A new algorithm for RP selection in PIM-SM Multicast Routing

Xiao bo Dong

A Major Report

in

The Department

of

Computer Science

Presented in Partial Fulfilment of the Requirements
for the Degree of Master at
Concordia University
Montreal, Quebec, Canada

August 2002

Abstract

The Protocol Independent Multicast-Sparse Mode (PIM-SM) protocol establishes a core-based tree to forward multicast datagrams in a network. In the previous works on the RP selection, an artificial cost function was always used to calculate the cost of the multicasting tree [10]. It can just get an approximate result comparing with the real tree cost. Consequently, it affected the accuracy of the RP selection obviously. Moreover, PIM-SM provisionally uses administrative selection or simple heuristics for locating the center of a group, but doesn't preclude the use of other methods that provide an ordered list of centers. As the sources and receivers in the multicast group join and leave the group, a bad result may occur in terms of high cost, delay, and overhead. In this report we investigate the problem of finding a good center in centralized fashion, propose a new algorithm for the real tree cost calculation function and a new algorithm for RP selection in PIM-SM Multicast Routing, and examine them in a virtual network environment. We also present simulation results to show that our new algorithms can provide better results than other algorithms.

Contents

List of Figures

Chapter 1

1 INTRODUCTION

1.1 Overview of Multicast

Multicast services have been increasingly used by various continuous media applications such as teleconferencing, distance learning, and voice & video transmission. Multicast methods typically use spanning trees, and minimize delay by distributing packets along the shortest path between a receiver and a sender. The Steiner-tree-based problem is a good example. It aims to minimize the total cost of a multicast tree, and is known to be NP-complete. Some heuristic algorithms were proposed to get an approximate result. Currently, building a multicast tree can be done with two approaches: the Source-Based Multicast Tree Approach and the Core-Based Multicast Tree Approach. A source-based tree is a tree rooted at a source which would send data packets to all group member and connects to every member in the multicast group. DVMRP [9], MOSPF [7], PIM_DM [4], and EXPRESS [6] fall in the category of source-based trees. A core-based tree is a tree rooted at a core which is a node selected with some mechanism and spans all the group members. CBT [2], PIM-SM [3], and SM [8] are representatives of core-based trees.

From the viewpoint of network management, the Core-Based Tree approach offers more favorable scaling characteristics than the Source-Based Tree approach by a

factor of the number of active sources. We don't need to build the minimum spanning tree for each of the source. Rather, we just build the multicasting tree for one time. Whenever a group member comes and goes, we just simply add or remove a branch to or from the existing tree. Therefore, a router does not have to maintain information about each source for each group and needs a single entry for each group instead. Moreover, for the Core-Based Tree approach, only on-tree routers have to maintain the group membership information. Comparing with the source-based tree approach, the core-based tree approach can provide more flexibility and scalability.

However, at the same time, the Core-Based Tree approach also consumes a lot of network resources. It builds the reserved shortest path from each group member to the current core to create the core-based tree. The number and the location of the group member don't affect the process of building a core-based tree. On the other hand, a source-based tree has to find a spanning tree with the minimum cost. Therefore, a topologically centered tree gives a delay bound of twice that of the source-based tree. If the root moved to a group member, the bound becomes three times that of the source-based tree [10].

In PIM-SM, whenever new members want to join a group or group members want to leave, they use an explicit join/leave message that propagates hop-by-hop from the member's local Designed Router (DR) toward a core, called Rendezvous Point (RP), of the distribution tree. The RP dominates the specific multicast group(s), and

facilitates the joining/leaving of the group members. 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, as the group members come and go, the current RP may have a higher tree cost when it is located far away from the sources and its group members. In order to select a best RP in terms of the tree cost for the current environment, it is necessary for the RP to be dependent on the sources and receivers and be relocated dynamically in the multicast group at any point in time. This report proposes a dynamic RP selection mechanism that is an extension to the PIM-SM multicast routing protocol.

1.2 Issues And Motivations

Protocol Independent Multicast-Sparse Mode (PIM-SM) [3] is the most widely used Core-Based Multicast routing protocol. In PIM-SM, there is only a single active RP for each multicast group. A receiver that wishes to join a multicast group contacts (via IGMP query/report messages) its directly attached router. The local router then creates a forwarding cache for the (*, group) pair and explicitly joins the distribution tree by sending a unicast PIM-join message to the group's RP along the shortest-path from the receiver to the RP. An intermediate router forwards the PIM-join message and creates the (*, group) pair. When a source host first transmits a multicast packet to a group, the local router encapsulates the packet in a PIM-register packet and unicasts it

to the RP. The primary RP then transmits a PIM-join message back to the source router. Upon receipt of the PIM-join message from the RP, the source router then ceases to encapsulate data packets in PIM-register but forwards them in the native multicast format to the RP (Fig 1.1).

In a widely dispersed group, any RP will, of necessity, be remote from many of the group members, and the paths for many group members will be much longer than the least-cost path. To help alleviate these drawbacks while maintaining the benefits of the PIM scheme, PIM allows a destination router to replace the group-shared tree with a shortest-path tree to any source. We call this phenomena tree shift (Fig 1.2). Once a

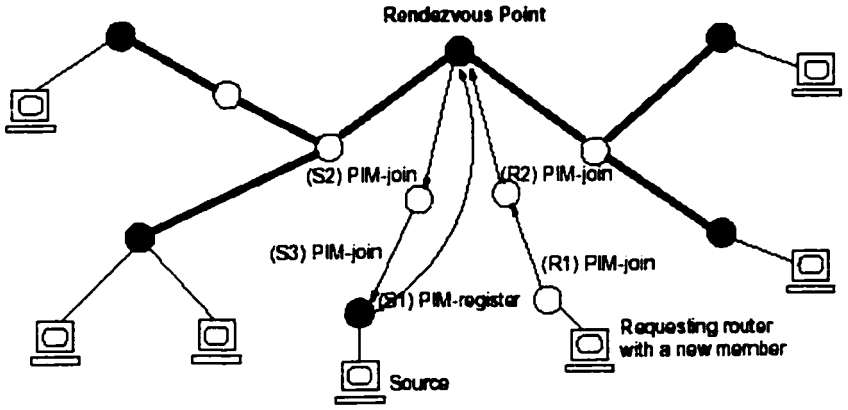


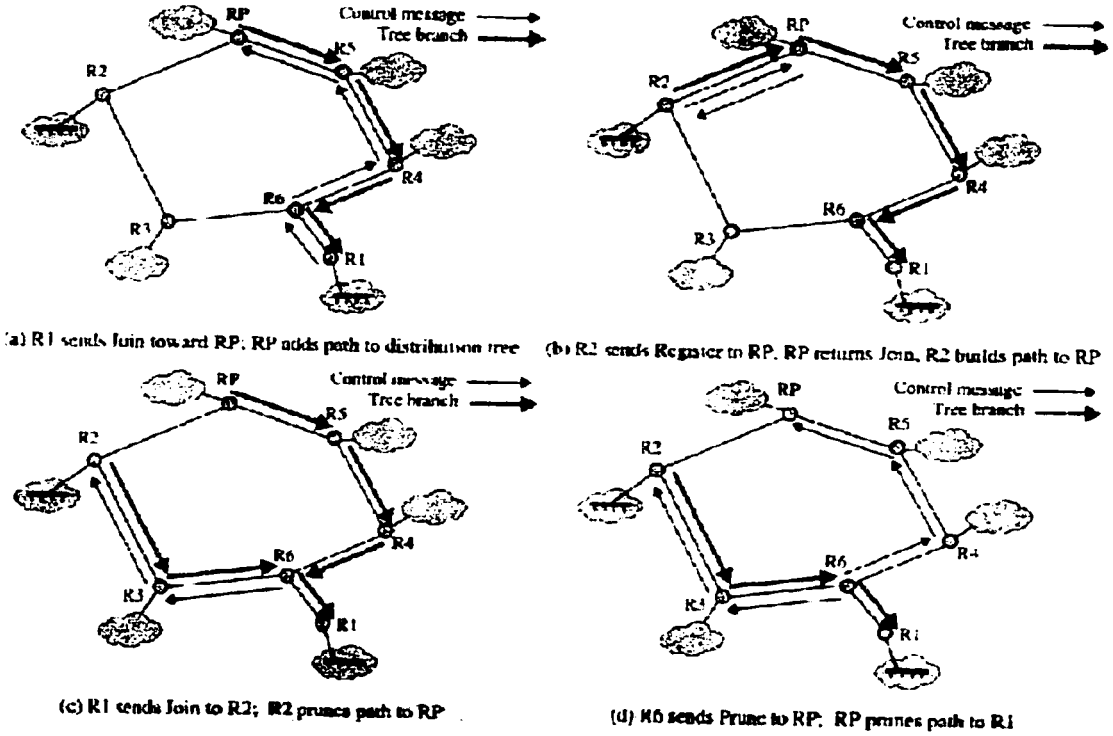
Fig 1.1 Member joining procedure in PIM-SM

destination router receives a multicast packet, it may elect to send a Join message back to the source router of that packet along a unicast shortest path. From then on, multicast packets between that source and all group members that are neighbors to that destination router follow the unicast shortest path. At this point the destination router will be receiving two copies of the packets – one from the shortest-path tree and

another from the multicast tree. While the destination begins to receive packets from the source by the shortest-path router, it sends a Prune message to the RP. This Prune message instructs the RP not to send any multicast packets from that source to this destination. The destination will continue to receive multicast packets from other sources via the RP-based tree, unless and until it prunes those sources. Any source router must continue to send multicast packets to the RP router for delivery to other multicast members [3].

