1617-1622, Dec. 1988.

[26] F. Bauer and A. Verma, "ARIES: A rearrangeable inexpensive edge-based on-line Steiner algorithm," *IEEE JSAC,* vol. 15, pp.382-397, Apr. 1997.

[27] R. Sriram, G. Manimaran, C. Siva Ram Murthy, "A rearrangeable algorithm for the construction of delay-constrained dynamic multicast trees," in *IEEE/ACM Trans. Networking,* vol.7, no.4, pp.514-529, Aug. 1999.

[28] M. Faloutsos, A. Banerjea, and R. Pankaj, "QoSMIC: Quality of service sensitive multicast internet protocol", in *Proc. ACM SIGCOMM,* Sept. 1998.

[29] Shigang Chen, Klara Nahrstedt, and Yuval Shavitt, "A QoS-Aware Multicast Routing Protocol", *IEEE JSAC,* vol.18, no.12, Dec. 2000.

[30] G. Manimaran, H. S. Rahul, and C. S. R. Murthy, "A New Distributed Route Selection Approach for Channel Establishment in Real Time Networks," in *IEEE/ACM Trans. of Networking,* vol. 7, no. 4, pp. 514-529, Aug. 99.

[31] A. Chakrabarti, G. Manimaran, "A Case for Scalable Multicast Tree Migration," in *Proc. IEEE Globecom,* 2001.

[32] K.Carlberg, "QoS Multicast using single metric Unicast Routing", PhD thesis, University College London, Oct. 1999.

[33] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *Proc. IFIP/ACM ICDSP,* pp. 329-350, Nov. 2001.

[34] F. Dabek, E. Brunnskill, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica, and H. Balakrishnan, "Building Peer-to-Peer Systems with Chord, a Distributed Lookup Service," in *Proc. HOTOS VIII,* May 2001.

[35] D. G. Anderson, H. Balakrishnan, M. F. Kaashoek, and R. Morris, in *ACM SOSP,* Oct. 2001.

[36] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "Enabling Conferencing Applications on the Internet using an Overlay Multicasting Architecture," in *Proc. SIGCOMM,* Aug. 2001.

[37] S. Banerjee, B. Bhattacharya, and C. Kommareddy, "Scalable Application Layer Multicast," in *Proc. SIGCOMM,* Aug. 2002.

[38] D. Pendarakis, S. Shi, D. Verma, and M. Waldgovel, "ALMI: An Application Level Multicast Infrastructure," in *Proc. USENIX Symp. on Internet Technologies and Systems,* Mar. 2001.

[39] M. Castro, P. Druschel, A. -M. Kermarrec, and A. Rowstron, "SCRIBE: A Large-scale and decentralized application-level multicast infrastructure," in *IEEE JSAC,* vol. 20, no. 8, Oct. 2002.

[40] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharya, and S. Khuller, "Construction of an Efficient Overlay Multicast Infrastructure for Real-Time Applications," in *Proc. INFOCOM,* Apr. 2003.

[41] L. Schwiebert and R. Chintalapati, "Improved fault recovery for core based trees," in *Computer Communications,* vol.23, no.9, Apr. 2000.

[42] UCB/LBNL/VINT Network Simulator - ns (version 2), *available at www.isi.edu/nsnam/ns* (retrieved on 11/25/2003 at 9AM CST).

[43] A. Chakrabarti, A. Striegel, and G. Manimaran, "A Case for Tree Evolution in QoS Multicasting," in *Proc. IEEE/IFIP IWQoS,* pp. 116-125, 2002.

[44] 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.

[45] 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.

[46] 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.

[47] J. Anderson, B. Doshi, S. Dravida, and P. Harshavadhana, "Fast restoration of ATM networks," *IEEE J. Select. Areas Communications,* vol.12, no.1, pp.128-138, Jan. 1994.

[48] S. Ramamurthy and B. Mukherjee, "Survivable WDM Mesh Networks, Part I - Protection," in *IEEE INFOCOM,* pp. 744-751, Mar. 1999.

[49] M. Sridharan, M. V. Salapaka, and A. K. Somani, "A Practical Approach to Operating Survivable WDM Networks," in *IEEE JSAC,* vol. 20, no. 1, Jan. 2002.

[50] F. Kuipers, P. Van Mieghem, T. Korkmaz, and M. Krunz, "An Overview of Constraint-Based Path Selection Algorithms for QoS Routing," in *IEEE Communications,* vol. 40, no. 12, pp. 50-55, Dec. 2002.

[51] D. H. Lorentz and A. Orda, "Optimal Partition of QoS Requirements in Unicast Paths and Multicast trees," in *Proc. IEEE INFOCOM,* Apr. 1999.

