CHAPTER 1
INTRODUCTION

The first step in routing is to collect network state information. This information is generally obtained from a link state protocol such as Open Shortest Path First. Link state protocols periodically broadcast a node's state to every other node so that each node knows the network topology and the state of every link. This would have been an acceptable means of broadcasting and maintaining network state information had it not been for the phenomenal growth of networks these days. As the network size grows, it becomes unrealistic to broadcast its entire topology to every node in the network as this will take an enormous amount of space, time and bandwidth. Here are a few proposed solutions to deal with the scalability problem:

(i) Reducing the frequency of topology updates

(ii) Reducing the size of topology updates

(iii) Combining the above two techniques

The goal of frequency reduction is to generate routing update messages as infrequently as possible without compromising routing performance [1]. The goal of size reduction is to reduce the size of these messages while preserving routing performance. This work deals with the topology updates size reduction technique.

1.1 Topology Aggregation

Topology updates size reduction leads to the concept of *Topology Aggregation*. Topology aggregation is perhaps the most important technique to achieve scalability in *Quality-of-Service* (QoS) routing[1] and routing in general, since it can potentially reduce the amount of link state update information by orders of magnitude [3]. It is achieved

[1] Quality of Service routing identifies paths that meet the Quality of Service requirements and selects one that leads to high overall resource efficiency [2].

by grouping neighboring nodes of the network into smaller and manageable routing domains. The process of topology aggregation as discussed in [3] usually consists of four steps:

(i) Grouping or partitioning the network into domains and forming the hierarchical routing

(ii) Deriving the port-to-port distances in each domain

(iii) Representing the port-to-port distances in a compact manner

(iv) Exchanging the aggregated information among domains

Thus, the internal details of a domain are aggregated before they are broadcast to other domains. This means that all the nodes within a domain have a complete view of their domain, but outside nodes only have an aggregated view. External nodes will make use of this aggregated information to make routing decisions, hence the aggregated topology must represent the domain state information as accurately as possible, else routing will suffer greatly. An efficient topology aggregation scheme, thus, is one that provides a proper balance between topology compaction and the impact of this compaction on the routing performance.

1.2 Topology Aggregation Schemes

In the *Asynchronous Transfer Mode* (ATM) Forum, *Private Network-Network Interface* (PNNI) [4] standard proposed a possible topology aggregation protocol. In this protocol, nodes in a network are grouped hierarchically into different peer groups. Although PNNI defines how the aggregated peer group should look like, it does not specify how to do the aggregation [5]. This choice is left for vendor differentiation. There exist several proposed topology aggregation schemes. All the schemes seek to summarize the topology of the routing domains as accurately as possible.

The Full Mesh representation is the basis for most other aggregation schemes. The idea is to first reduce the topology of a routing domain into its full mesh representation, which only consists of the border nodes of the domain. Border nodes are those nodes that are connected via physical links to nodes belonging to external domains. In the full mesh, each pair of border nodes is connected by one or several

logical links[2]. These logical links are then assigned weights. After a domain topology is reduced to its full mesh, it may be further aggregated into more compact forms. This reduction is usually done by pruning links from the full mesh. In Section 2, existing topology aggregation schemes will be surveyed.

1.3 Topology Aggregation for Multiple Additive Metrics

Link metrics may be additive or non-additive. An additive metric associated with every link along a path is added up to determine if a path is acceptable. An example of additive metric is delay. On the other hand, a non-additive metric associated with every link is used to independently determine whether that link can be a part of the path or not. An example of non-additive metric is bandwidth. Effective topology aggregation schemes have been developed for networks with one additive or non-additive metric [6]. However, aggregating topologies with multiple additive metrics still poses significant challenges. We will use an example to show the difference between aggregating topologies with one metric and multiple additive metrics. Consider the topology in Figure 1.1. Let us first assume that each link contains one additive metric as shown in the figure and that the edge nodes are node 1 and node 2. In this case, the best path from node 1 to node 2 can be easily found to be 1-3-2 with metric 20 using any standard shortest path algorithm such as Bellman-Ford or Djikstra's Shortest Path Algorithm.

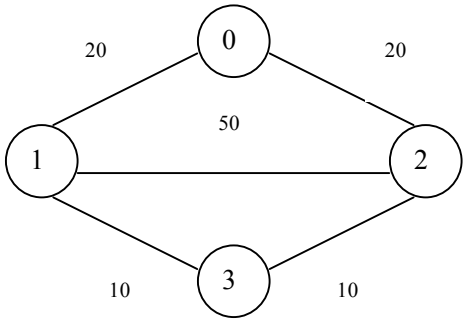

Figure 1.1. Domain topology with links having one additive metric

[2] A logical link between two nodes is a physical link or a physical path between them.