So far, we have seen how the PIM-SM works. Also, we have to know that the precondition for that scenario is that the multicasting tree has already been built. Steiner tree is a well-known example of finding a minimum multicast tree. Given a list of static network nodes, we can find a spanning tree with the minimum tree cost. However, in the real network environment, for the multicast tree in PIM-SM, the set of sources and the set of group members are dynamically changed. That is, after sources and/or group members come and go, the least cost multicast tree may be different. Therefore, considering a single group in which no member switches over to the shortest path tree (SPT), there is a set of multicast trees, {T0, T1, T2, ...}, where Ti is a multicast tree. Since the tree Ti is rooted at a core RP_i, the problem of finding the sequence {Ti} is reduced to the problem of finding the sequence {RP_i}, where the RP_0 is the initial hash RP. In this report, assuming RP_i is the current RP of G, we propose a new algorithm for RP selection which selects a list of potential RPs when the tree cost reduction, TreeCost (RP_i) - TreeCost (RP_{i+1}), is larger than a

pre-defined



(a) R1 sends Join toward RP; RP adds path to distribution tree (b) R2 sends Register to RP. RP returns Join. R2 builds path to RP

(c) R1 sends Join to R2; R2 prunes path to RP (d) R6 sends Prune to RP; RP prunes path to R1

Fig 1.2 Multicast Tree shift to Shortest Path Tree

threshold, q.

After we select a list of RPs, we have to let all nodes in the domain know that. In order to broadcast the set of candidate RPs to the network nodes, the bootstrap router (BSR) is elected for the domain. BSR originates Bootstrap messages to distribute the set of RPs information, which are distributed hop-by-hop throughout the domain. There is only one RP-set per PIM-SM domain. By using a hash function, each router can uniquely map a group address G to one of the routers in the RP-set. That is, a candidate RP is chosen as RP if it yields highest hash value.

Chapter 2

2 RP SELECTION MECHANISM

2.1 Previous Work

A number of methods have already been proposed for center location [10]. The main difference between those methods is the node set from which they select the best node as the new RP. In this section we present a brief overview of such methods and their performance.

Ideally, we use all nodes in the network as the candidate RPs and get the best one in the domain. This method can be called "Optimal Center-Based Tree" (OCBT) [10]. It chooses the RP by calculating the actual cost of the tree rooted at each node in the network, and picking up the one that gives the lowest cost.

In the Random Source-Specific Tree (RSST) heuristic, the center is chosen randomly among the sources and does not move. It is exactly what PIM [3] does to select an RP for a group by selecting the first source or the initiator of the multicast group. This approach only gives a single RP, rather than a list of possible RPs which is required for fault tolerance.

Wei and Estrin [11] show that the Minimum Shortest Path Tree (MSPT) approach performs almost as well as OCBT, and suggest that it is adequate for use with center-based trees. It calculates the actual costs for the trees rooted at each group member, and chooses the member with the lowest cost. It reduces to OCBT when all nodes are group members.

Recently, in [1] an approximation algorithm was introduced based on a method from [10]. We use this result as a reference for comparison with our algorithm. The input for the referenced algorithm is the current RP and the output is a node that has a significantly less cost than the current RP. First of all, the algorithm gets the current RP's cost by using an estimated cost function, saves it to two variables: CurrentCost and RPcost, and sets the current RP as the CurrentNode. And then, it checks the cost of all neighbors of the CurrentNode, selects the lowest one among them, and compares it with the CurrentNode. If the lowest cost is less than the CurrentCost, it assigns the lowest cost to the CurrentCost, sets the node that has the lowest cost as the CurrentNode, and checks all neighbors of the CurrentNode again. The algorithm stops and compares the CurrentCost with Rpcost until the lowest cost for the CurrentNode is no less than the CurrentCost. If the difference is greater than a predefined threshold, the algorithm returns the CurrentNode as the potential RP; otherwise, the algorithm returns NULL which means there is no improvement for other nodes in the network and there is no need to move the current RP.

Although the referenced algorithm may sometimes get the right answer, it still needs improvement. One problem is that it uses iteration fashion to get the best RP. The stop condition for the iteration is that there is no improvement for the node that is currently being checked. If the number of nodes in the network is small, this algorithm would work very well because the number of steps to get to the stop condition is limited. However, in the real world, there is usually a huge batch of nodes in the network domain. The algorithm would need to spend a lot of steps to get the answer. Moreover, the computation to calculate the real cost for the current node would increase tremendously as the number of nodes increases. Another thing is that the algorithm just selects one neighbor of the CurrentNode with the lowest cost to be the candidate for the next CurrentNode, which could miss the real best RP when it is not a neighbor of the node with the lowest cost. It is possible that the cost of one node is less than another even though its precursor's cost is greater than another's. If that is the case, that algorithm can not get the right answer. Therefore, the referenced algorithm is impractical in a large network environment and can not get the best RP in some cases.

In the next section, we propose our dynamic algorithm for the RP selection which can solve the problems above properly.

2.2 Our new algorithm for RP Selection

Some previous works for the RP selection used an estimated cost function to calculate the tree cost [10]. The estimated function is defined as follows:

Let S be the set of sources for the multicast group, u a source in S, 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,

$$\text{Est. Cost(min)} = \max_{u \in S} d(root,u) + \text{number of duplicate distance nodes in S}$$

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 is 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,

$$\text{Est. Cost(max)} = \begin{cases} \text{sum } d(root,u) & \text{if } |S| <= deg(root) \\ \text{sum } d(root,u) - (|S| - deg(root)) & \text{otherwise} \end{cases}$$

where $u \in S$

The final estimated cost function is given by,

$$\text{Est. Cost} = (\text{Est. Cost(min)} + \text{Est. Cost(max)})/2$$

The problem for this estimated cost function is that it uses just the upper and lower bounds to approximate the multicast tree cost. Because of the complexity of the real network environment, this estimated cost function can not give the right real tree cost. Consequently, the RP selection algorithm will not find the real best RP. Here, we introduce another algorithm to get the real tree cost in order to increase the accuracy of the RP selection.

The algorithm that we use to get the real tree cost is called reserved path algorithm (RPA) (Fig 2.1). First of all, we define the real cost of a tree to be the sum of the costs of the links in the multicast tree. Assume the cost of every link to be 1, in which case, the tree cost is the number of links in the tree. Let S be the set of multicast group members. We define the following weight function for a given set ***S*** and ***root***:

Actual Cost (T) = number of links in T, where T is the tree rooted at *root* and spanning all of *S*

To get the real cost of the given node, the algorithm 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).

```
Cost=0;
For each of the group member do
{
    set group member as CurrentNode
    Do while( CurrentNode <> RP)
    {
        check whether the CurrentNode is On-Tree
        if (yes)
        {
            terminate Do while loop;
        }
        cost=cost+1;
        set CurrentNode as On-Tree node
        set the precursor of the CurrentNode as the
    CurrentNode;
    }
}
```

Fig 2.1 Reserved Path algorithm (RPA)

Second, in PIM-SM, once a node wants to join the multicast group, a tree branch will be built from the node to the current RP by using the shortest path. Therefore, we use the reserved path (from group member to RP) to calculate the actual tree cost.

Third, the problem for calculating a real tree cost is that there are some links shared by

a set of group members on the multicast tree. In order to avoid counting the same link several times, we categorize two kinds of nodes:

- On-Tree nodes, the ones that are already counted the link forwarding to RP in the Current Tree Cost
- Off-Tree node, the ones that are going to be counted and to become an On-Tree node.

At the beginning of counting the real tree cost, all the nodes lying on the multicast tree are Off-Tree nodes. Once an Off-Tree node is counted, it is marked as an On-Tree node, which means it won't be counted again. Finally, all the nodes become On-Tree nodes and we get the real tree cost without any duplication.

In order to get the tree cost, we have to add all the number of links from any group member to current RP together. The worst case for the given network environment is that all the group members have the furthest distance. So, the complexity of RPA is ***O (K*D)***, where k is the number of group members and D is the furthest distance from any group member to current RP.

So far for each node in the network, RPA can calculate the real tree cost rooted at the node itself and know how to reach any other node in the same network domain. Next, we will present our RP selection algorithm (Fig 2.2):

1. Given the current RP, the algorithm sets it as the CurrentNode that is going to be checked, gets the tree cost rooted at CurrentNode, and

saves the cost as RPcost.

