RP RELOCATION IN

PROTOCOL INDEPENDENT MULTICAST – SPARSE MODE

Ritesh Mukherjee

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Computer Science at

Concordia University

Montreal, Quebec, Canada

April 2002

ABSTRACT

RP Relocation in Protocol Independent Multicast – Sparse Mode

Ritesh Mukherjee

Protocol Independent Multicast – Sparse Mode is the most widely used multicast routing architecture. It builds a shared distribution tree centered at a *Rendezvous Point* and then builds source-specific trees for those sources whose data traffic warrants it. Current implementations of the protocol decide on the locations of the Rendezvous Point administratively, which leads to congestion and delays. An attractive solution would be dynamic relocation of the Rendezvous Point depending on the members of the multicast group. In this thesis we present a Rendezvous Point calculation and relocation mechanism for Protocol Independent Multicast – Sparse Mode.

ACKNOWLEDGEMENTS

I would like to express the deepest appreciation to my supervisor, Dr. J. W. Atwood, who has convincingly conveyed a spirit of adventure in regard to research and scholarship, and an excitement in regard to teaching. Without his guidance and persistent help this thesis would not have been possible.

Special thanks should be given to the Computer Science Department and my student colleagues who helped me in many ways. Finally, words cannot express the thanks I owe to my parents and my brother for their encouragement and assistance.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Traditional network computing applications involve communication between two computers. However, some important emerging applications such as LAN TV, desktop conferencing, corporate broadcasts, and collaborative computing require simultaneous communication between groups of computers. This process is known generically as multipoint communications.

There are three ways to design multipoint networking applications: unicast, broadcast and multicast.

- Unicast: With a unicast design, applications can send one copy of each packet to each member of the multicast group. This technique is simple to implement, but it has significant scaling restrictions if the group is large. In addition, it requires extra bandwidth, because the same information has to be carried multiple times – even on shared links.

- Broadcast: In a broadcast design, applications can send one copy of each packet and address it to a broadcast address. This technique is even simpler than unicast for the application to implement. However, sending

the broadcast everywhere is a significant usage of network resources if only a small group actually needs to see the packets.

- Multicast: With a multicast design, applications can send one copy of each packet and address it to the group of computers that want to receive it. This technique addresses packets to a group of receivers rather than to a single receiver, and it is the responsibility of the network to forward the packets to only the network hosts that need to receive them.

Thus multicast is a bandwidth conserving technology that reduces traffic by simultaneously delivering a single stream of information to recipients. Multicast delivers source traffic to multiple receivers without adding any additional burden on the source or the receivers, while using the least bandwidth of any competing technology. Multicast packets are replicated in the network by routers enabled with multicast protocols, resulting in the most efficient delivery of data to multiple receivers possible. Thus the use of multicast routing technology is growing in today's Internet.

Multicast is based on the concept of a group. An arbitrary group of receivers expresses an interest in receiving a particular data stream. This group does not have any physical or geographical boundaries – the hosts can be located anywhere on the Internet. Hosts that are interested in receiving data flowing to a particular group must join the group using the Internet Group Management

Protocol (IGMP). A host must be a member of the group to receive the data stream.

Existing multicast routing protocols can be categorized into two classes as follows:

Source-Based: Traditionally, the problem was thought of as a source sending data to a group of receivers. Assuming that the membership distribution is fairly dense, the approach adopted is to broadcast or flood the network with the data. Routers that do not want data from a particular source can build a source specific prune state and convey this upstream towards the source. That way, data will only be forwarded to routers that have receivers connected to them or have downstream routers that do not have a prune state for that static stream. This requires each router to maintain a source specific prune state if they do not want traffic to be forwarded to them. With a large number of groups and sources this space requirement can become a limiting factor. In the worst case a router can have O(S x G) state stored where S is the number of sources and G is the number of groups.

Multicast routing protocols that are designed to work well in environments where it is reasonable to assume that receivers are densely distributed are known as dense mode routing protocols. Examples of such protocols are Distance Vector Multicast Routing Protocol (DVMRP), Multicast Extensions to Open Shortest

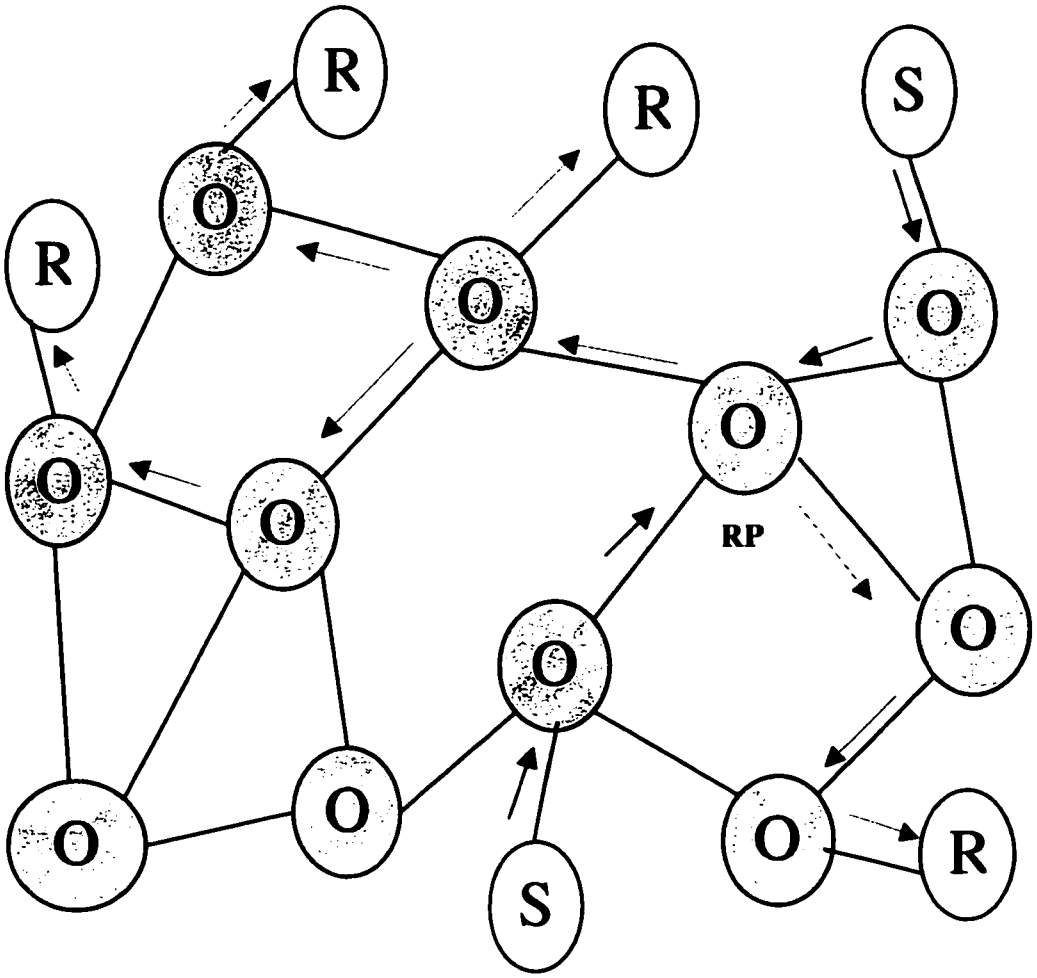
Path First (MOSPF) and Protocol Independent Multicast – Dense Mode (PIM–DM).

Group-Shared: The other approach is to assume that receivers for a particular group are distributed sparsely in the network. Therefore it is appropriate for the sender not to send to any other node until it explicitly shows its interest in the reception of that traffic. This approach leads to a *group-shared tree*, where each group has an associated tree. The state requirement in this case is O(G) where G is the number of multicast groups.

Multicast routing protocols that are designed to work well in environments where it is reasonable to assume that receives are sparsely distributed are known as sparse mode routing protocols. Examples of such protocols are Core Based Trees (CBT) and Protocol Independent Multicast – Spare Mode (PIM–SM).

Due to scalability of the approach, Group-Shared trees are a very common form of tree building in multicast routing protocols. Protocol Independent Multicast – Sparse Mode (PIM-SM) [1] is the most widely used Group-Shared multicast routing protocol. PIM-SM uses explicit joins that propagate hop-by-hop from members' directly connected routers towards the distribution tree. This builds a shared multicast distribution tree centered at the Rendezvous Point (RP), and then builds source-specific trees for those sources whose data warrants it.

The RP is a router that has been configured to be used as the root of the non–source–specific distribution tree for a multicast group.



S - SOURCE, R - RECEIVER, O - ROUTER

Figure 1: Group Shared Multicast tree

Switching to source-specific trees for low data rate sources would increase the number of states to be stored in the router to O(S x G) where S is the number of sources and G is the number of multicast groups. Thus a group-shared tree

centered at the RP is used when there are a large number of sources with comparable data rates.

The current method of choosing the RP for a multicast group is by administratively selecting a position in the network depending upon the expected source and receiver locations. However, with the variety of today's high bandwidth applications it is impossible for an administrator to "guess" where the sources and receiver are most likely to be. An RP located far away from the sources and receivers would lead to high tree costs, delays and reduced network efficiency. To improve network performance, it is necessary for the RP to be dependent on the sources and receivers in the multicast group at any point in time. With sources and receivers joining and leaving groups an RP has to be dynamically relocated to maintain network efficiency and reduce delays. This thesis proposes a dynamic RP calculation and relocation mechanism.

CHAPTER 2

PROTOCOL INDEPENDENT MULTICAST – SPARSE MODE

Protocol Independent Multicast – Sparse Mode (PIM-SM) routes multicast packets to multicast groups, and is designed to efficiently establish distribution across wide area networks. PIM-SM is called protocol independent because it can use the route information that any routing protocol enters into the Multicast Routing Information Base (MRIB).

PIM-SM must be able to route data packets from sources to receivers without either the sources or receivers knowing *a priori* of the existence of the others. This is essentially done in three phases, although as senders and receivers may come and go at any time, all three phases may occur simultaneously.