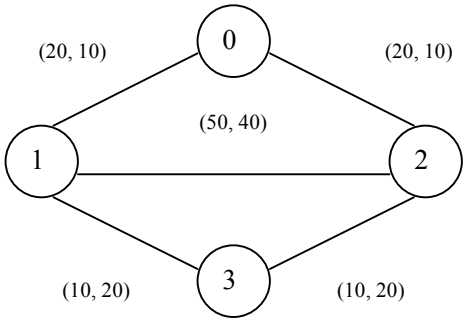

Figure 1.2. Domain topology with links having two additive metrics

Now, consider the case when two additive metrics are involved as shown in figure 1.2. There are two best paths between the nodes 1 and 2. The first path is 1-0-2 with metrics (40, 20) and the second path is 1-3-2 with metrics (20, 40). Both these paths are the best paths with respect to one additive metric. This example shows two significant differences between aggregating topology with one metric and two metrics. First, with two metrics, multiple paths between two nodes might need to be computed. Further, the problem of computing "optimal" paths with multiple additive metrics is NP-hard [7]. Therefore, aggregating topology of a domain when multiple additive metrics is much more difficult than aggregating topologies with one metric. Second, with multiple metrics, it is not clear whether a path is better or worse than another path, this leads to the difficulty in evaluating a topology aggregation scheme.

In this paper, we propose a method to compute the full mesh aggregation with limited path heuristic [2] and demonstrate that the distortion of the heuristic is small. We also propose a novel *area-differences* based scheme to evaluate the performance of an aggregation. Finally we develop a number of spanning tree based aggregations and evaluate them.

Having introduced the concept of topology aggregation and the challenges in aggregation with multiple additive metrics in Section 1, the rest of the paper is organized as follows. Section 2 discusses various existing aggregation schemes and describes the proposed performance evaluation scheme. Section 3 discusses the

computation of the full mesh and the spanning tree based aggregation schemes. The results of performance evaluation are presented in Section 4. Section 5 concludes the paper.

CHAPTER 2
AGGREGATION SCHEMES AND EVALUATION SCHEME

2.1 Existing Topology Aggregation Schemes

As discussed in section 1, in the ATM forum, PNNI standard proposed a possible topology aggregation protocol. However, it does not specify how to do the aggregation. In this section, we discuss the advantages and disadvantages of existing aggregation schemes.

Full Mesh Representation

The Full Mesh representation [8], [9] of a routing domain is constructed by connecting each pair of border nodes by logical links. If each link in the domain has only one associated metric, then the full mesh retains all the distances between the border nodes of the original topology. This is by far the most accurate representation. However it is also the least compact one. A full mesh topology update is a matrix of size $n(n-1)/2$, where n is the number of border nodes. This scheme does not scale well if the size of the network and thereby the size of the domain grows.

Single Node Representation

While the full mesh scheme that advertises too much information lies on one end of the spectrum, the Single Node scheme [8] lies on the other. This approach collapses a routing domain with multiple nodes into a single virtual node. Obviously, it offers the greatest reduction of advertised information as it reduces the routing information size complexity to O(1). It is also very scalable. This scheme hides the border nodes and assigns a matrix of parameters to the virtual node. These parameters are derived from

the topology information of the original domain. A common way to assign the matrix of parameters to the virtual node is to find the best, worst or the average value for every metric associated with all the links in the domain and give the virtual node this value. Another technique is to assign the virtual node metrics of the diameter of the domain. Though this approach reduces the size of the update messages drastically, it may not represent the domain adequately enough to make efficient routing decisions.

Star Representation

The Star representation [8] is a compromise between the two extreme representations: full mesh and single node. In this scheme, border nodes are connected via virtual links to a virtual center node, sometimes called as the *nucleus* [5]. Links going from the border nodes to the nucleus as well as the links going from the nucleus to the border nodes are assigned weights. These weights are obtained from the appropriate links in the full mesh. Lui and Nahrstedt propose a method to assign weights to the links in the star representation in [5]. To make the representation more accurate, *bypasses* may be introduced. Bypasses in the star scheme are links connecting two border nodes and do not pass through the nucleus.