2. The algorithm checks the tree costs of all neighbors of the CurrentNode,

```
FindRP (Current RP)
{
    set current RP as CurrentNode;
    get the cost of current RP and save it as Rpcost;
    set LowestCost = ∞;
    loop M times do
    {
        check the cost of all neighbors of CurrentNode;
        select N nodes with lowest costs;
        set those N nodes as following CurrentNodes;
        if lowest cost is less than LowestCost, save it as LowestCost and
        put the node in a nodelist;
    }
    if  (the cost of nodelist – LowestCost) is greater than a predefined
        threshold
        returns nodelist with those nodes;
    else
        returns NULL;
}
```

Fig 2.2 Our new RP selection algorithm

selects M of them with the lowest cost as the next checking nodes and compares their costs with a variable LowestCost which records the lowest cost so far the algorithm found. (Note: the LowestCost is

originally set to an infinite number).

3. If one of those costs is less than LowestCost, the algorithm sets the cost as the new LowestCost, saves that node in a nodelist, which will be used as a set of potential RPs, and shifts all nodes in the nodelist according to the cost.
4. The algorithm goes to step 1 with those nodes set as CurrentNodes and repeats step 1 N times.
5. After running N times, the algorithm compares the cost of all nodes in the nodelist one by one with the RPcost. If the difference is greater than a predefined threshold, that node will be a candidate of potential RP. Finally, the algorithm returns the nodelist whose nodes have a meaningful cost improvement compared with the current RP. If there is no such improvement, the algorithm returns NULL signaling that it is not necessary to move the current RP.

There are two values that are not determined in our algorithm, M and N, which can solve the accuracy and overhead problems we mentioned before. First of all, let us consider M, the number of neighbors that the algorithm selects with the lowest cost as the next CurrentNode. Even though it is possible that one node has less cost than another while its precursor's cost is greater than the other's, as the region of the nodes that the algorithm is going to check becomes increasingly bigger (M is increasing), the accuracy that the algorithm can select the real best node increases. Note that

there is a tradeoff between the accuracy and the complexity that is spent on those extra neighbors. The simulation result will show us what is the ideal value for M balancing the accuracy and complexity.

Second, let's consider N, the number of steps that the algorithm executes. As we mentioned before, as the number of nodes in the network domain increases, the reference algorithm has to spend many more steps to get the answer, which leads to a great deal of extra consumption. Therefore, we limit the number of steps that the algorithm executes so that the algorithm will be terminated after it is executed N times. There is a tradeoff between the accuracy and the complexity that is spent on those steps. If we spend few steps in the algorithm, the complexity of the algorithm decreases, however, we may not get the node with real lowest cost in the network domain.

Moreover, our algorithm returns a list of potential RPs instead of one with the lowest cost. The RP discovery is done using BSR (Bootstrap Router) mechanism [5]. The active BSR for a network domain includes a set of potential Candidate-RPs (the RP-Set), along with their corresponding group prefixes, in Bootstrap messages and broadcasts it periodically. Bootstrap messages are distributed hop-by-hop throughout the entire domain. All the PIM (Protocol Independent Multicast) routers in the domain receive and store Bootstrap messages originated by the BSR. Each router continues to use the contents of the most recently received Bootstrap message from the BSR

until it accepts a new Bootstrap message. When a DR (Designated Router) receives an indication of local membership (typically from IGMP (Internet Group Management Protocol)) or a data packet from a directly connected host, it sends a Join or Register-encapsulates message towards the RP which is selected from the Candidate_RPs provided by the BSR message. The basic function of the BSR mechanism is to provide robustness in case of RP failure. If the current active RP is crashed, the BSR router will delete the current RP in the RP-set, shift all rest candidate-RP with proper priority, and send it to all PIM-SM routers in the network domain. And then, all the routers know the current RP crashing and switch to the new RP thatthe Bootstrap message provides. So, in our algorithm, we use a list of potential RPs as the input of the BSR router and support the failure recovery mechanism of BSR. Another reason that we use a list of potential RPs is that not all the routers in the network domain are RP-capable. If the first node in the nodelist is not RP-capable, we can shift it to the second node which is RP-capable in order to avoid returning a useless result for the current network.

Chapter 3

3 SIMULATION

In our simulations, all links were symmetric with unit cost, so that the tree cost is simply the total number of links in the tree. For the purpose of constructing trees, we also assume that all sources are also group members.

3.1 Generating Random Graphs

Graphs are commonly used to model the structure of networks, for the study of problems ranging from routing to resource reservation. The networking literature contains a variety of random graphs used to model networks. All are variations on the standard random graph model that distributes vertices at random locations in a plane and then considers each pair of vertices; an edge is added between a pair of vertices with probability p. We call this the Pure Random model or simply the Random model [12].

The most common random graph model is the one proposed by Waxman [13], with the probability of an edge from u to v given by:

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

Where $\alpha > 0$, $\beta \leq 1$ are parameters of the model, d is the Euclidean distance from u to

v, and L 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.

Exponential model is another random graph model intending to relate edge probability to distance between vertices (as in the Waxman model), but with more straightforward probability functions. The exponential model uses:

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

Where $\alpha > 0$ is the parameter of the model, d is the Euclidean distance from u to v, and L is the maximum distance between any two nodes. The probability of an edge in this model decreases exponentially with the distance between the two vertices.

In our simulation, we use those two models as our basic network graphs and the algorithms will be executed in those network virtual environments. Ellen [12] shows the relationship between the parameters selection and edge connection in the network graph. Typically, we set $\alpha = 0.2$, $\beta = 0.15$ for Waxman model, and $\alpha = 0.06$ for Exponential model.

3.2 Our Simulator

We developed a simulator in Visual C++ under Windows 2000 platform. There are a list of parameters about the network topology and our algorithm that the user can set

up: the number of nodes in the network, the number of sources for the multicasting, the model of the network (Waxman and Exponential) , the ID of current RP for the multicasting, the number of neighbors that our algorithm is going to check for each step, the number of steps the algorithm is going to execute, and the probability that the user can adjust to affect the connection of the edge in the graph. Meanwhile, the simulator can show the routing table for each selected node to the user, and the user can save the current topology to a text file in order to record the current network situation for the future investigation. The simulator works with the following steps:

1. Taking the number of nodes that the user enters, say N, the simulator generates the X and Y coordinates for those N nodes by using a random number generating function and builds the edge between any of two nodes with different probability equations and parameters corresponding to the model type that the user selects (Waxman/ Exponential). The user can adjust the edge connection through changing the probability. For example, if the user wants to get a dense network, he can decrease the current probability; on the other hand, if the user wants to get a sparse network, he can increase the probability as well.

2. Taking the number of sources that the user enters, say M, the simulator randomly selects M of N nodes, and sets them as source nodes.

3. For each of the nodes in the network, the simulator generates the routing table one by one. We use Dijkstra's algorithm to build the link state table. The precondition for the Dijkstra's algorithm is that all nodes have already known their neighbors. The Dijkstra's algorithm works as follows:

   A) For the selected node, it checks its all neighbors and adds an entry to the routing table of those neighbors to record the next node to reach the selected node.

   B) Based on those neighbors, the algorithm checks their all neighbors one by one, adds an entry to the routing table of those following neighbors to record the next node to reach the selected node, and adds an entry to the routing table of the selected node in order to record the cost to reach these nodes temporarily. If two or more nodes can reach the same destinations with different costs, it will replace its record for that destination with the lowest cost.

   C) The algorithm repeats step B until all other nodes have an entry in the routing table of the selected node indicating the cost to reach them. At the same time, all the nodes except the selected node have an entry indicating the next node to reach the selected node in their routing table.

   D) Shift to another node and set it as the selected node, repeat step A

until all nodes in the network generate the routing table.

After the algorithm is done, each node knows the cost to reach any other nodes in the domain and the next node to reach those nodes.

4. Taking the ID of the current RP that the user enters, say R, the simulator randomly sets that node as RP.

5. The simulator runs our algorithm by passing the current RP to it. If the algorithm returns a list of candidate-RPs, it selects the first one in the list and shows the improvement to the user; otherwise, it shows nothing to the user to indicate that current RP is a good one and there is no need to move it.

The user can change any parameters about the graph or the algorithm in order to check the algorithm in different network situations or different multicast group situations keeping the same network topology.

3.3 Simulation Result And Analysis

Each simulation point reflects an average over 100 runs. The first simulation result will show the relationship between the number of sources and the tree cost for the Waxman Model (Fig3.1). Some of the parameters about the simulator are fixed for

that situation: the total number of nodes is 150, the number of neighbors is 2, the number of steps is 5, and the probability to build a link is 0.08. The range of the sources is from 1 to 100.