2.1 Phase One: RP Tree

In this phase, a multicast receiver expresses its interest in receiving traffic destined for a multicast group. Typically it does this using IGMP (Internet Group

Management Protocol) or MLD (Multicast Listener Discovery), but other mechanisms might also serve this purpose. One of the receiver's local routers is elected as the Designated Router (DR) for that subnet. On receiving the receiver's expression of interest, the DR then sends a PIM Join message towards the RP for that multicast group. This Join message is known as a (*, G) Join because it joins group G for all sources to that group. The (*, G) Join travels hop-by-hop towards the RP for the group, and in each router it passes through, multicast tree state for group G is instantiated. Eventually the (*, G) Join either reaches the RP, or reaches a router that already has (*, G) Join state for that group. When many receivers join the group, their Join messages converge on the RP, and form a distribution tree for group G that is rooted at the RP. This is known as the RP Tree (RPT), and is also known as the shared tree because it is shared by all sources sending to that group. Join messages are resent periodically as long as the receiver remains in the group. When all receivers on a leaf-network leave the group, the DR will send a PIM (*, G) Prune message towards the RP for that multicast group. However if the Prune message is not sent for any reason, the state will eventually time out.

A multicast data sender just starts sending data destined for a multicast group. The sender's local router (DR) takes those data packets, unicast-encapsulates them, and sends them directly to the RP. The RP receives these encapsulated data packets, decapsulates them, and forwards them onto the shared tree. The packets then follow the (*, G) multicast tree state in the routers on the RP Tree,

being replicated wherever the RP Tree branches, and eventually reaching all the receivers for that multicast group. The process of encapsulating data packets to the RP is called registering, and the encapsulation packets are known as PIM Register packets.

At the end of phase one, multicast traffic is flowing encapsulated to the RP, and then natively over the RP tree to the multicast receivers.

2.2 Phase Two: Register Stop

Although Register-encapsulation may continue indefinitely, the RP will normally choose to switch to native forwarding. To do this, when the RP receives a register-encapsulated data packet from source S on group G, it will normally initiate a (S, G) source-specific Join towards S. This Join message travels hop-by-hop towards S, instantiating (S, G) multicast tree state in the routers along the path. (S, G) multicast tree state is used only to forward packets for group G if those packets come from source S. Eventually the Join message reaches S's subnet or a router that already has (S, G) multicast tree state, and then packets from S start to flow following the (S, G) tree state towards the RP. These data packets may also reach routers with (*, G) state along the path towards the RP – if so, they can join the RP tree at this point.

While the RP is in the process of joining the source-specific tree for S, the data packets will continue being encapsulated to the RP. When packets from S also start to arrive natively at the RP, the RP will be receiving two copies of each of these packets. At this point, the RP starts to discard the encapsulated copy of these packets, and it sends a RegisterStop message back to S's DR to prevent the DR unnecessarily encapsulating the packets.

At the end of phase 2, traffic will be flowing natively from S along a source-specific path to the RP, and from there along the shared tree to the receivers. Where the two trees intersect, traffic may transfer from the source-specific tree to the RP tree, and so avoid taking a long detour via the RP.

It should be noted that a sender may start sending before or after a receiver joins the group, and thus phase two may happen before the shared tree to the receiver is built.

2.3 Phase Three: Shortest-Path Tree

Although having the RP join back towards the source removes the encapsulation overhead, it does not completely optimize the forwarding paths. For many receivers the route via the RP may involve a significant detour when compared with the shortest path from the source to the receiver.

To obtain lower latencies, a router on the receiver's LAN, typically the DR, may optionally initiate a transfer from the shared tree to a source-specific shortest-path tree (SPT). To do this, it issues a (S, G) Join towards S. This instantiates state in the routers along the path to S. Eventually this join either reaches S's subnet, or reaches a router that already has (S, G) state. When this happens, data packets from S start to flow following the (S, G) state until they reach the receiver.

At this point the receiver (or a router upstream of the receiver) will be receiving two copies of the data - one from the SPT and one from the RPT. When the first traffic starts to arrive from the SPT, the DR or upstream router starts to drop the packets for G from S that arrive via the RP tree. In addition, it sends a (S, G) Prune message towards the RP. This is known as (S, G, rpt) Prune. The Prune message travels hop-by-hop, instantiating state along the path towards the RP indicating that traffic from S for G should *not* be forwarded in this direction. The prune is propagated until it reaches the RP or a router that still needs the traffic from S for other receivers.

By now, the receiver will be receiving traffic from S along the shortest-path tree between the receiver and S. In addition, the RP is receiving the traffic from S, but this traffic is no longer reaching the receiver along the RP tree. As far as the receiver is concerned, this is the final distribution tree.

2.4 RP Discovery

For correct operation, every PIM router within a PIM domain must be able to map a particular multicast group address to the same RP. If this is not the case then black holes may appear, where some receivers in the domain cannot receive some groups.

A small set of routers from a domain is configured as candidate bootstrap routers (C-BSRs) and, through a simple election mechanism, a single BSR is selected for that domain. Sets of routers within a domain are also configured as candidate RPs (C-RPs); typically these will be the same routers that are configured as C-BSRs. Candidate RPs periodically unicast Candidate-RP-Advertisement messages (C-RP-Advs) to the BSR of that domain, advertising their willingness to be an RP. A C-RP-Adv message includes the address of the advertising C-RP, as well as an optional list of group addresses and a mask length field, indicating the group prefix for which the candidacy is advertised. The BSR then includes a set of these Candidate-RPs (the RP-Set), along with their corresponding group prefixes, in Bootstrap messages it periodically originates. Bootstrap messages are distributed hop-by-hop throughout the domain.

All the PIM routers in the domain receive and store Bootstrap messages originated by the BSR. When a DR receives an indication of local membership (typically from IGMP or MLD) or a data packet from a directly connected host, for

a group for which it has no forwarding state, the DR uses the following algorithm for performing the group-to-RP mapping:

1. Perform longest match on group-range to obtain a list of RPs
2. From this list of matching RPs, find the one with highest priority. Eliminate any RPs from the list that have lower priorities.
3. If only one RP remains in the list, use that RP.
4. If multiple RPs are in the list, use the PIM hash function to choose one.

The hash function is used by all routers within the domain, to map a group to one of the RPs from the matching set of group–range–to–RP mappings (RPs in this set have the same highest priority). The candidate RP with the highest resulting hash value is chosen as the RP. If more than one RP has the same highest hash value, the RP with the highest IP address is chosen.

The DR then sends a Join message towards that RP if the local host joined the group, or it Register-encapsulates and unicasts the data packet to the RP if the local host sent a packet to the group. A Bootstrap message indicates liveness of the RPs included therein. If an RP is included in the message, then it is tagged as `up' or functioning at the routers; RPs not included in the message are removed from the list of RPs over which the hash algorithm acts. Each router continues to use the contents of the most recently received Bootstrap message from the BSR until it accepts a new Bootstrap message.

CHAPTER 3

NEW RP CALCULATION

The location of the RP decides the efficiency of the multicast tree. From the user's point of view, efficiency would mean minimizing end-to-end delay. The network would prefer to minimize the tree cost and be able to reduce congestion. The shared tree routed at the RP does not optimize the delivery path through the network. If the RP is located far from the participants, a longer packet delay is experienced and excessive resources are consumed. If an RP is selected for a set of participants that later changes drastically, the RP location will severely affect the quality of the tree for new participants. Therefore the RP should be dynamically located depending upon the participants in the multicast group.

Ideally, a group–shared tree would use a minimal spanning tree to minimize total bandwidth usage. Finding this tree for some subset of nodes in a graph is known as the Minimal Steiner Tree Problem, and is known to be NP–complete [6]. While locating the best center is simple given complete topological information, such information is not always available in distributed routing protocols.

Ideally, we would want the RP to be located in the vicinity of the senders to the multicast group. However, there is not enough information to proceed with such a

calculation. Another method would be to use the RP-set broadcast by the BSR in the bootstrap message [5]. But this RP-set is too small and unlikely to contain the address of the best RP for the multicast group. An algorithm that considers all the potential routers would be best suited for calculating the best RP [3].

Assume RPold is the current RP of G; we propose an RP relocation mechanism that relocates the RP to RPnew. It should clearly be noted that this relocation is done only when there is a significant improvement in tree cost. The relocation comes into effect after a certain time period and when there are changes in the group membership.

To determine the position of RPnew [4], RPold uses the topology information available to it from the Interior Gateway Protocol (such as OSPF (Open Short Path First), which has the required information in the Link State Database). The cost of a tree is the sum of the number of links in the tree. We begin by describing two functions that will help in deciding if the RP should be moved to a new location or not.

1. The function cost_calculate(root) uses the estimation function to calculate the cost of a tree at a given root:

   Let S be the set of sources for the multicast group, u a source in S, root the supplied root to the algorithm and d(a, b) the distance from a to b.

First, get a lower bound on the cost of the tree at some node. In this case, the best-case tree is linear, that is, all group members lie on the path from the root to the farthest member. When distances are given as hop counts, we can get a slightly tighter bound. Specifically if two group members are at an equal distance, the distribution tree cannot be completely linear, but must have one additional link. Thus,

Minimum Est. Cost =

max d(root, u) + number of duplicate distance nodes in S

To get an upper bound on the cost of the tree routed at some node, in the worst-case tree no links are shared among the paths to each member. Thus the maximum tree cost is the sum of the member distances. If the number of group members (other than the root, if it a member) is greater than the root degree, we may tighten the bound by subtracting the difference to account for the knowledge of sharing those links. Thus,

Maximum Est. Cost = sum d(root, u) if |S|<=deg(root)

sum d(root, u)-(|S|-deg(root)) otherwise

|S| is the number of sources in the multicast group.

The final estimated cost is given by,

Final Est. Cost = (Minimum Est. Cost + Maximum Est. Cost)/2

2. The function why_move() follows the following algorithm:

   1. Initialize a variable i equal to 0 and an array RPmove[] of size 5 equal to NULL. Assign to RPmove[0] the value RPold.

   2. Call cost_calculate with RPmove[0] as the root. Store the cost in a variable currcost and the visited node in a list dontrepeat[].

   3. Call cost_calculate with all of the neighboring unvisited nodes of RPmove[0] as the root. Store the lowest cost in a variable lowestcost, the lowest cost node in a variable RPlow and add the visited nodes to the list dontrepeat[].