Spanning Tree Representation

A Spanning Tree representation [6] of a domain is a tree whose nodes are the border nodes of a domain and each pair of nodes has only one unique path between them. Therefore a domain with *n* border nodes, when aggregated into a spanning tree will have *n-1* links that need to be advertised. Given the topology of any domain, the first step is to convert it into a full mesh. When each link has only one associated metric, a spanning tree can be created by choosing appropriate links between border nodes. When there are several metrics involved, a method to first convert these metrics into one virtual metric or to choose the most important metric must be applied before the tree is constructed. To make the representation more accurate, links that are crucial, or links that are used frequently may be included in the tree. If each link

has only one associated metric, then the scheme is fairly accurate. It is also much more scalable than the full mesh.

2.2 Proposed Evaluation Scheme

All aggregation schemes may deviate from the original topology by varying degrees. Some amount of distortion of information is invariably introduced. This distortion leads to an increase in the rates of connection rejection for those calls that could have been supported and ultimate failure of connection requests that are accepted. Therefore, it is greatly desirable to reduce this distortion to the maximum possible extent. An effective aggregation scheme, thus, is one in which distortion is almost zero. In other words, an effective aggregation allows for the admittance of calls whose requirements can be supported by the domain and rejects those whose requirements cannot. An aggregation scheme may be evaluated by comparing its performance against the results obtained when no aggregation is done.

For topologies with one additive metric, the distortion of an aggregation scheme can easily be defined as length (shortest path in the aggregation graph) / length (shortest path in the original graph). Unfortunately, this definition cannot be extended to the case with two metrics since the concept of shortest path in networks with multiple metrics is not well defined. Consider that a link in the original network with metrics (5, 5) is approximated by a link with metrics (5, 6) in aggregation scheme X and by a link with metrics (7, 5) in another aggregation scheme Y. In this case how do we decide which aggregated link represents the original link most accurately? In other words, how do we evaluate the performances of the aggregation schemes X and Y and decide which scheme is better than the other?

We propose an *area-differences* based scheme to evaluate the performance of an aggregation. For an aggregation to be distortion free, it must advertise the same information to external domains that would be advertised if no aggregation were done. To evaluate an aggregation using the area-differences based scheme, QoS metrics supported by a set of ‘optimal’ paths between two nodes are represented by areas. We will assume that the two additive QoS metrics are delay and cost. For example,

consider the case when there exist three paths between two nodes A and B with associated metrics (1, 3), (2, 2), and (3, 1). The area covered by the paths between A and B can be found as follows:

(i) Sort the list of paths by the first metric

The above list in the sorted form is (1, 3), (2, 2), (3, 1).

(ii) area = (1 - 0) * 3 + (2 -1) * 2 + (3 – 2) * 1 = 6

In general, for a sorted list, (*x1, y1), (x2, y2), (x3, y3),……., (xn, yn)*

*area = (x1 – 0) * y1 + (x2 – x1) * y2 + (x3 – x2) * y3 +……+ (xn-xn-1) * yn* ------- (1)

Any path with QoS metrics falling in this area is a better path than one or more original paths.

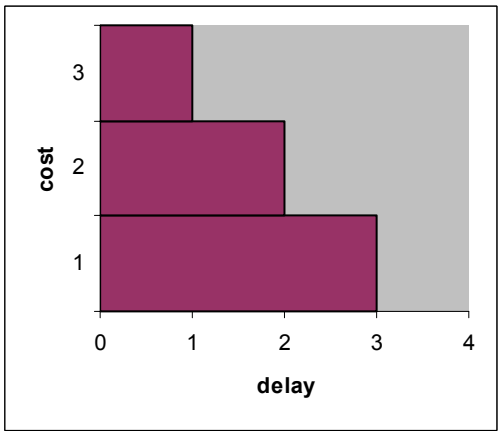

Figure 2.1. Area covered by paths with metrics (1, 2), (2, 2) and (3, 1)

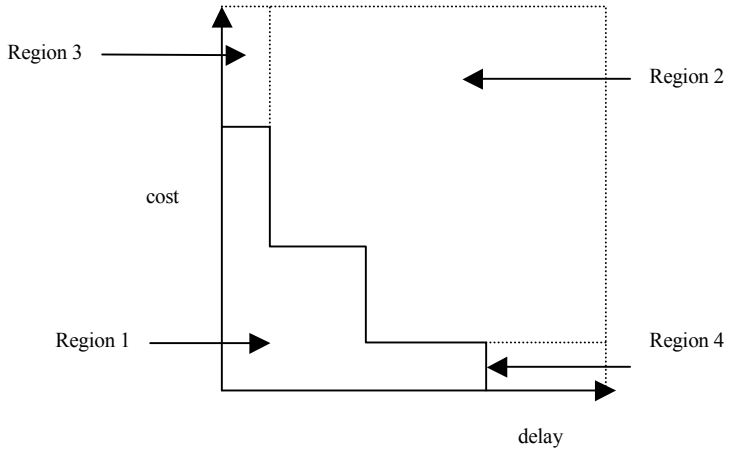

Figure 2.2. Geometric representation of the four area regions

In general, the set of paths between two nodes partitions QoS metrics space into four regions as shown in Figure 2.2.