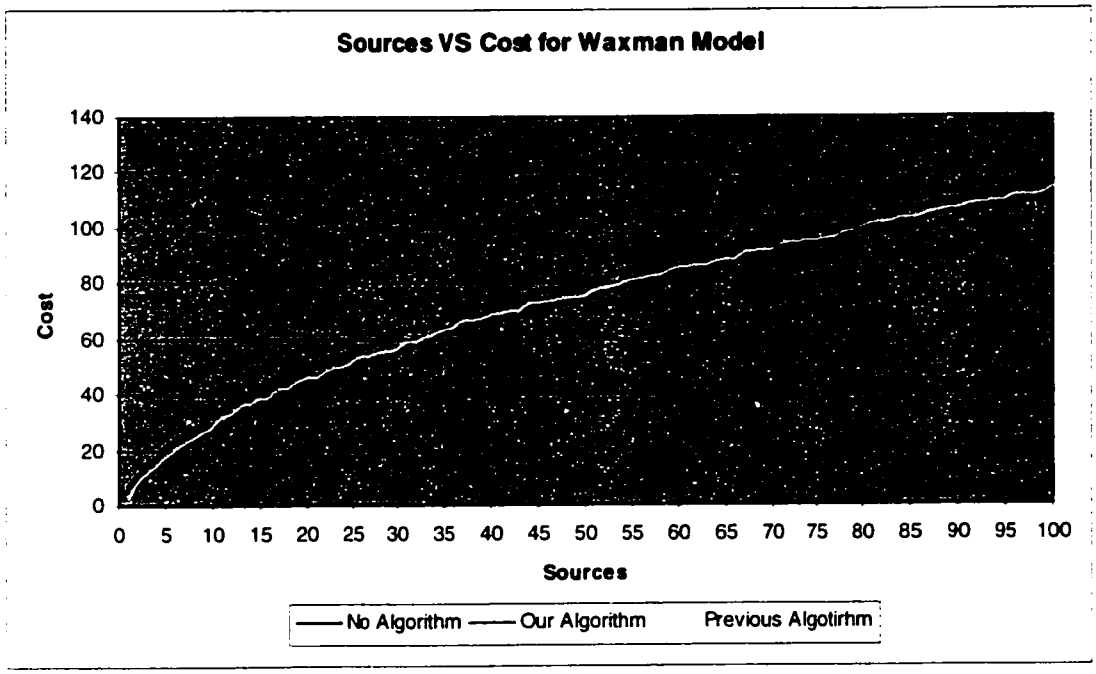


Fig 3.1 Sources VS cost for Waxman Model

We can see from Fig 3.1, as the number of sources increases, the cost of the multicast tree increases as well. We always can find a tree whose cost is less than both the tree rooted at current RP and the tree rooted at the node selected by the referenced algorithm. Because we fix the number of the steps our algorithm can run, we get an average of 5.149% cost improvement for each number of the sources while the referenced algorithm gets an average of 3.44%. Therefore, as the tree cost increases, the percentage improvement in cost decreases. Fig 3.2 shows the relationship between the number of sources and the percentage improvement in cost for Waxman model.

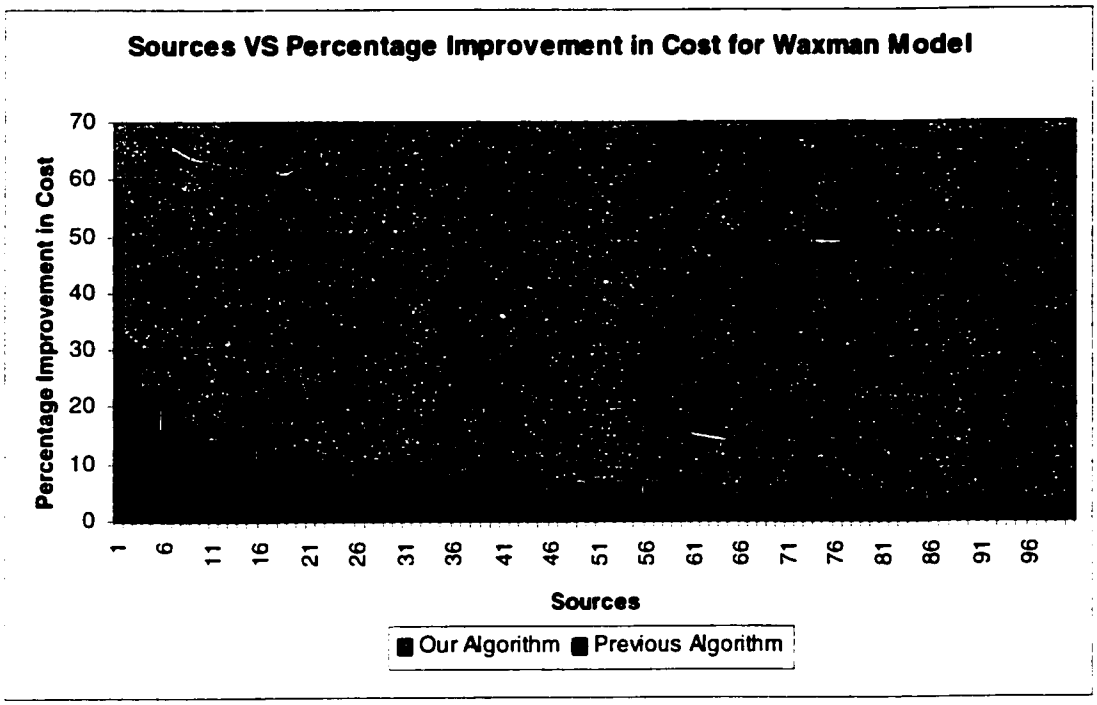


Fig 3.2 Sources VS Percentage Improvement in Cost for Waxman Model

In Fig 3.2, the average over the number of sources percentage improvement of our algorithm is 8.98% and the improvement of the referenced algorithm is 6.28%, which can show that our algorithm can provide 42.99% improvement over the referenced algorithm. As the number of sources increases, the percentage improvement goes down.

We run the same situation with Exponential Model. The relationship between the number of sources and the tree cost is shown in Fig 3.3. All the parameters of the graph are the same as in Fig 3.1, except the probability, which is equal to 0.06.

Again, as the number of sources increases, we always can find a tree whose cost is

less than both the tree rooted at current RP and the tree rooted at the node selected by the referenced algorithm. Our algorithm can get an average improvement of

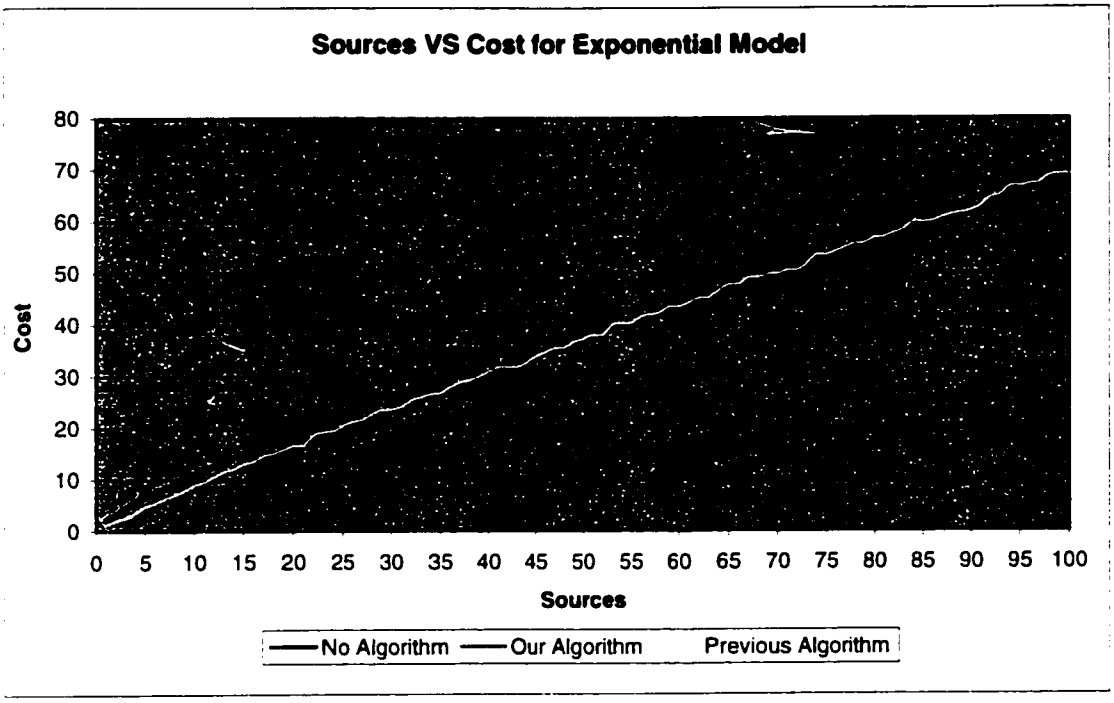


Fig 3.3 Sources VS Tree cost for Exponential Model

2.77% while the referenced algorithm can get 2.33%. However, the average improvement 2.77% is much less than Waxman Model's. Fig 3.4 shows the relationship between the number of sources and the percentage improvement in cost for Exponential Model.