[52] D. H. Lorentz, A. Orda, D. Raz, and Y. Shavitt, "Efficient QoS Partition and Routing of Unicast and Multicast," in *IEEE/IFIP IWQoS,* June 2000.

[53] A. Sprintson and A. Orda, "A Scalable Approach to the Partition of QoS requirements in Unicast and Multicast," in *Proc. IEEE INFOCOM,* Apr. 2002.

[54] Turgay Korkmaz and Marwan Krunz, "Multi-Constrained Optimal Path Selection," in *Proc. IEEE INFOCOM,* Apr. 2001.

[55] X. Yuan, "Heuristic Algorithms for Multi-constrained quality-of-service routing," *IEEE/ACM Trans. of Networking,* vol. 10, no. 2, pp. 244-256, 2002.

[56] G. Liu and K. G. Ramakrishnan, "A*Prune: an algorithm for finding k shortest paths subject to multiple constraints," in *Proc. INFOCOM 2001,* pp. 834-843, Apr. 2001.

[57] Hussein F. Salama, Douglas reeves, and Yannis Viniotis, "A Distributed Algorithm for Delay-Constrained Unicast Routing," in *Proc. IEEE INFOCOM,* Apr. 1997.

[58] K. P. Gummadi, M. J. Pradeep, and C. S. R. Murthy, "An Efficient Primary Segmented Backup Scheme for Dependable Real-Time Communication in Multihop Networks," *to appear IEEE/ACM Trans. of Networking.*

[59] Shree Murthy and J.J. Garcia-Luna-Aceves, "Loop-Free Internet Routing using Hierarchical Routing Trees," in *Proc. INFOCOM,* 1997.

[60] S. Shenker and L. Breslau, "Two issues in reservation establishment," *Computer Communications Review,* vol.25, no.4. pp.14-26, 1995.

[61] A. Banerjea, D. Ferrari, D. Mah, M. Moran, D. Verma and H. Zhang, "The Tenet real-time protocol suite: design, implementation, and experiences," in *Technical Report TR-94-059, International Computer Science Institute,* November 1994.

[62] J. Csirik, J.B.G. Frenk, M. Labb and S. Zhang, "Heuristic for 0-1 Min-Knapsack Problem," in *Acta Cybernetica* vol.10, no.1-2, pp.15-20, Jan. 1991.

[63] P. Francis, "Yoid: Your own Internet Distribution," *www.aciri.org/yoid,* Apr. 2000 (retrieved on 11/25/2003 at 8:30 AM CST).

[64] J. Jannotti, D. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O' Toole Jr, "Overcast: Reliable Multicasting with the overlay network," in *Proc. Symp. on Operating Systems Design and Implementation (OSDI),* Oct. 2000.

[65] N. Deo and S. L. Hakimi, "The shortest Generalized Hamiltonian Tree," in *Proc. Annual Allerton Conference,* pp. 879-888, 1968.

[66] G. Zhou and M. Gen, "Application to Degree Constrained Minimum Spanning Tree Problem using Genetic Algorithm," in *Engineering Design and Automation,* vol. 3, no. 2, pp. 157-165, 1997.

[67] G. Korsatz and D. Pelleg, "Generating Low-Degree 2-Spanners," in *SIAM Journal on Computing,* vol. 27, no. 5, pp. 1438-1456, Oct. 1998.

[68] Athanasios Papoulis, "Probability, Random Variables and Stochastic Processes," *Third Edition, McGraw-Hill Inc.,* 1991.

ACKNOWLEDGMENTS

I would like to take this opportunity to thank my major professor Dr. Manimaran Govindarasu, without whose help the dissertation could not have been completed. He not only introduced me to the concepts of QoS multicasting and networking in general, but also he guided me throughout this five year doctoral process. He also made every effort to make this period as enjoyable and fruitful for me as possible. I would also like to thank Dr. Somani and Dr. Kamal for agreeing to be in my doctoral committee, and for sparing time to discuss with me the different aspects of the dissertation. Their comments and suggestions have been very useful in making the dissertation better. I would also like to extend my sincere gratitude to Dr. Baca and Dr. Davis for being in my committee and providing assistance whenever I needed them. In addition, I would like to thank Dr. Aaron Striegel, Dr. Murari Sridharan and Dr. Srinivasan Ramasubramanian for their inputs and discussions. I am also grateful to my colleagues Suzhen Lin and Basheer Al-Duwairi for their useful comments and all the help they provided. I would also like to thank Neha Kothari for proof-reading the dissertation and for making useful suggestions.

On personal front, I am grateful to my mother for being so patient and understanding during these five years. I would also like to thank my cousin Bodhisatva Das for making me feel at home in the foreign country and providing me with support whenever I needed it. I would like to thank my uncle, aunt and cousins for being loving all the time. Finally, I would like to thank all my friends for their constant encouragement and support.