   4. If lowestcost is lower than currcost then assign lowestcost to currcost else goto step 6.

5. Shift all the values of the array RPmove[] to the next location (i.e., assign RPmove[1] := RPmove[0] and so on till RPmove[4] := RPmove[3]). Assign RPmove[0] with the value RPlow and goto step 3.

   6. If RPmove[0] is different from RPold and the estimated cost of the tree

with RP at RPold is greater than lowercost by more than the threshold, then return RPmove[] else make all values of RPmove equal to NULL and return RPmove[].

The value of the variable threshold used in the algorithm is a predefined value set by the network administrator. Thus calling the function why_move() returns RPmove[]. If RPmove[0] is NULL then the RP should not be relocated. If RPmove[0] is not equal to NULL then the RP should be relocated. This leaves us with the problem of dynamically switching from RPold to RPnew[0].

CHAPTER 4

RP RELOCATION

Figure 1 on the next page shows the outline of the relocation algorithm. When the relocation timer has expired and there is a change in the group membership the present tree cost is calculated. The tree cost with the RP at the best position is calculated.

If RPnew[0] is not NULL then the RP has to be relocated. For this a MOVE_JOIN message is sent to RPnew[0] and a Move_RP message is sent to the BSR. The variable reloc_var keeps track of the number of bootstrap messages received with the RP-set for the multicast group unchanged after the relocation has started.

```
RELOCATE(G)
set_of_sources S
begin
  while(1) do
    reloc_var = -1
    if (tr expired and S changed)
      RPnew[] = why_move()
      while(reloc_var != 0) do
       if (RPnew[0] != NULL)
        reloc_var = 1
        gen_move_join(RPold, RPnew[0], G)
        gen_move_RP(RPold, RPnew[0], G)
       else
        reloc_var = 0
       endif
      endwhile
    endif
  endwhile
end

tr - relocation timer
```

Figure 2: Relocation Algorithm

RPnew[0] can ignore all the messages if it is not an RP-capable router or is configured not to act as RP for any group. If RPnew[0] is RP-capable then it does the following:

On receiving the MOVE_JOIN message RPnew[0] assumes the role of RP for group G. When it receives an encapsulated message from RPold it sends the message out on the multicast tree. When it receives a message from a source it sends the message out on the multicast tree and encapsulates and sends this to RPold. On receiving the Move_RP message, if the BSR does not find RPnew[0] in the local pool of candidate RPs, it assumes that RPnew[0] is not RP-capable and ignores the message. Otherwise, it does the following:

1. To exclude the current members of the RP-set from being the RP for the group, the BSR can do one of the following:
   i) It can change the priority of RPnew[0] for that group to the minimum value of priority in that group (the lower the value of priority, the higher is the priority) and increase the priorities of all other candidates for that group by 1; or

   ii) It can only include the address of RPnew[0] in the RP-set being broadcast in the bootstrap message and leave out all the remaining candidates.

2. The BSR then sends out its bootstrap message.

The routers on receiving this bootstrap message join the group at RPnew[0]. While this transition is taking place RPold continues to act as the RP for the group. The MOVE_JOIN message sent to RPnew[0] ensures that RPold receives packets from RPnew[0] during the transition. When RPold receives a message from a source, it continues to send to the group and also encapsulates and sends the packet to RPnew[0]. On receiving a bootstrap message RPold checks the reloc_var. If the reloc_var for the group is greater than 0 and the RP-set for the group is unchanged, then it increases the value of reloc_var by 1.

Once all the members have pruned off from RPold, the tree to RPold is torn down. RPold then sends a MOVE_OVER message to RPnew[0] and resets the reloc_var to zero. The state is then killed and the tree is now set up at RPnew[0]. On receiving the MOVE_OVER message RPnew[0] stops sending encapsulated messages to RPold. This method of RP relocation ensures no loss of messages during the transition.

If RPold gets three bootstrap messages from the BSR with the same RP-set then the value of reloc_var will be four. If the value of reloc_var reaches four then RPold assumes that RPnew[0] is not RP-capable. It then checks the value of RPnew[1]. If RPnew[1] is NULL then it assumes that none of the other nodes can serve as RP for the group so it sets the reloc_var to zero and continues to serve

as RP for the multicast group. RPold will also stop sending encapsulated packets to RPnew[0]. If RPnew[1] is not NULL it shifts the values of the array to the previous locations (i.e., RPnew[0] := RPnew[1] and so on until RPnew[3] := RPnew[4] and finally RPnew[4] is assigned NULL). It then tries to relocate to the new value of RPnew[0], that is, the next best position where the RP of the multicast group should be located.

4.1 CONTROL MESSAGES

4.1.1 MOVE_JOIN: This message is used to tell RPnew[0] that it should act as the new RP for the multicast group G and should send any messages during the transition period to RPold. The IP address is set to the address of RPold and the destination address is set to RPnew[0]. The IP Time To Live (TTL) of the packet is the system's normal unicast TTL.

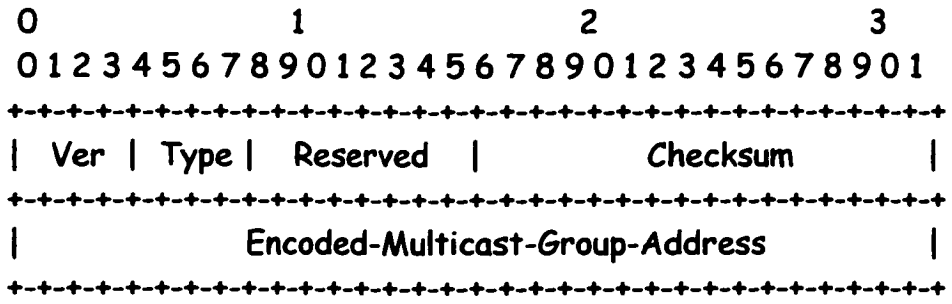


Figure 3: MOVE_JOIN message format

Ver : PIM Version number is 2.

Type : 9 = Move_Join

Reserved : Set to zero on transmission. Ignored upon receipt.

Checksum : The checksum is a standard IP checksum.

Encoded-Multicast-Group-Address : The encoded multicast group address for the group G whose RP has to be relocated.

4.1.2 MOVE_RP: This message is used to tell the Bootstrap router that the RP for the group should be shifted from RPold to RPnew[0]. The IP address is set to the address of the RPold and the destination address is set to the address of the BSR. The IP TTL of the packet is the system's normal unicast TTL.

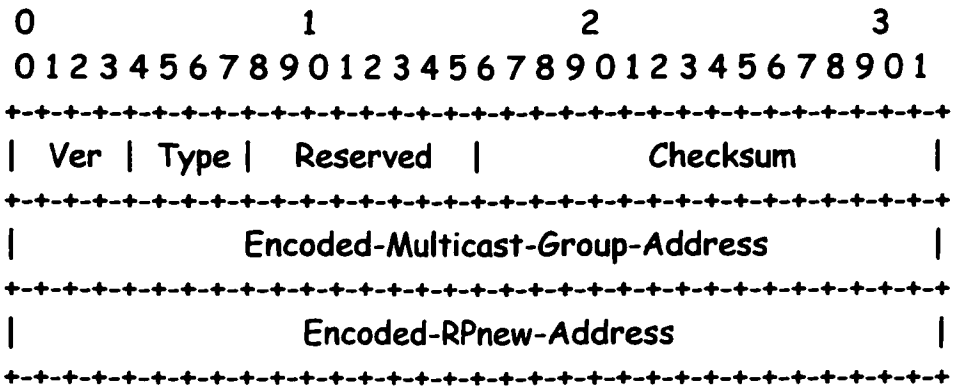


Figure 4: MOVE_RP message format

Ver : PIM Version number is 2.

Type : 10 = Move_RP

Reserved : Set to zero on transmission. Ignored upon receipt.

Checksum : The checksum is a standard IP checksum.

Encoded-Multicast-Group-Address : The encoded multicast group address for the group G whose RP has to be relocated.

Encoded-RPnew-Address : The encoded address of the router to which the RP should be relocated.

4.1.3 MOVE_OVER: This message is used to tell RPnew[0] that all the members of group G have left the multicast tree at RPold so there is no need to send any more messages to RPold. The IP address is set to the address of RPold and the destination address is set to RPnew. The IP TTL of the packet is the system's normal unicast TTL.

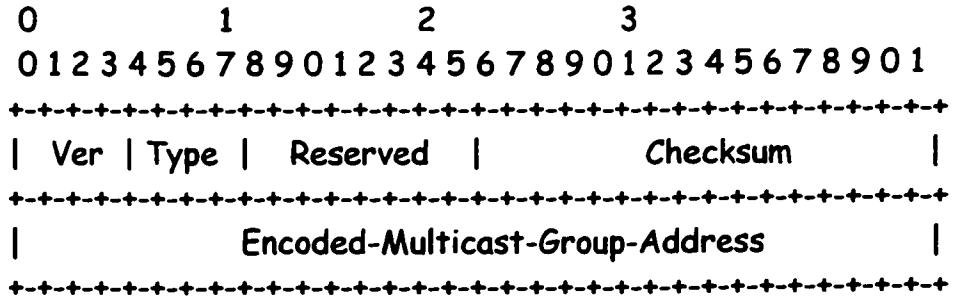


Figure 5: MOVE_OVER message format

Ver : PIM Version number is 2.

Type : 11 = Move_Over

Reserved : Set to zero on transmission. Ignored upon receipt.

Checksum : The checksum is a standard IP checksum.

Encoded-Multicast-Group-Address : The encoded multicast group address for the group G whose RP has to be relocated.

CHAPTER 5

SIMULATION

The aims of this simulation are to do the following:

1. To demonstrate that the RP calculation algorithm returns a position in the network that results in a local minimum cost of the shared multicast tree.
2. To study the improvements in tree cost with varying number of sources and network connectivity.
3. To study the improvement in time taken to multicast a fixed amount of data with various multicast scenarios.

The simulation consists of two parts: One measures the improvements in tree costs using random distribution of sources and the other measures improvements in time taken to deliver multicast data using specific multicast scenarios.