(i) Requests whose QoS requirement are in region 2 can be satisfied by the set of paths, while requests whose QoS requirement are in regions 1, 3, and 4 cannot be satisfied.

(ii) Paths whose QoS metrics in region 2 are worse than one or more paths in the set of paths. Paths whose QoS metrics in region 1 are better than one or more paths in the set of paths. Paths whose QoS metrics in region 3 and 4 are neither better nor worse than paths in the set.

In the aggregated graph, another set of paths with potentially different QoS metrics are used to approximate the original set of paths. The quality of the approximation scheme can be determined by measuring how closely the approximated contour matches the original contour. Figure 2.3 depicts this case. Let the solid line denote the QoS performance in the original network and the dotted line denote the approximation. The difference between these two contours can be measured by the differences in the areas as shown in the figure. There can be two kinds of differences as indicated in Figure 2.3 by region 1 and region 2. Requests whose QoS requirements fall in region 1 can be satisfied by the original graph but cannot be satisfied in the aggregated graph. Requests whose QoS requirements fall in region 2 cannot be satisfied in the original graph, but can be satisfied in the aggregated graph (hence, the aggregated graph mis-predicts). Since in both cases, the aggregated graph does not reflect the original graph, we will call both region 1 and region 2 in Figure 2.3 mis-predict regions. Intuitively, a better aggregation scheme should have a smaller mis-predict region. Based on this intuition, we propose to use the following formula to define the distortion of a topology aggregation scheme for networks with two additive metrics:

Distortion = sum of the size of all mis-predict regions / size of original contour(size of region 1)

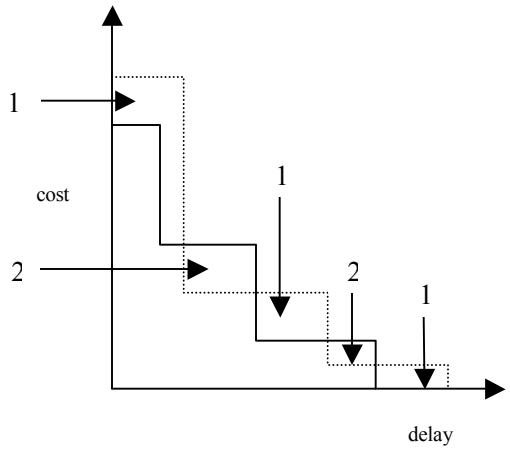

Figure 2.3. Diagrammatic representation of area differences

CHAPTER 3

TOPOLOGY AGGREGATION FOR NETWORKS WITH TWO ADDITIVE METRICS

The network is modeled as a directed graph G(N, E), where N is the set of nodes representing routers and E is the set of edges representing links that connect the routers. Each edge $e = u \rightarrow v$ is associated with two independent weights $w_1(e)$ and $w_2(e)$. The notation $w(e) = w(u \rightarrow v) = (w_1(e), w_2(e))$ is used to represent the weights of a link. It is assumed that the weights associated with a link are additive. Thus for a path $p = v_o \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_n$, where $v_i \in E$, $w_1(p) = \sum_{i = 1,..,n} w_1 (v_{i-1} \rightarrow v_i)$ and $w_2(p) = \sum_{i = 1,..,n} w_2 (v_{i-1} \rightarrow v_i)$. We will use delay and cost as the two additive metrics in all examples in this section.

3.1 Computing Full Mesh Summary

The first step in topology aggregation is to compute the full mesh summary that gives QoS metrics that can be supported between each pair of border nodes. As described in Section 1, a full mesh representation is constructed by connecting each pair of border nodes of the domain by one or several logical links. Each link in the network is associated with two additive metrics and assumed to be symmetric, that is, a link has the same properties in both directions. As mentioned earlier, computing paths between two nodes with two additive metrics is an NP-hard problem. Fortunately, with the recent advances on multi-constrained QoS routing, effective heuristics for this problem have been developed. We propose to use the limited path heuristic [2] to compute the paths between all edge routers so that the full mesh topology can be obtained. The heuristic is a minor modification of the version in [2] in that the QoS requirement is no longer a factor in the selection of paths and thus, all optimal paths can potentially be stored.