We can see from Fig 3.4, as the number of sources increases, the percentage improvement decreases due to the fixed number of steps. Even though the average improvement for Exponential Model is much less than the one for

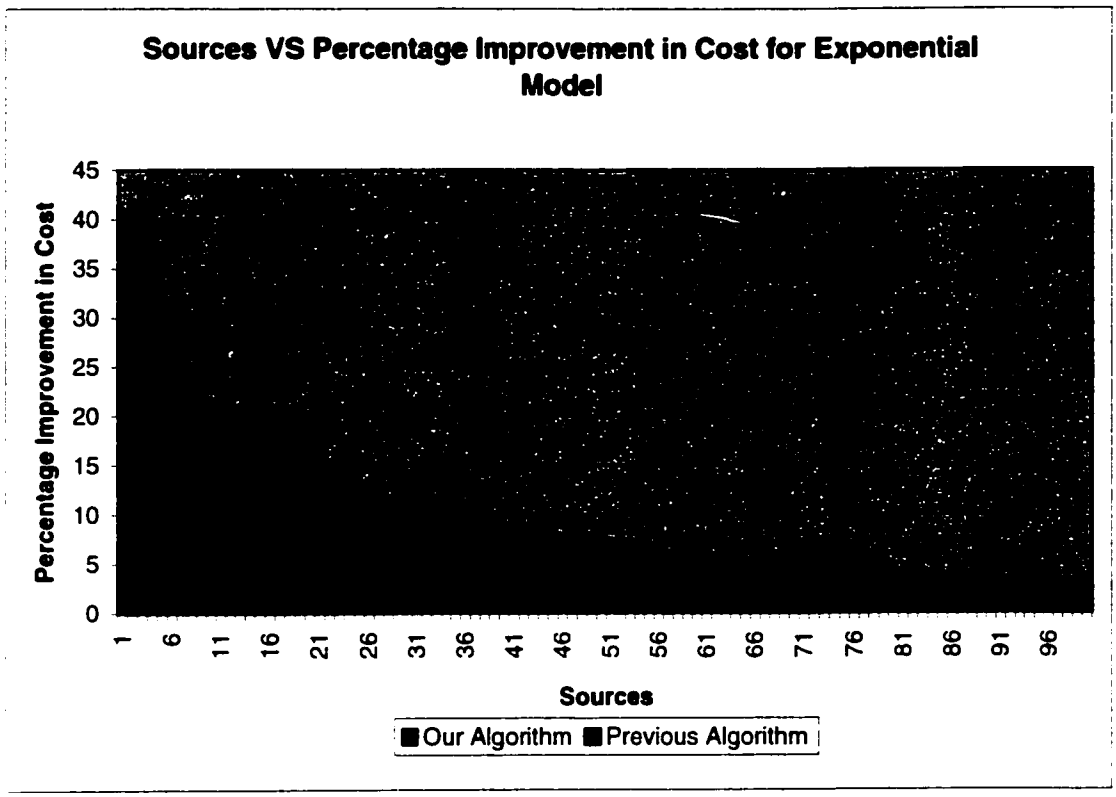


Fig 3.4 Sources VS Percentage Improvement in Cost for Exponential Model

Waxman Model, the average percentage improvement in cost, 10.77%, is almost the same as Waxman Model's 8.98%, which means that our algorithm can provide a constant cost improvement for different network topology. The average percentage improvement of the referenced algorithm for Exponential Model is 9.49%. We can see our algorithm gives 13.49% improvement over the referenced algorithm.

The probability is another important parameter for the network model. The relationship between the probability and the tree cost for the Waxman Model is shown in Fig 3.5. Some of the parameters about the simulator are fixed for this situation: the total

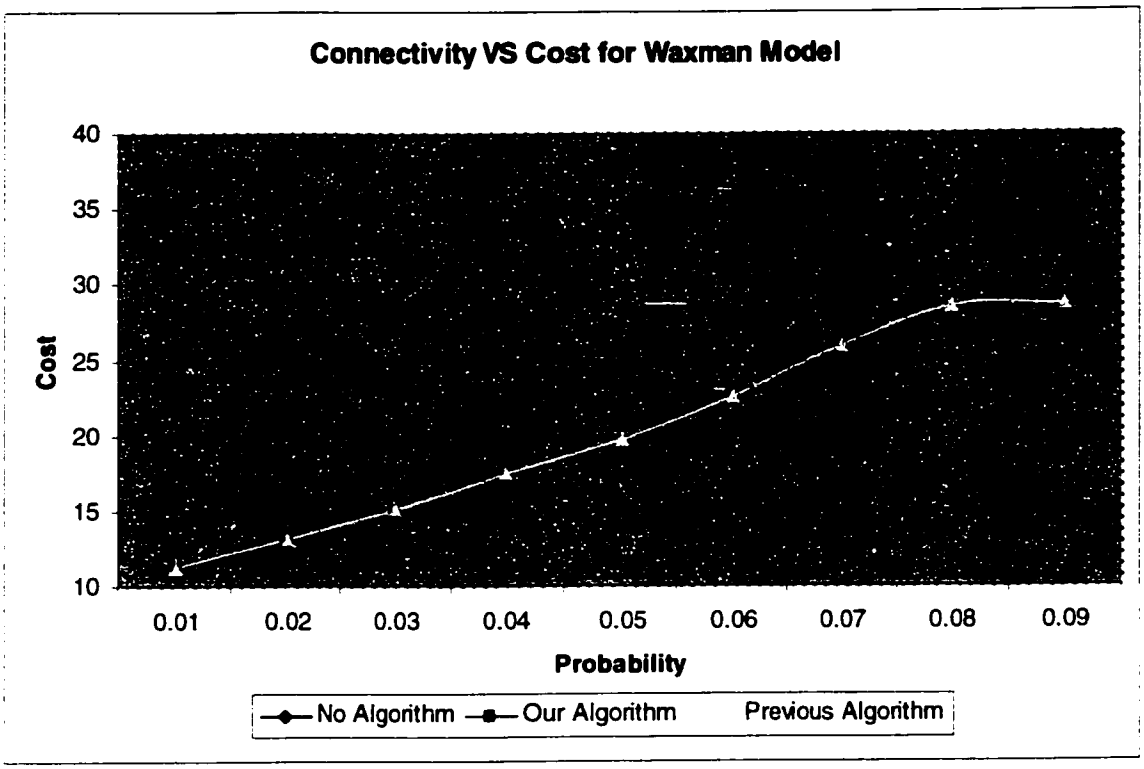


Fig 3.5 the relationship between the probability and cost for Waxman Model

number of nodes is 150, the number of sources is 10, the number of neighbors is 2, and the number of steps is 5. The range of the probability is from 0.01 to 0.1.

The probability of the Waxman Model is calculated by equation

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

The simulator asks the user to enter the threshold for the probability. If the probability calculated by that equation is greater than the threshold, there will be a link built between two random nodes. So the threshold is a crucial parameter affecting the network topology. The chance to build a link between any two nodes decreases as

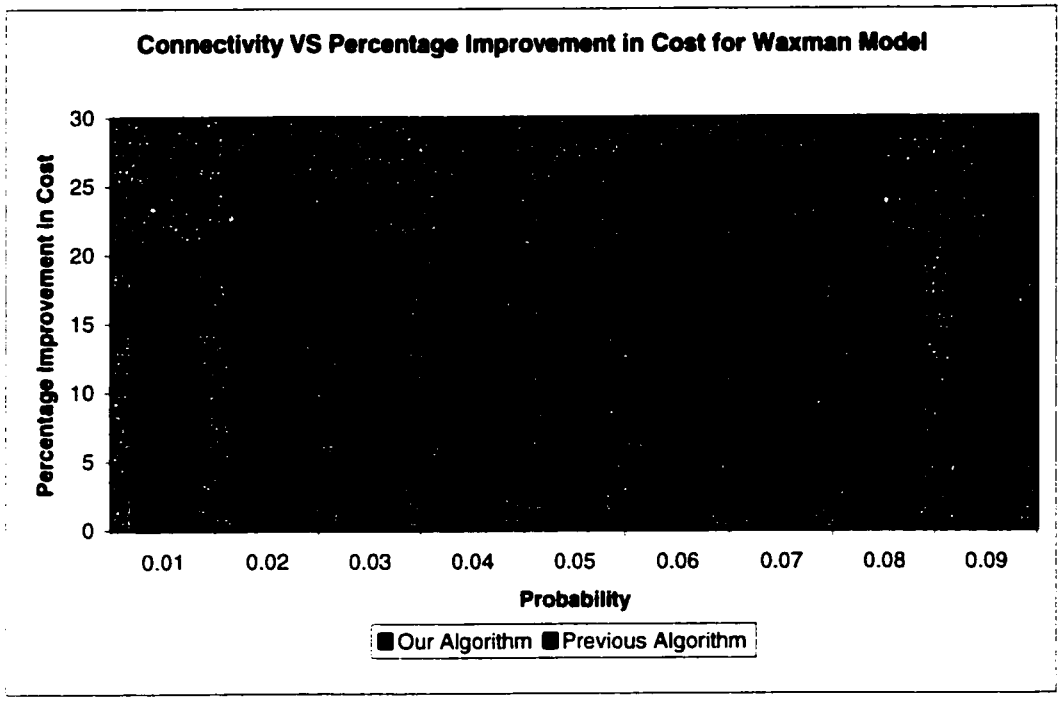


Fig 3.6 the relationship between the probability and percentage improvement in cost for Waxman Model

the threshold entered by the user increases, which means that the cost to reach any other nodes in the network increases. Therefore, the multicast tree cost increases. From Fig 3.5 we can see that our algorithm can get an average cost improvement of 5.37% while the referenced algorithm can get 4.39%. The percentage improvement in cost for Waxman Model is shown in Fig 3.6.