5.1 SIMULATION TO MEASURE TREE COSTS USING RANDOM DISTRIBUTION OF SOURCES

5.1.1 SIMULATION USING RANDOM GRAPHS

Graphs are commonly used to model the structure of networks, for the study of network problems. The networking literature contains a variety of random graphs used to model networks. All the random graph models distribute vertices at random locations in a plane and then consider each pair of vertices; an edge is added between a pair of vertices with probability p.

Waxman proposes one of the most common random graph models, with the probability of an edge from u to v given by:

$$P(u, v) = \alpha e^{-L / (d * \beta)}$$

where $0 < \alpha, \beta <= 1$ are parameters of the model, L is the Euclidean distance from u to v, and d is the maximum distance between any two nodes. An increase in α will increase the number of edges in the graph, while an increase in β will increase the ratio of long edges relative to shorter edges. Another model is the Exponential model, which uses:

$$P(u, v) = \alpha e^{-L / (d - L)}$$

The probability of an edge in this model decreases exponentially with the distance between the two vertices. A study of the parameter selection [10] for these models gives us typical values for the parameters to model networks.

This part of the simulation has been written in Visual C++ using Microsoft Visual Studio 6.0. The application asks the user to enter the number of nodes in the network, the number of sources and the type of network model to use to run the simulation. The user can specify two types of network models: Waxman or Exponential. The application uses a random number generator seeded with time to generate random numbers. It does the operations as follows:

1. First it initializes the number of nodes entered by the user. During this process it randomly selects an x and y coordinate for each node in the network, sets the link costs to zero and removes all links.

2. It then creates links between nodes in the network based on the model selected by the user. If the user selects Waxman Model it does the following for all nodes in the network:

    a. It takes two nodes in the network and calculates a probability using the following equation.

$$P = \alpha * e^{-L / (d * \beta)}$$

where α and β are constants set in the program.

L is the Euclidean distance between the two nodes

d is the maximum possible distance between any two nodes

b. If the probability is greater than a certain value it builds a link otherwise it does not build a link.

If the user selects the exponential model then it does the following for all nodes in the network:

a. It takes two nodes in the nodes in the network and calculates a probability using the following equation.

$$P = \alpha * e^{-L / (d - L)}$$

where α is a constant.

L is the Euclidean distance between the two nodes

d is the maximum possible distance between any two nodes

b. If the probability is greater than a certain value it builds a link otherwise it does not build a link.

3. After the nodes and links have been created the application builds the link state tables for all the nodes by running Dijkstra's algorithm. This leads to a table for each node that gives the next hop and the minimum cost to reach any other node in the network. The network is now ready.

4. It then randomly selects an RP for the group and randomly selects the number of sources entered by the user. This forms the multicast group.

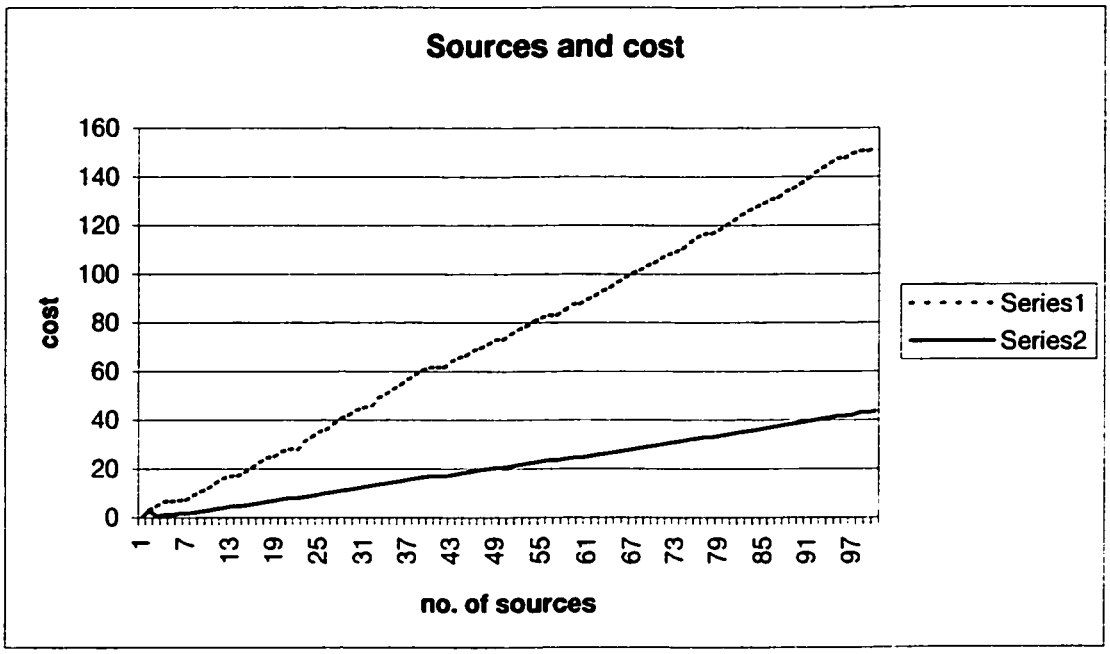
5. It then calls the function why_move( ) with the value of current RP. This function does the following.

    a. It calculates the cost of the tree with the current RP using the function cost_calculate( ).

    b. It then calculates the cost of the tree using the neighbors of the current RP as RP for the multicast group. If none of the costs are less than the cost of the tree with the current RP then it stops and returns the value of current RP else it loops back to 5) a) with the RP of the lowest cost tree as the current RP.

6. It displays the cost of the actual RP, the cost after relocation and the difference in the costs. (This cost is the estimated cost computed by cost_calculate( ).)

It draws the network on the screen whenever an OnPaint( ) call is made. It also calls a generate_file( ) function that can save the generated network to a file. There is another separate application that does the same operations but inputs the network from a file instead of generating a new network. This is used to keep the network the same and change the number of sources. Running such a simulation with varying sources results in the following graph for the Waxman Model.



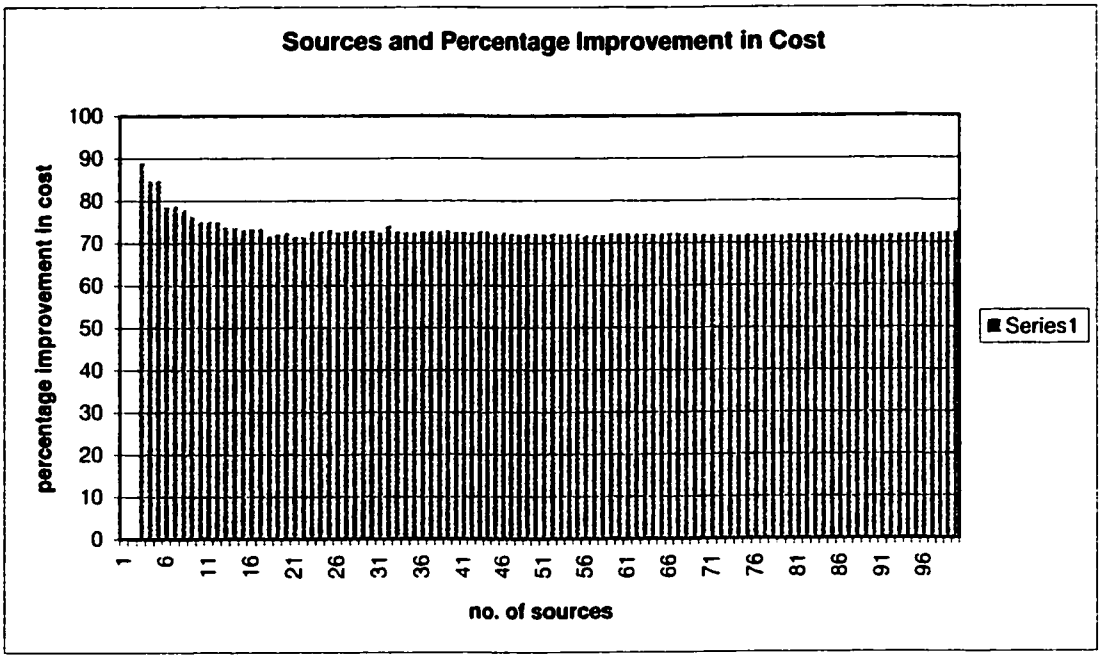
Series 1 - cost of the multicast tree without RP relocation

Series 2 - cost of the multicast tree after RP relocation

Figure 6 Sources and cost graph for Waxman Model

The graph in figure 6 represents an average of five different random networks. It shows a significant improvement in cost as the number of sources increases in a

multicast group. However the percentage improvement in cost remains fairly constant. The graph between the number of sources and the percentage improvement in cost is shown in figure 7.

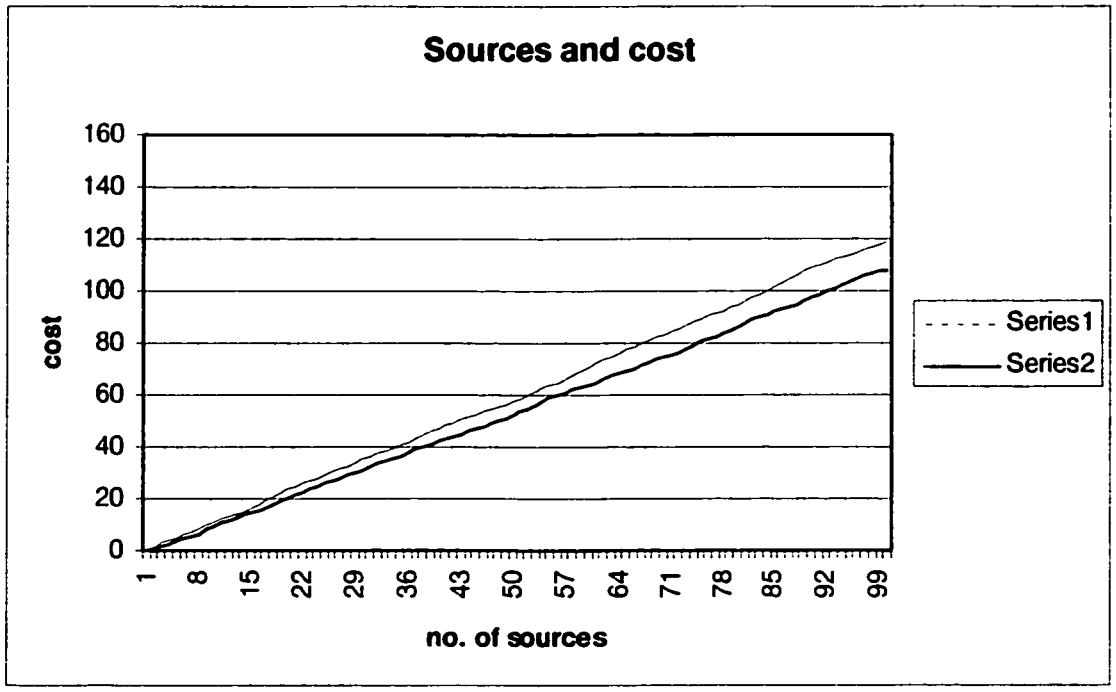


Series 1 - percentage improvement in cost after RP relocation

Figure 7: Sources and percentage improvement in cost graph for Waxman Model

This shows a fairly constant percentage improvement in cost of the multicast tree. The average percentage cost improvement is **71.40%.**