The limited path heuristic uses an extended Bellman-Ford algorithm, which differs from the original Bellman-Ford algorithm as described in [11] in that it has each node *u* maintain a set *PATH (u*) that records all optimal paths from the source node to *u*. An optimal path between two nodes satisfies particular QoS constraints that no other path can. This algorithm can find a path that satisfies all the QoS constraints when such a path exists by recording all optimal paths in each node [2]. Since the number of optimal paths from the source node to each node *u* may grow exponentially with the size of the network, a check is placed in the algorithm (line (9)) so that a new path is added to the set only if the size of the set is less than NPATH. As a result, it is possible that not all optimal paths between the source and destination nodes are found, and thus, only an approximate solution is found. Thus, the value for NPATH must be selected carefully so that the routing performance can be maintained. As will be demonstrated in the performance evaluation section, the limited path heuristic performs fairly well in practice and the full mesh representation produced with the heuristic gives a fairly accurate approximation of the original network. The algorithm is summarized as follows:

```
RELAX(u, v, w)
(1) For each w(p) in PATH(u)
(2)     flag = 1
(3)     For each w(q) in PATH(v)
(4)         if(w(p) + w(u, v) ≥ w(q)) then
(5)             flag = 0
(6)         if(w(p) + w(u, v) < w(q)) then
(7)             remove w(q) from PATH(v)
(8)     if(flag = 1) then
(9)         if (size(PATH(v)) < NPATH) add w(p) + w(u, v) to PATH(v)

Limited-path-heuristic(G, w, src, dst)
(1) For i = 0 to |N(G)| - 1
(2)     PATH(i) = Φ
```

(3) PATH(*src*) = {0}

(4) For i = 1 to |N(G)| -1

(5) For each edge (u, v) Є E(G)

(6) RELAX(u, v, w)

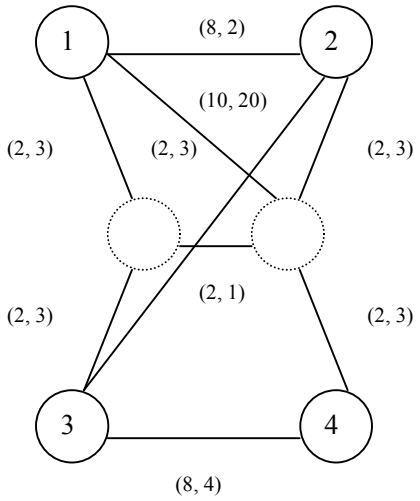

Figure 3.1. Domain topology

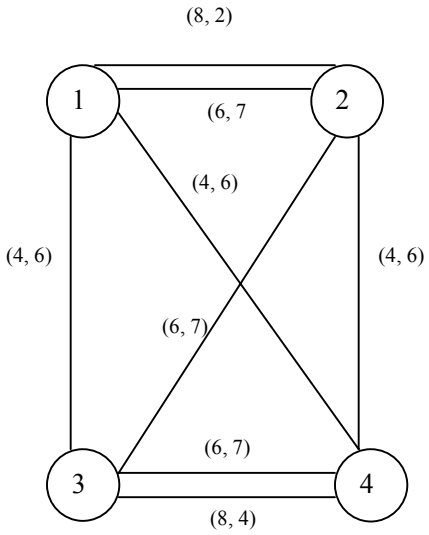

Figure 3.2. Full mesh of the domain

In the figure 3.1, the entire topology of the domain is seen. We have border nodes (1, 2, 3 and 4) as well as internal nodes (nodes shown in dotted lines). Links between the nodes have two associated metrics indicated as (*delay, cost*). In the full mesh (as shown in figure 2.2) paths between each pair of border nodes are shown. Some pairs can have more that one optimal path between them. In the figure 3.2, border nodes 1 and 2 have two paths between them. The first path with metrics (8, 2) is better in terms of cost as compared to the second path which has associated metrics (6, 7). On the other hand, path (6, 7) is better that path (8, 2) in terms of delay. For these two paths between nodes 1 and 2 to have been found, the value of NPATH had to be at the very least 2.