We can see from Fig 3.6 that each of the algorithms can give a fairly constant percentage improvement in cost for Waxman Model. The average percentage improvement of our algorithm is 22.19% while the one for the referenced algorithm is 18.61%, which can show that our algorithm can provide 19.24% improvement over the referenced algorithm.

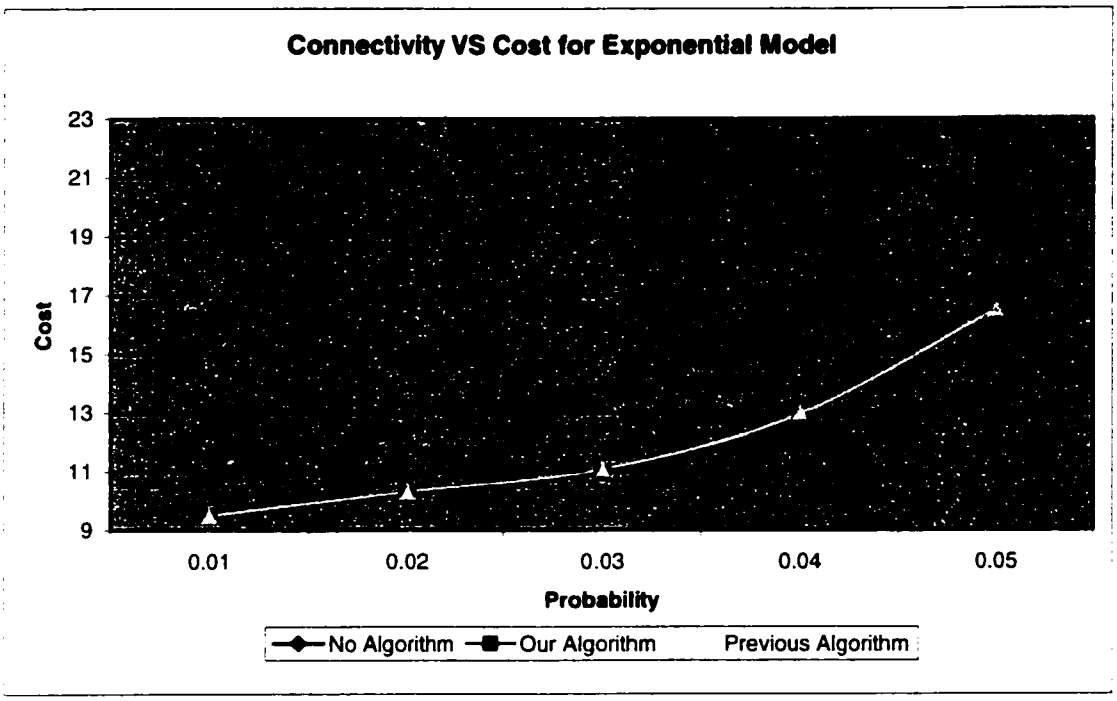


Fig 3.7 the relationship between the probability and cost for Exponential Model

We run the same situation with Exponential Model and get the simulation result in Fig 3.7. The probability of the Exponential Model is calculated by the equation

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

As the threshold increases, the tree cost increases and our algorithm can find a node whose tree cost is less than the tree rooted at the current RP and the tree rooted at the node selected by the referenced algorithm. Our algorithm can get an average cost improvement of 3.15% while the referenced algorithm can get 2.9%. The percentage improvement in cost for Waxman Model is shown in Fig 3.8.

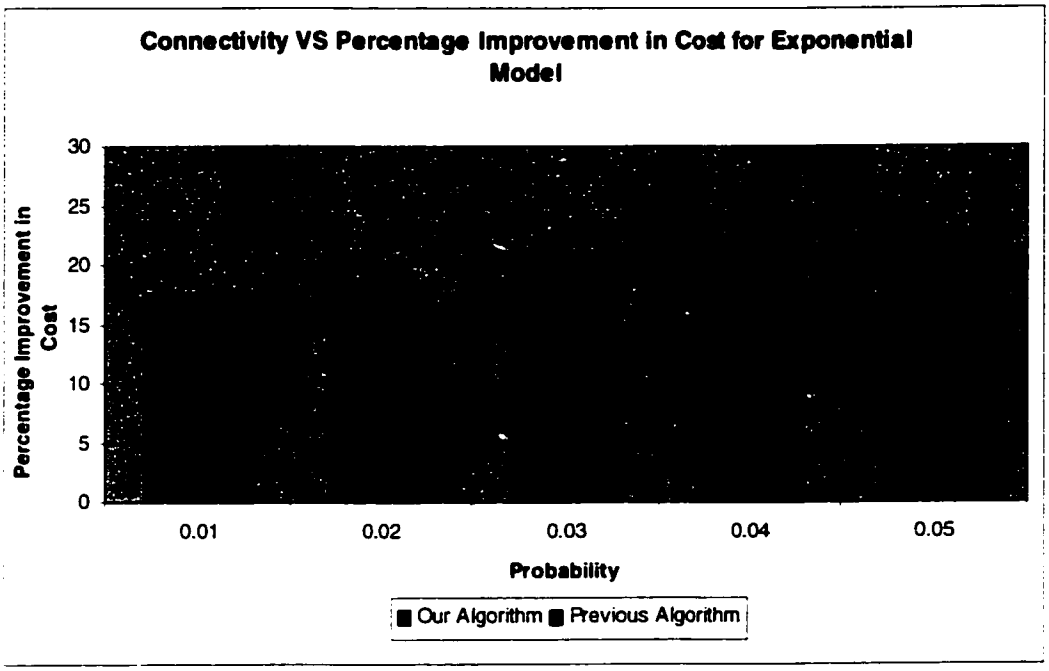


Fig 3.8 the relationship between the probability and percentage improvement in cost for Exponential Model

We can see from Fig 3.8 that each of the algorithms can give a fairly constant percentage improvement in cost for Exponential Model. The average percentage improvement of our algorithm is 20.52% while the average improvement for the referenced algorithm is 19.16%, which can show that our algorithm can provide 7.10% improvement over referenced algorithm.

As we mentioned, our algorithm has two important values, the number of neighbors that the algorithm selects with the lowest cost as the next CurrentNodes and the number of steps that the algorithm executes. Next, we will test these two values respectively. For these simulations, we use the same network environment. The number of nodes is 150, the number of sources is 10, and the thresholds are 0.08 for

Waxman Model and 0.05 for Exponential Model.

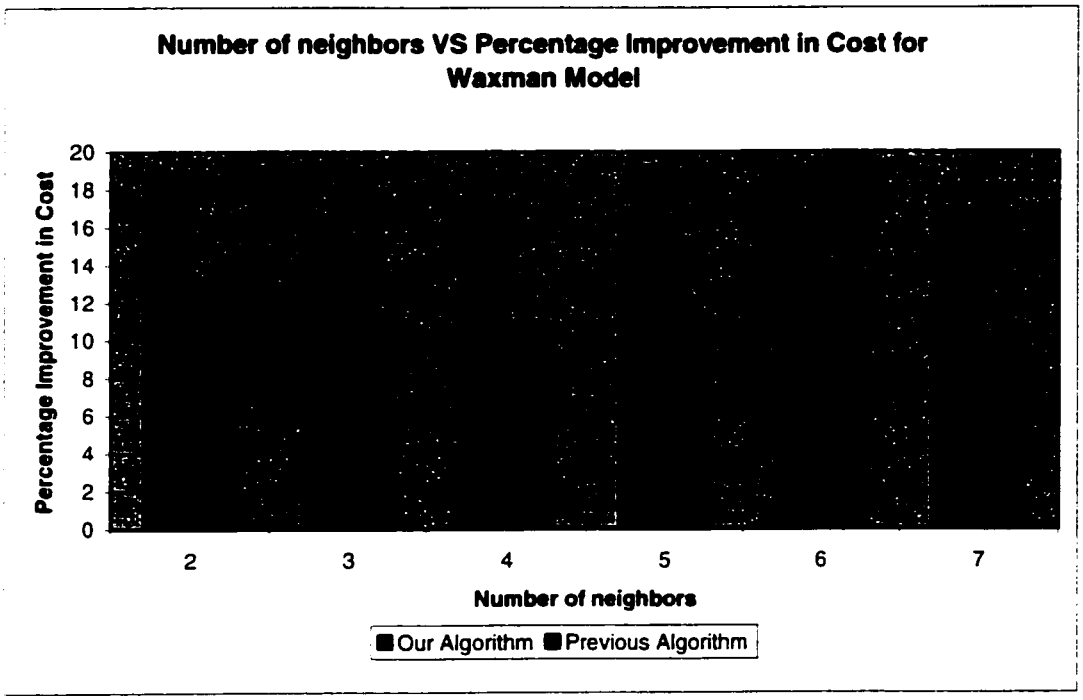


Fig 3.9 Number of neighbors VS percentage improvement in cost for Waxman Model

The relationship between the number of neighbors and the percentage improvement in cost for Waxman Model is shown in Fig 3.9.