Running the same simulation with the Exponential Model gives the graph in figure 8:

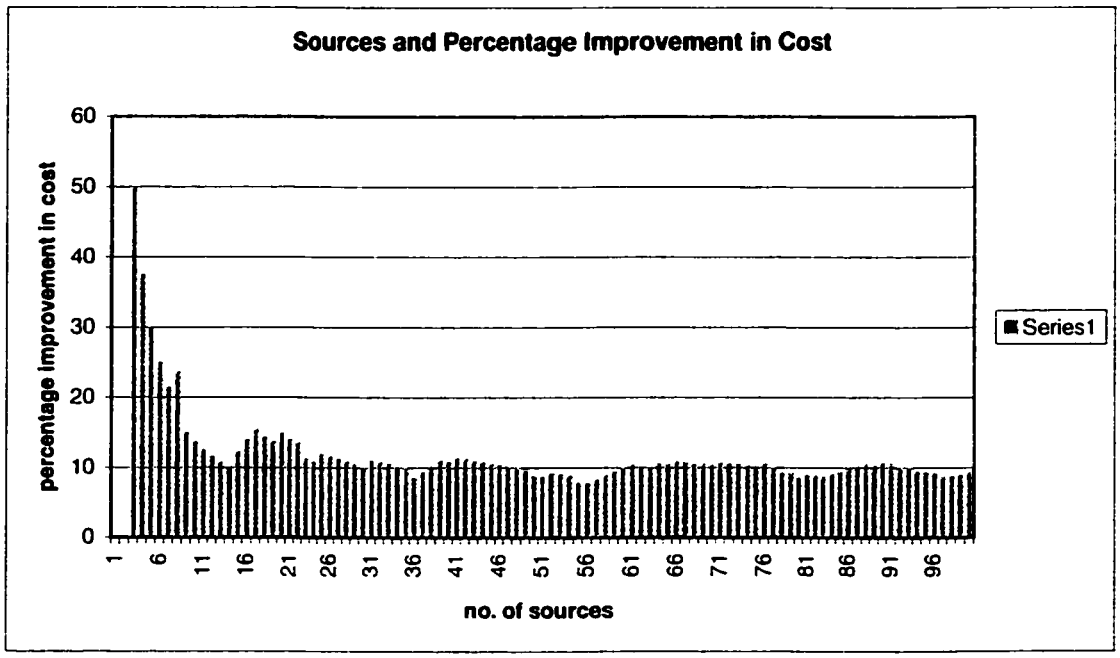


Series 1 - cost of the multicast tree without RP relocation

Series 2 - cost of the multicast tree after RP relocation

Figure 8: Sources and cost graph for Exponential Model

The graph in figure 8 represents an average of five different random networks. It shows a small improvement in cost as the sources increase. However the percentage improvement in cost is fairly constant. The graph between the number of sources and the percentage improvement in cost is as follows:



Series 1 - percentage improvement in cost after RP relocation

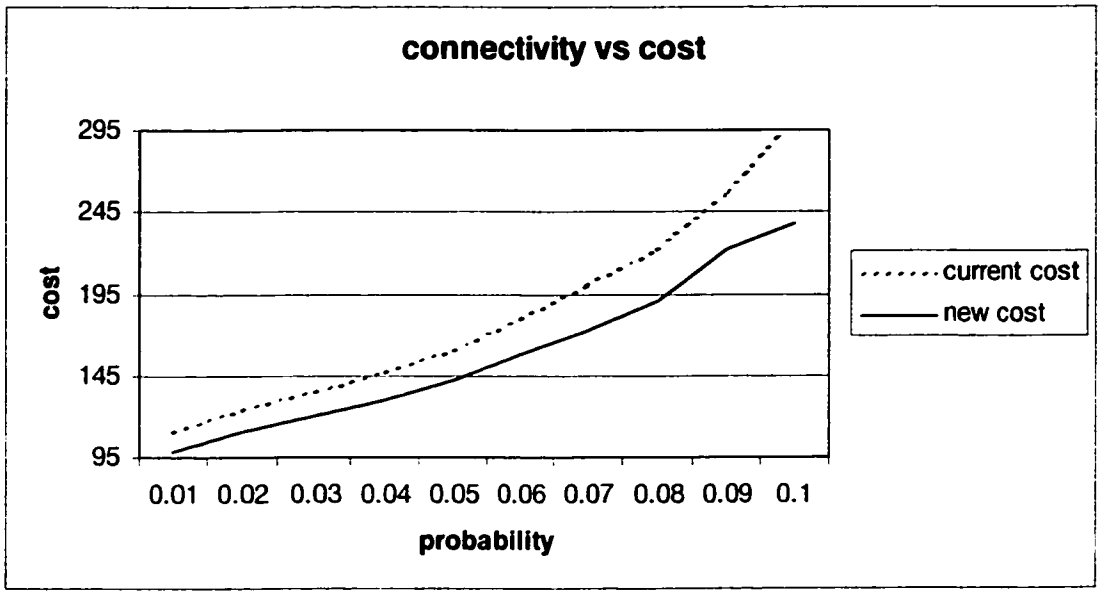
Figure 9: Sources and percentage improvement in cost graph for Exponential Model

This shows a fairly constant percentage improvement in cost of the multicast tree. The average percentage cost improvement is **11.42%.**

The possibility of a network being so well interconnected as the exponential model is very low. This leads us to do a simulation where we vary the connectivity of the network and keep the multicast group and the sources as they are. A separate application was created that asks the user to enter the probability above which a link will be created. This application keeps the network nodes as they are and changes the links depending upon the probability entered by the

user. The application creates a link if the calculated probability is greater than the probability entered by the user. Thus the probability of creating a link decreases as the probability entered by the user increases. It varies the connectivity while keeping the nodes constant.

Changing the connectivity of the network while keeping the multicast group and the nodes in the network as they are gives us the following graph for the Waxman Model.



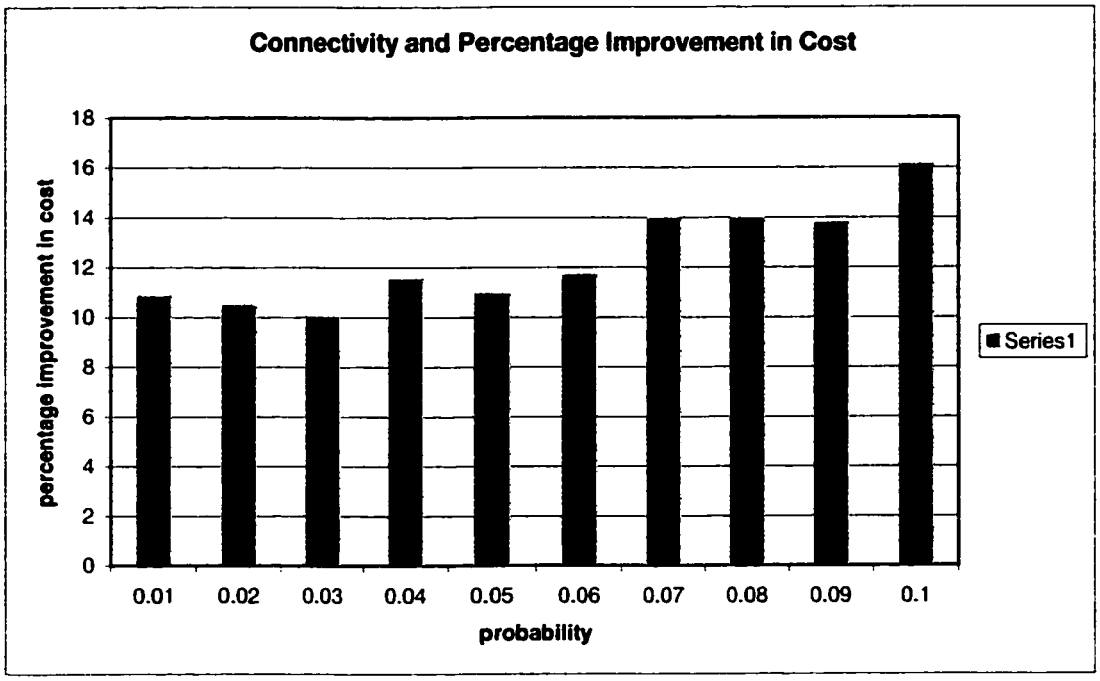
current cost - cost of the multicast tree without RP relocation

new cost - cost of the multicast tree after RP relocation

Figure 10: Connectivity and cost graph for Waxman Model

The graph in figure 10 represents an average over five different random networks. Note that probability on the x-axis is the probability above which the

calculated probability must be in order to create a link between two nodes. As the connectivity of the network decreases the need for RP relocation rapidly increases. However the percentage improvement in cost after relocation remains fairly constant. This can be seen from the figure 11.

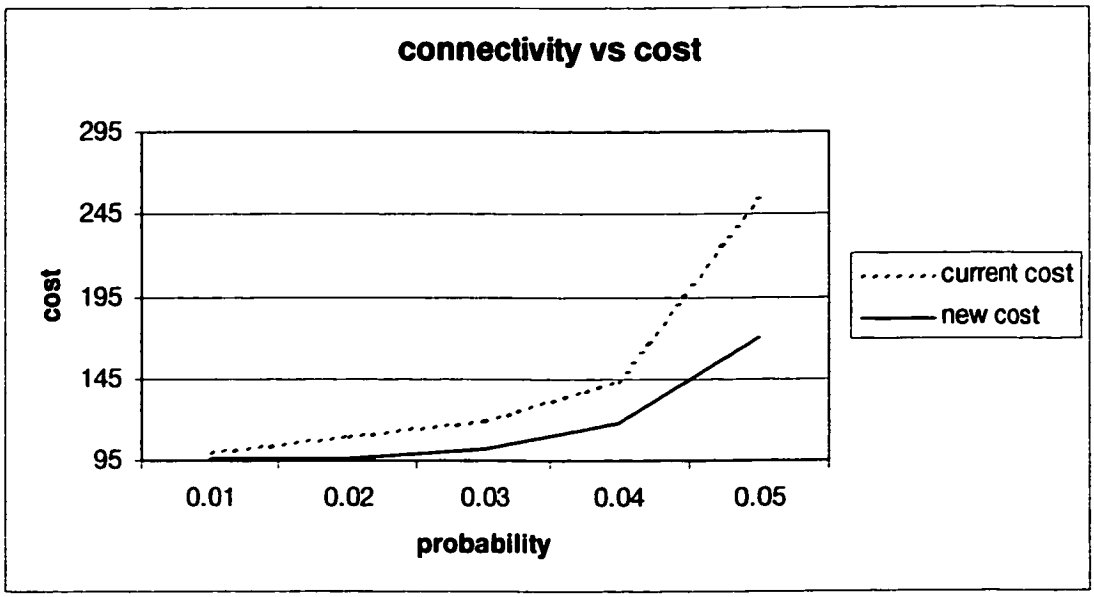


Series 1 - percentage improvement in cost after RP relocation

Figure 11: Connectivity and Percentage Improvement in cost graph for Waxman Model

This shows a fairly constant percentage improvement in cost of the multicast tree. The average percentage cost improvement is **12.34%.**

Changing the connectivity of the network while keeping the multicast group and the nodes in the network as they are gives us the graph for the Exponential Model shown in figure 12.

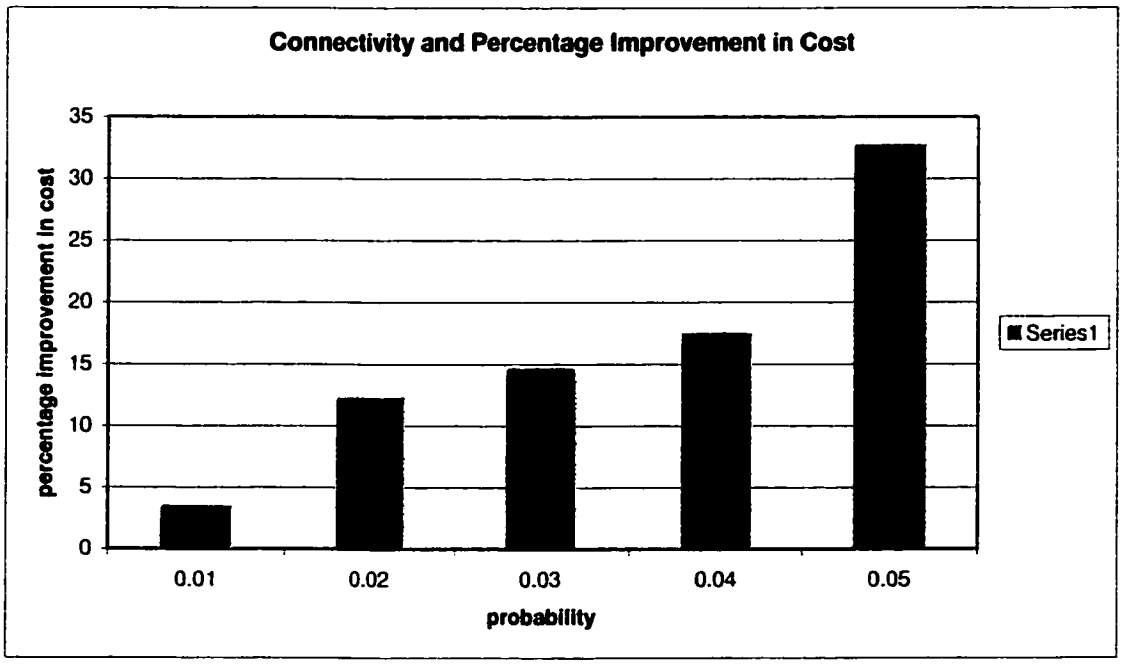


current cost - cost of the multicast tree without RP relocation

new cost - cost of the multicast tree after RP relocation

Figure 12: Connectivity and cost graph for Exponential Model

The graph in figure 12 represents an average over five different random networks. It shows that as the connectivity of the network decreases the need for relocation increases rapidly. However the percentage improvement in cost remains fairly constant as seen from the graph in figure 13:



Series 1 - percentage improvement in cost after RP relocation

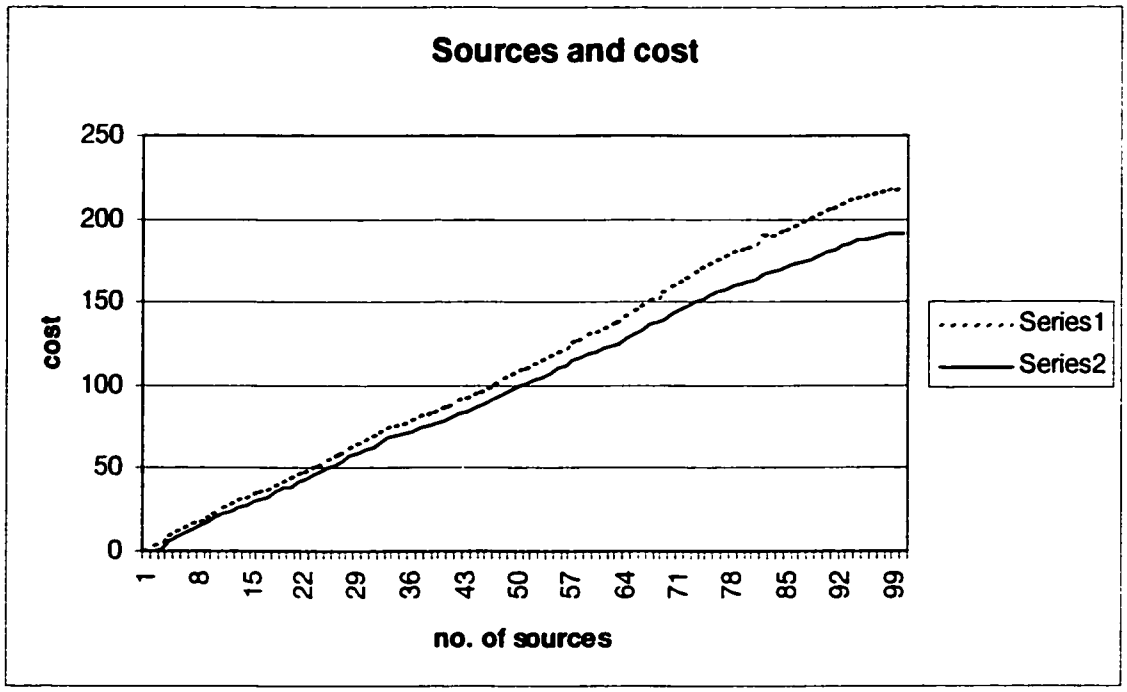
Figure 13: Connectivity and Percentage Improvement in cost graph for Exponential Model

The average percentage cost improvement is **16.13%.**

5.1.2 SIMULATION USING HIERARCHICAL GRAPHS

The Internet can be viewed as a collection of interconnected routing domains, which are groups of nodes that are under a common administration and share routing information. Each routing domain in the Internet can be classified as either a stub domain or a transit domain. A domain is a stub domain if the paths connecting any nodes u and v go through that domain only if u or v is in that domain; transit domains do not have this restriction. The purpose of transit domains is to interconnect stub domains efficiently; without them, every pair of stub domains would need to be directly connected.

The hierarchical graph required for the simulation is generated from the hierarchical network graphs available in Network Simulator (ns2). The hierarchical graphs in ns2 have been generated using the transit-stub model described above using Georgia-tech's SGB (Stanford GraphBase) graphs. These were used to generate network files for use with our VC++ simulator. The simulator inputs the network from the generated file. Running the simulation with varying sources gives the following graph:

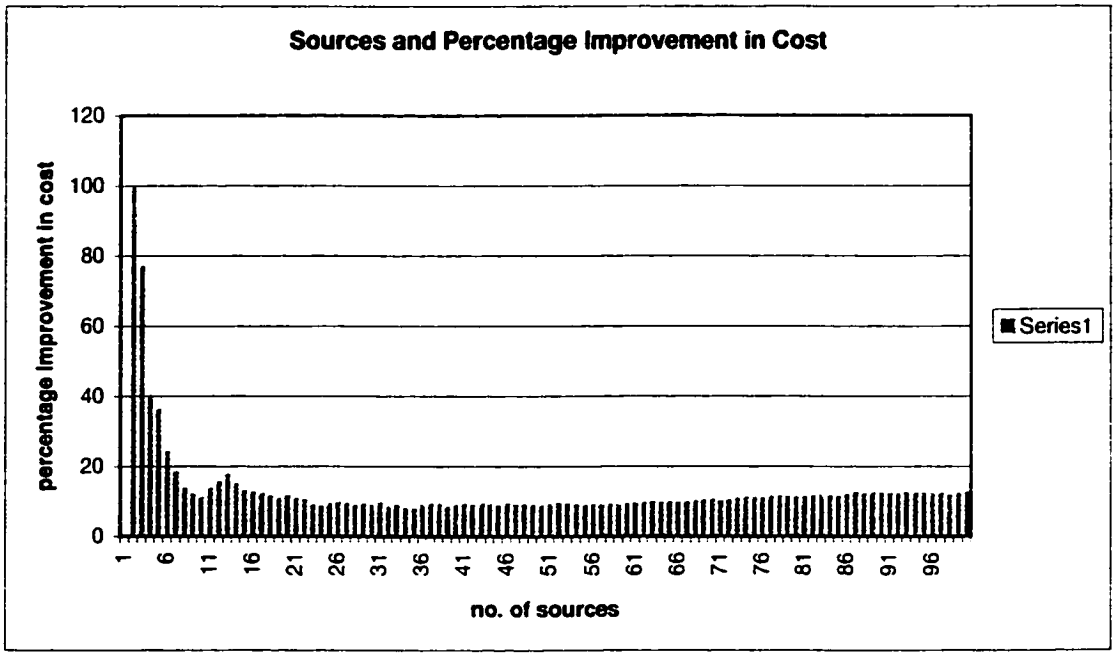


Series 1 - cost of the multicast tree without RP relocation

Series 2 - cost of the multicast tree after RP relocation

Figure 14: Sources and cost graph for Hierarchical Model

The percentage improvement in cost remains fairly constant. The graph between the number of sources and the percentage improvement in cost is as follows:



Series 1 - percentage improvement in cost after RP relocation

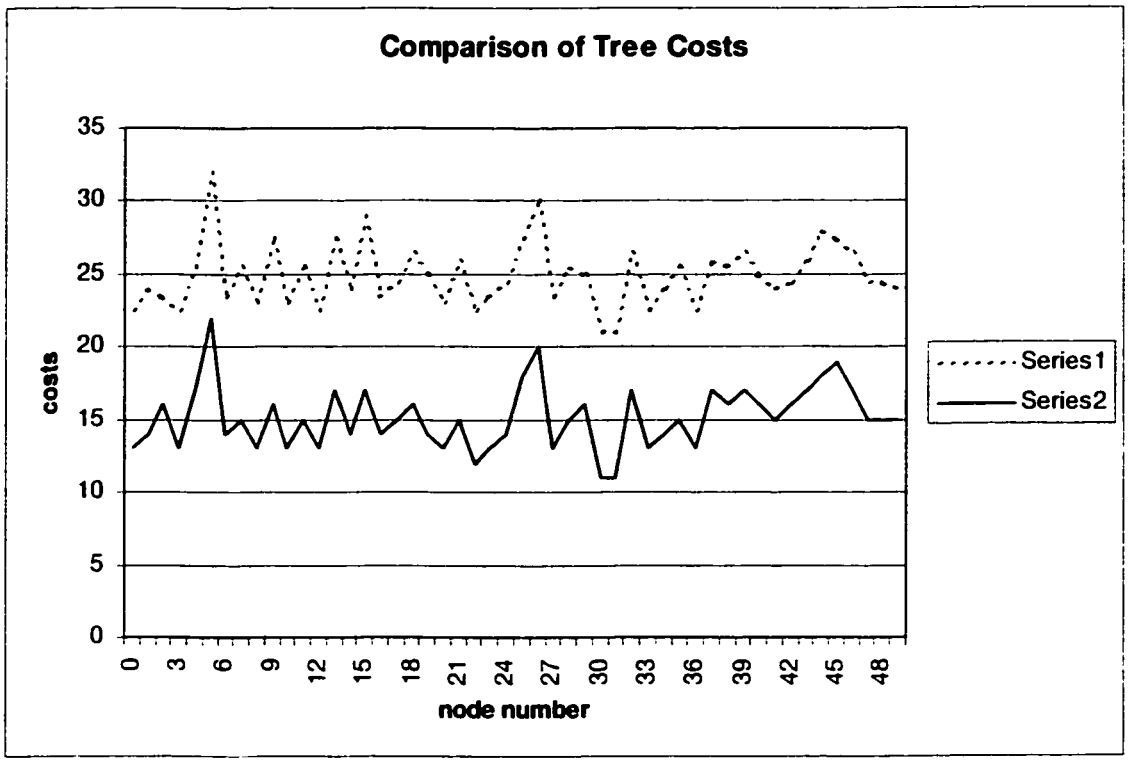
Figure 15: Sources and Percentage Improvement in cost graph for Hierarchical Model

This shows a fairly constant percentage improvement in cost of the multicast tree. The average percentage cost improvement is **12.69%**.

Another interesting observation in case of hierarchical model is that even though there is an improvement in tree costs after relocation, the RP always relocates to the top of the hierarchical structure for a random distribution of sources. This leads us to do another simulation where we measure the actual time taken for the multicast data to arrive at the receivers and compare them before and after RP relocation.

In cases of simulations using random distribution of sources we note that as sources join the multicast group the RP relocates to the best position in the network. When a number of sources join the multicast group the RP relocates to the same best position and the best position in not affected by addition of more sources. Experimental results show that random additions of sources after the RP has relocated to the best position and number of sources is more than 5% of the number of nodes in the network does not change the best RP position anymore.

For the estimation function to succeed in getting the best position of the RP we would want the trend in change of estimated costs of the tree taking different routers as the route to follow the same trend as the change in actual costs of the tree. The VC++ simulator was modified to show both estimated costs of the tree and actual costs of the tree taking each router as the route. The multicast group was chosen randomly and then kept fixed for the simulation. Then different routers were taken as the root of the multicast tree and the estimated and actual tree costs were calculated. For this the number of nodes in the network was taken to be 50 and the number of sources in the multicast group to be 15. The results when using the Waxman model are shown in the graph on figure 16.

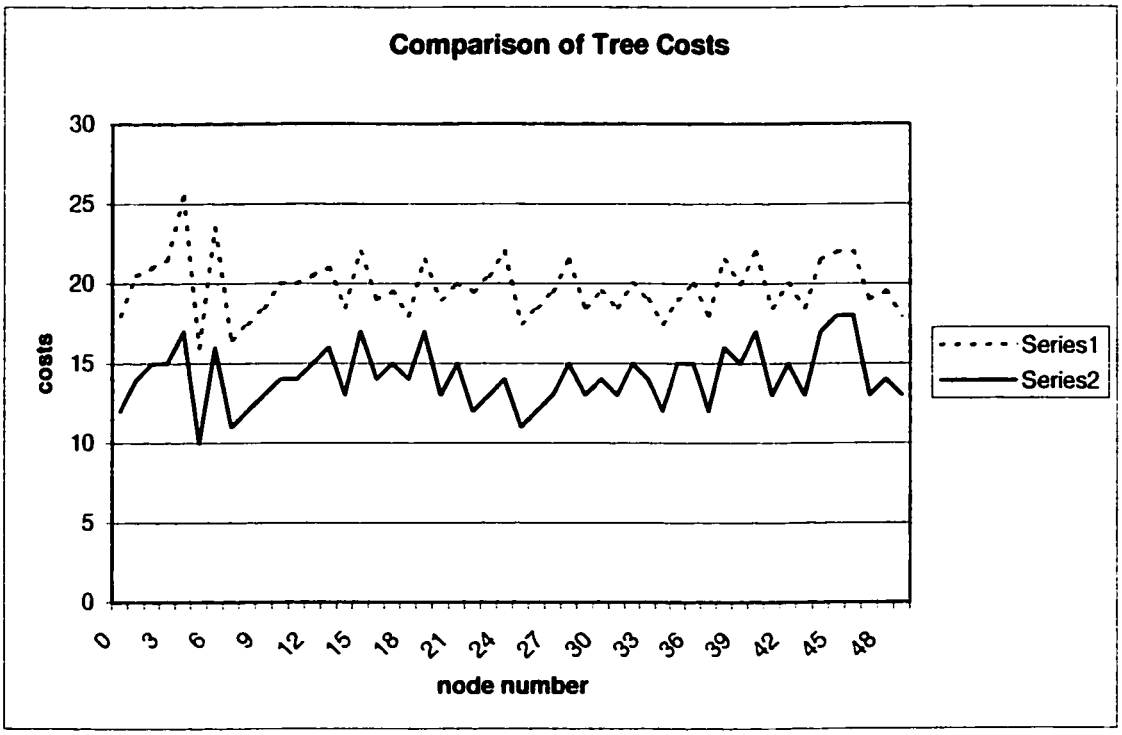


Series 1 - estimated cost of the multicast tree

Series 2 - actual cost of the multicast tree

Figure 16: Node and cost graph for Waxman Model

The above figure shows that for the Waxman model, the change in actual cost of the multicast tree taking different routers as the root of the tree and the change in estimated cost of the multicast tree taking different routers the root of the tree follows the same trend. Running the simulation with the Exponential model gives the results as shown in figure 17.

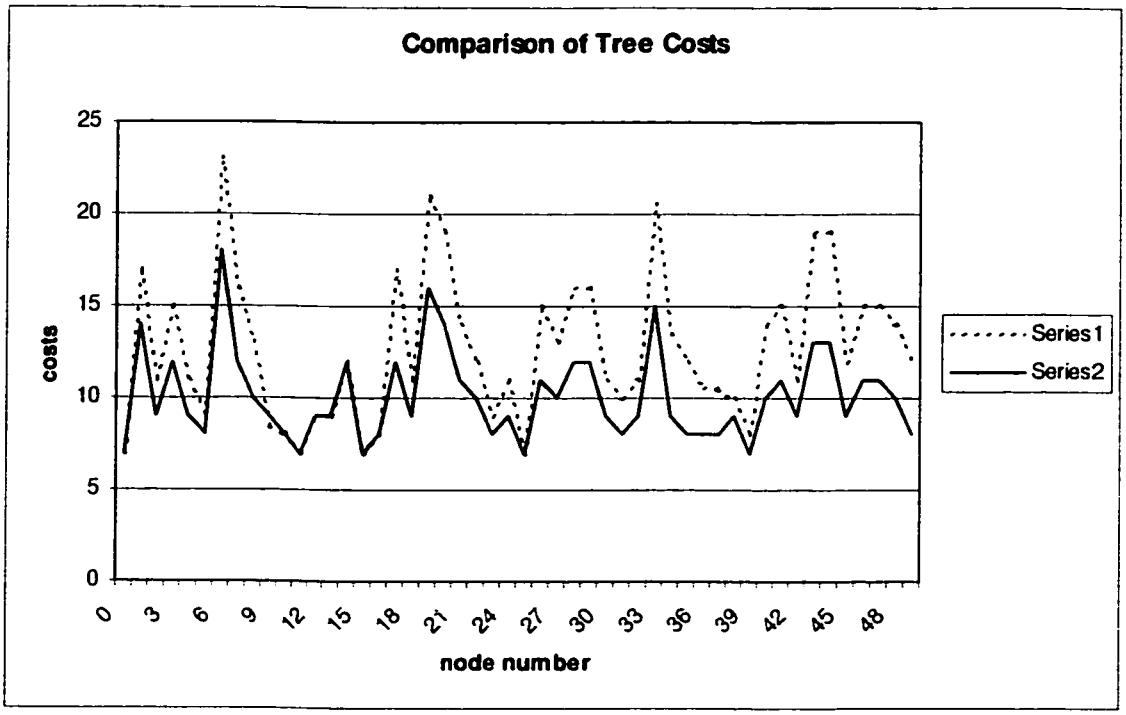


Series 1 - estimated cost of the multicast tree

Series 2 - actual cost of the multicast tree

Figure 17: Node and cost graph for Exponential Model

The above figure shows that for the Exponential model, the change in actual cost of the multicast tree taking different routers as the root of the tree and the change in estimated cost of the multicast tree taking different routers the root of the tree follows the same trend. Running the simulation with the Hierarchical model gives the results as shown in figure 18.