We can see in Fig 3.9 that, as the number of neighbors increases, our algorithm can provide fairly constant percentage improvement in cost. The average percentage improvement for our algorithm is 17.16% while the one for the referenced algorithm is 11%. For every one of possible choices on the number of neighbors, our algorithm can provide 56% improvement over the referenced algorithm.

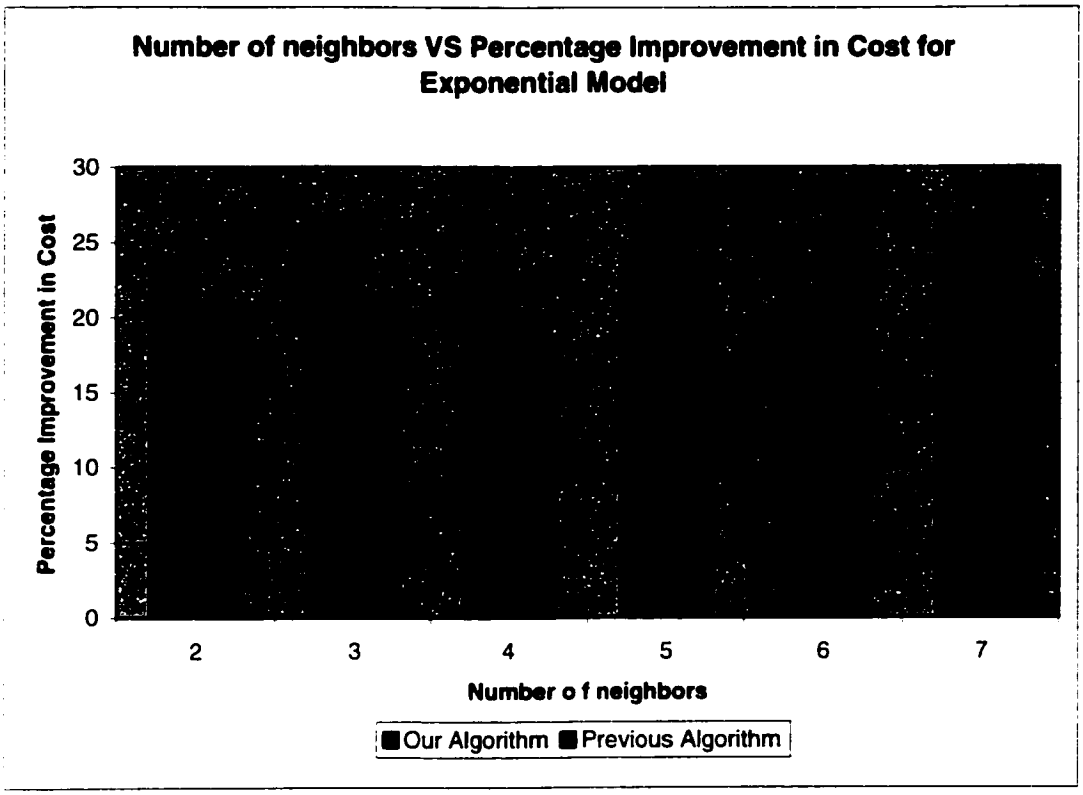


Fig 3.10 Number of neighbors VS Percentage Improvement in Cost for Exponential Model

We run the same situation with Exponential Model and get the simulation result in Fig 3.10.

We can see in Fig 3.10 that, our algorithm can provide fairly constant percentage improvement in cost for Exponential Model, too. The average percentage improvement for our algorithm is 24.31% while the one for the referenced algorithm is 20.01%. For any number of neighbors that we select, our algorithm can provide 21.49% improvement over the referenced algorithm.

From Fig 3.9 and Fig 3.10, as the number of neighbors increases, our algorithm

doesn't provide a significant cost improvement for both network models

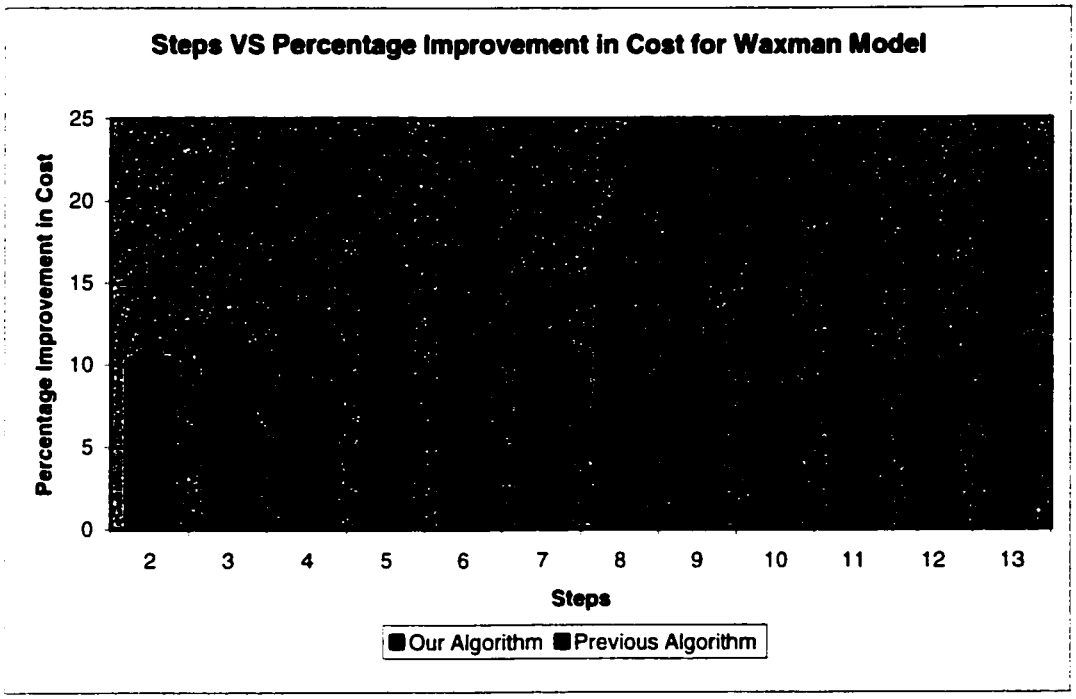


Fig 3.11 Steps VS Percentage Improvement in Cost for Waxman Model

correspondingly. Smaller number of neighbors and larger number of neighbors can almost provide similar cost improvement. Therefore, we can select a small number of neighbors to save a lot of cost spending on those extra nodes. Even though a large number of neighbors may give us more accurate results for some specific cases, "2" is the best number of neighbors when we consider the tradeoff between the accuracy and the cost.

For another parameter of our algorithm, the number of steps our algorithm executes, the relationship between the number of steps and the percentage improvement in cost for Waxman Model is shown in Fig 3.11.

We can see from Fig 3.11 that there are two ranges for the number of steps. When the number of steps is less than 5, our algorithm can provide a fairly constant percentage improvement in cost. The average percentage

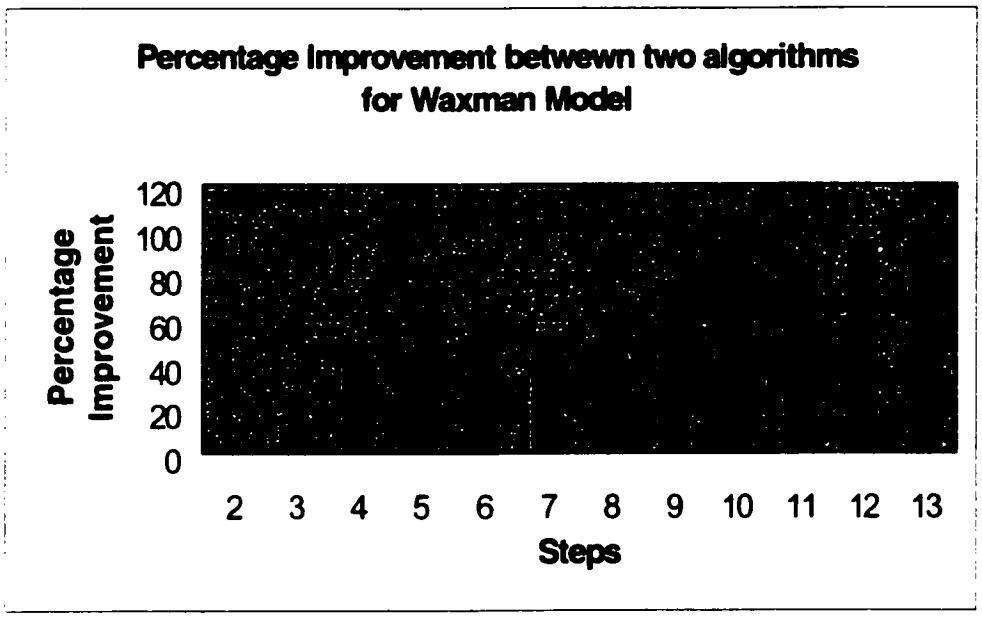


Fig 3.12 Percentage Improvement between two algorithms for Waxman

improvement is 11.78%. As the number of steps is greater than 5, our algorithm can provide another fairly constant percentage improvement. The average percentage improvement for that range is 18.34%, which is almost double than another range. However, comparing with the referenced algorithm, the percentage improvement increases as the number of steps increases. We show the percentage improvement between two algorithms in Fig 3.12. As the number of steps that our algorithm executes increases, the cost improvement gets better compared with the referenced algorithm.