Series 1 - estimated cost of the multicast tree

Series 2 - actual cost of the multicast tree

Figure 18: Node and cost graph for Hierarchical Model

The above figure shows that for the hierarchical model, the change in actual cost of the multicast tree taking different routers as the root of the tree and the change in estimated cost of the multicast tree taking different routers the root of the tree follows the same trend.

5.2 SIMULATION TO MEASURE ACTUAL MULTICAST DATA DELIVERY TIME USING SPECIFIC MULTICAST SCENARIOS

In order to anticipate what the most common multicast applications will be, we need to examine the reason why one would use multicast.

The reason for using multicast is to overcome inefficient use of network resources such as: bandwidth, transmission time and computational resources. This becomes relevant when there is more than one participant in the session. The larger the number of participants, the larger the ability to save and the larger the volume of traffic, the more the crucial the problem becomes, and thus the savings are more predominant. For the purpose of this simulation we take a look at the application areas in which multicast is a potential asset and for each of those cases we run separate simulations. The following is a list of multicast application scenarios:

- Shared whiteboards
- Dynamic group applications
- Multi-party audiovisual conferencing
- Multimedia streaming/Pay per view
- Data distribution

This part of the simulation has been done using Network Simulator – ns2 (available from the Information Sciences Institute at the University of Southern California). NS is a discrete event simulator targeted at networking research. The simulation is viewed using Network Animator (NAM). NAM is a Tcl/TK based animation tool for viewing network simulation traces and real world packet traces. The PIM-SM protocol was implemented in ns2 for running simulations. Since it does not support dynamic changes, the simulation for RP relocation was done by issuing join and prune messages from the appropriate nodes and setting up a new group with the RP at the best position.

The simulation scripts have been written using Tcl. For all the simulations in this section we consider network links to be bi-directional with a bandwidth of 1.5Mb and delay ranging from 100ms to 500ms chosen at random. The simulation builds the network according to the hierarchical model and chooses an RP at random. It then makes sources and receivers according to the scenario. We measure the time taken to transmit a 3 MB file to the all the receivers without relocation. We then perform RP relocation and measure the time taken to transmit the same amount of data to all receivers. The measured time in this case also includes the time required to relocate the RP from the original position to the new calculated position.

5.2.1 SCENARIO 1: SHARED WHITEBOARD

A shared whiteboard is a software application that allows users to participate in distributed meetings. Each participant in the meeting uses a software application that displays a common whiteboard. Whenever someone writes or draws on the whiteboard, the software multicasts the data to all other participants in the meeting. Thus the number of sources and receivers in this case is small. They are situated fairly close to each other geographically. The data to be transmitted is small and intermittent.

5.2.2 SCENARIO 2: DYNAMIC GROUP APPLICATIONS

Dynamic group applications are those applications where the number of sources and receivers are not fixed but are higher than those compared to shared whiteboard applications. For the purpose of this simulation we consider only one-way non-interactive transmission. Distance learning applications are a typical example used to bring together expert instructors and large number of students. The number of sources in this case is small and the number of receivers is fairly large. They are situated far away from each other geographically. The data to be transmitted is high (considering one-way non-interactive video) and continuous.

5.2.3 SCENARIO 3: MULTI-PARTY AUDIO VISUAL CONFERENCING

Multi-party audiovisual conferencing allows users to participate in two-way interactive audio and video conferencing. Examples of such applications are corporate meetings, conferences, etc. The number of sources and receivers in this case is fairly small. They are situated far away from each other and the data to be transmitted is high and continuous.

5.2.4 SCENARIO 4: MULTIMEDIA STREAMING/PAY PER VIEW

Steaming multimedia applications or pay per view applications are those in which the user can watch audio and video over the network. The number of sources in this case is low and the number of receivers is high (example of such applications are users watching live football games or movies). The data to be transmitted is high and continuous.

5.2.5 SCENARIO 5: DATA DISTRIBUTION

Data distribution applications are those that require extremely large amounts of data to be transmitted to large number of receivers. Examples of such applications are software distribution, content distribution (sports, weather, etc.), finance (stock tickers), News, etc. The number of sources in this case is low, the number of receivers is extremely high, the receivers and sources are widely spread out and the data to be transmitted is high.

Using these multicast scenarios and finding the time required to multicast a 3MB file before and after RP relocation leads us to following results:

Scenario	No. of sources	No. of receivers	Time taken without RP relocation (sec.)	Time taken with RP relocation (sec.)	Percentage Improvement in time (%)
Shared Whiteboard	7	7	6.9000	4.8125	**30.2**
Dynamic Group Applications	2	35	7.2462	5.5465	**23.4**
Multi-party Audio Visual Conferencing	14	14	7.5463	5.2587	**30.3**
Multimedia Streaming/Pay per view	1	65	6.6573	5.1255	**23.0**
Data Distribution	1	98	6.7979	5.4485	**19.85**

Table 1 : Summary of simulation results for various scenarios

The number of sources and receivers in each of the scenarios was chosen depending on the reasoning given in each scenario. In each of the scenarios we see that with RP relocation there is approximately a 20% improvement in time taken for multicasting a fixed amount of data.

CHAPTER 6

CONCLUSION

The goal of this thesis was to establish the need for RP relocation in Protocol Independent Multicast - Sparse Mode and to devise an RP calculation and relocation mechanism. It is clearly established that random selection of the RP does not result in efficient placement of the RP thus resulting in delay and traffic congestion. Also it is not possible for the network administrator to determine beforehand where the sources and receivers of a multicast group should be located. The simulation clearly demonstrates that the RP calculation algorithm suggested results in lower cost of the multicast tree thus saving network resources and avoiding delay. An RP relocation mechanism was also designed. This relocation mechanism ensures that no data are lost during relocation.

Some other interesting results are also drawn from the simulation. They are:

- The need for RP relocation increases as the number of sources increases in the multicast group.
- The need for RP relocation increases as the connectivity of the network decreases.
- The need for RP relocation increases as the number of nodes in the network increases.

These factors can be used to determine the threshold value. If the estimated improvement in cost is above this threshold then the RP should be relocated to the new position in order to reduce the cost of the multicast tree. The simulation results also show that there is approximately a 20% improvement in time taken to multicast a fixed amount of data with RP relocation.

Thus, we believe that the efforts in this thesis have gone towards successfully establishing the need for RP relocation and providing a RP calculation and relocation mechanism for Protocol Independent Multicast – Sparse Mode.

6.1 FUTURE WORK

We believe that future work in this direction should involve thorough testing of the RP calculation and relocation mechanism. This should include having a test bed and real world Internet traffic simulation and packet drops. This would give an idea about the value of threshold for relocation.

BIBLIOGRAPHY

[1] Bill Fenner, Mark Handley, Hugh Holbrook, Isidor Kouvelas, "PIM-SM, Protocol Specification (Revised)", Internet Draft, Work in Progress, draft-ietf-pim-sm-v2-new-04.ps

[2] Bill Fenner, Mark Handley, Roger Kermode, David Thaler, "BSR Mechanism for PIM Sparse Mode", Internet Draft, Work in Progress, draft-ietf-pim-sm-bsr-02.ps

[3] J. William Atwood, Ritesh Mukherjee, "RP Relocation in PIM-SM Multicast", Internet Draft, Work in Progress, draft-atwood-pim-sm-rp-01.txt

[4] David G. Thaler, Chinya V. Ravishankar, " Distributed Center Location Algorithms", IEEE Journal of Selected Areas in Communications, April 1997

[5] Ying-Dar Lin, Nai-Bin Hsu, Ren-Hung Hwang, "RP Relocation extension to PIM-SM Multicast Routing", Internet Draft, Work in Progress, draft-ydlin-pim-sm-rp-01.txt

[6] Michael R. Garey, David S. Johnson, "Computer and intractability: A guide to the theory of NP–completeness", June 1988

[7] Kevin Fall, Kannan Varadhan, "The ns Manual", August 2000

[8] Robert Voigt, Robert Barton, Shridhar Shukla, "A Tool for configuring Multicast Data Distribution over Global Networks", April 1995

[9] PIM-SM Multicast Routing Protocol, White Paper, Microsoft Windows 2000, December 21, 1999

[10] Ellen W. Zegura, Ken Calvert, S. Bhattacharjee, "How to Model an Internetwork", Proceedings of IEEE Infocom '96, San Francisco, CA

[11] Ellen W. Zegura, Kenneth L. Calvert, Michael J. Donahoo, "A quantitative comparison of graph-based models for Internet topology", IEEE/ACM Transactions on Networking, December 1997

[12] Christophe Diot, Walid Dabbous and Jon Crowcroft, "Multipoint Communication: A Survey of Protocols, Functions and Mechanisms", IEEE Journal on Selected Areas in Communications, Vol. 15, No. 3, April 1997

[13] Ganesh Ramasivan, "Enhanced Communication Services for many-to-many Multicasting using XTP", High Speed Protocols Laboratory, Department of Computer Science, Concordia University, March 2000