We run the same situation with Exponential Model. Fig 3.13 shows the simulation result.

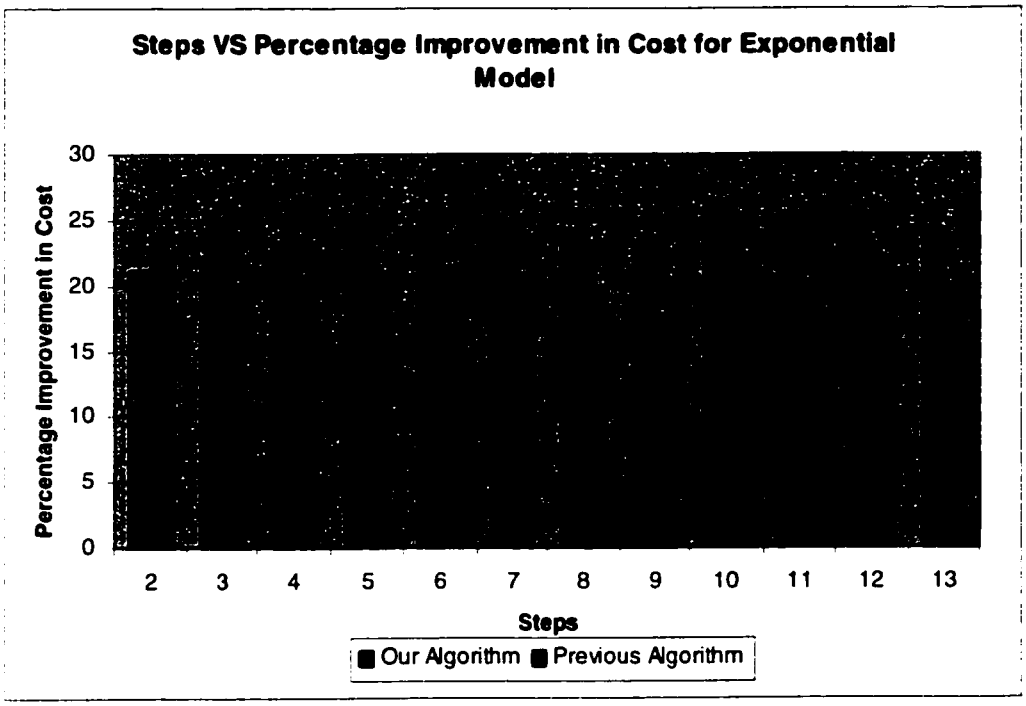


Fig 3.13 Steps VS Percentage Improvement in Cost for Exponential Model

We can see from Fig 3.13 that those two algorithms can provide a fairly constant percentage improvement in cost. The average percentage improvement for our algorithm is 23.48%, while the one for the referenced algorithm is 19.98%. Except the smallest number "2", our algorithm always provides 3.87% improvement over the referenced algorithm.

For both network models, as the number of steps of our algorithm increases, the computation spending for those extra nodes also increases. Again, although larger

number of steps can provide a more accurate result, we recommend "5" as the number of steps that our algorithm execute because it can balance the accuracy and the overhead of the algorithm.

Chapter 4

4 CONCLUSION AND FUTURE DIRECTIONS

4.1 Conclusion

PIM-SM, a kind of Multicast routing protocol, is based on the notion of center-specific trees and distributes data packets from all sources over a single shortest-path tree rooted at some center. Many researchers have worked on the problem of center selection, but they didn't use a rigorous estimated tree cost function to get the real tree cost. Moreover, they did not discuss the tradeoff between the improvement the algorithm can provide and the cost spending on the algorithm.

In this report we have investigated the problem of finding a good center in a centralized fashion, proposed a new algorithm for the real tree cost calculation and a dynamic RP selection algorithm. The complexity of our algorithm is ***O (K*D)***, where k is the number of group members and D is the furthest distance from any group member to current RP. As the group members come and go, our RP selection algorithm can provide a list of potential RPs whose cost improvement compared with the current RP are greater than a predicted threshold. Moreover, we balance the cost improvement and the complexity of the algorithm in order to select a potential RP that is as good as possible while

spending the least cost.

The simulation results for both algorithms show that our algorithm can provide 7.1% -- 56% improvement over the referenced algorithm for both network models with various parameters. As the number of sources and connecting probability of the network environment increases, our algorithm can provide increasing cost improvement and fairly constant percentage improvement in cost. For the two key values in our algorithm, the simulation results show that as the number of neighbors increases, there is no obvious cost improvement. However, as the number of steps our algorithm executes increases, our algorithm can provide a significant percentage improvement in cost. Therefore, in order to balance the cost improvement the algorithm can provide and the cost spending on the algorithm, we recommend "2" and "5" as the number of neighbors and the number of steps, respectively.

4.2 Future Directions

Our RP selection algorithm can select approximately the "best" potential RP among all of the nodes for the current network environment. However, there are still some issues that need to be resolved before we apply it in the PIM-SM protocol. These issues include:

1. The estimation of the threshold for RP selection. After we get the potential RP whose cost is less than the current RP, we need to

determine whether there is a significant improvement if we move the current RP to the potential RP. Our RP selection algorithm needs to spend network resources to do that. If the cost improvement of the new RP is less than the cost that we spend on the algorithm, we don't need to move the current RP to the potential RP. We have to test the RP selection algorithm with a real network environment to select the best threshold for RP relocation. If the cost improvement of the potential RP is greater than the threshold, we will move the current RP to the potential RP; otherwise, we will keep the current RP.

2. The definition of the timer for RP selection. As the group members come and go, the costs of all nodes in the network domain change correspondingly. The problem is how often we execute our RP selection algorithm to select a potential RP. If the frequency is too high, we may waste network resources on those cases where is no need to move the current RP. On the other hand, if the frequency is too low, we may miss the chance to select the best potential RP for recent network environment. So, we have to test our RP selection algorithm in the real network environment to investigate the relationship between the frequency and the cost we spend on our RP selection algorithm in order to set a proper value for the timer in PIM-SM to do the RP relocation.
3. Selecting the best RP in a distributed network environment. The current

version of our algorithm is used in a centralized fashion in which any router in the network domain can get the information about the entire domain. However, in a large-scale network environment, such as the Internet, it is impossible for a router to know the topology of the entire Internet. Therefore, we have to consider the cost calculation function in a distributed fashion. David G. has already make efforts in this direction [10].

4. Selecting the best RP in a hierarchical environment. In the hierarchical environment, the network 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 with members would select its own internal center for a tree extending to all internal members as well as to its inter-domain gateways. Another tree would be similarly constructed for the backbone which would extend to all domains having group members. So, we have to consider the cost calculation function in a hierarchical fashion.

REFERENCE:

[1] J. William Atwood, Ritesh Mukherjee. "RP Relocation in PIM-SM Multicast", draft-atwood-pim-sm-rp-01.txt, November 21, 2001.

[2] A. Ballardie, B. Cain, and Z. Zhang. "Core based trees(CBT version 3) multicast routing." Internet draft, 1998.

[3] S. Deering, D. Estrin, and et. al. "Protocol independent multicast-sparse mode (PIM-SM): Motivation and architecture." Internet draft,1998

[4] D. Estrin, D. Farinacci, V. jacobson, C. Liu, L. Wei, P. Sharma, and A. Helmy. "Protocol independent multicast(PIM) sparse mode/dense mode." Internet draft, 1996.

[5] Bill Fenner/AT&T, et. Al. PIM WG, "Bootstrap Router (BSR) Mechanism for PIM Sparse Mode", INTERNET-DRAFT, November, 2001

[6] H. W. Holbrook and D. R. Cheriton. IP multicast channels: EXPRESS support for large-scale single-source applications. In ACM SIGCOMM'99,1999.

[7] J. Moy. "Multicast extension to OSPF." Internet draft, 1988.

[8] R. Perlman, C.-Y. Lee, A. Ballardie, J. Crowcroft, Z. Wang, and T. Maufer. "Simple multicast: a design for simple, low-overhead multicast." Draft-perlman-simple-multicast-01.txt,1998.

[9] T. Pusateri. "Distance vector routing protocol." Draft-ierf-idmr-dvmrp-v3-07, 1998

[10] David G. Thaler, Chinya V. Ravichankar, "Distributed Center Location Algorithms",

IEEE Journal of Selected Areas in Communications, April 1997.

[11] Liming Wei and Deborah Estrin. " A comparison of multicast trees and algorithms." Technical Report USC-CS-93-560, Computer Science Department, University of Southern California, September,1993.

[12] Ellen W. Zegura, Kenneth L. Calvert, and Samrat Bhattacharjee, "How to Model an Internetwork", Proceedings of IEEEInfocom'96, San Francisco,CA

[13] Bernard M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9): 1617-1622,1988.