3.2 Computing Spanning Tree Aggregation

A full mesh aggregation of a domain constructed using the method described above would have at least *n(n-1)/2* links that need to be advertised, where *n* is the number of border nodes. This is still a large amount of data. The mesh is aggregated further so that fewer links would need to be advertised, thereby reducing the amount of space, time and bandwidth that each topology update would require. We compute a spanning tree from the full mesh aggregation. Since all spanning tree construction algorithms work for networks with a single metric, we need to first convert the weights on logical links into a single weight. Notice that the goal is to produce a spanning tree with the minimum distortion, which is based on area and is defined in Section 2. There are two issues in this process: (1) converting the weights on a logical link into a single value, and (2) computing the spanning tree with minimal distortion. We follow the methods in [6] to compute spanning trees. Specifically, we consider *Minimum Spanning Trees* (MST), *Random Spanning Trees* (RST), and a combination of these two kinds of trees. To generate a MST, a list of all the links from the full mesh is created and is sorted in the ascending order by the weight. Links are added one by one to the tree such that the properties of a spanning tree do not get violated. In order to construct a RST, a sorted list is not required. Next, we will describe various schemes to convert weights into a single value.

Area-Based Spanning Trees

The first set of trees is *area-based.* Each logical link is represented by a weight equal to the area covered by the physical paths found between the pair of nodes connected by that link. The reason for using area as the metric is that we are trying to produce a spanning tree where the distance between two nodes has the smallest area. Notice that area-based trees may not achieve that since area is neither additive nor concave. Area-based full mesh representation of figure 3.2 obtained by using equation (1) is shown in figure 3.3.

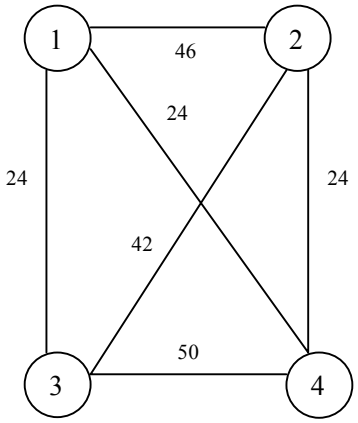

Figure 3.3. Full mesh with weights represented by *area*

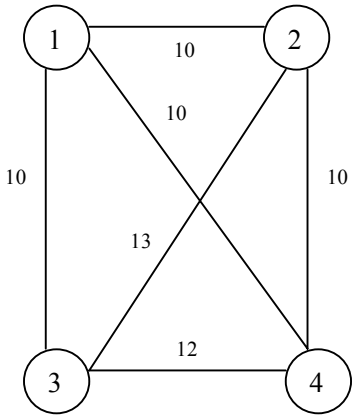

Figure 3.4. Full mesh with weights represented by *min*(*a1* * delay + *a2* * cost)

Sum-Based Spanning Trees

Sum-based spanning tree is constructed by conversion of two weights into one weighted sum of the two weights. Traditionally, this has been a typical method to convert multiple metrics into one so that algorithms for one metric can be used to solve routing problems with multiple metrics. The weight of a link is derived as:

weight of link = *function*(*a1* * delay + *a2* * cost)

Here *a1* and *a2,* which gives weights to the two metrics, are parameters that are inputted by the user. This approach allows priorities to be given to different metrics in the optimization. In the examples below, we will assume *a1* = 1 and *a2* = 1. There are three different ways to assign weights based on the function used.

Case 1: weight of link = *min*(*a1* * delay + *a2* * cost)

Referring to figure 3.2, the weight of the logical link between nodes 1 and 2 can be computed as:

weight of link = *min*((1 * 8 + 1 * 2), (1 * 6 + 1 * 7))

= 10

The new full mesh representation is shown in figure 3.4.

Case 2: weight of link = *max*(*a1* * delay + *a2* * cost)

The weight of the logical link between nodes 1 and 2 can be computed as:

weight of link = *max*((1 * 8 + 1 * 2), (1 * 6 + 1 * 7))

= 13

The full mesh with weights computed using the *max* function is shown in figure 3.5.

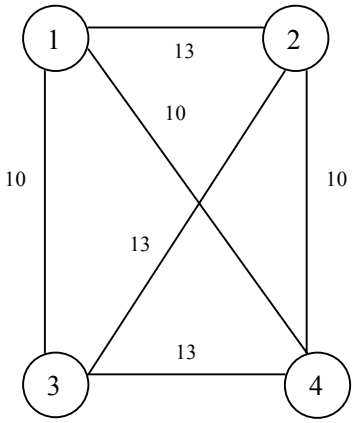

Figure 3.5. Full mesh with weights represented by *max*(*a1* * delay + *a2* * cost)

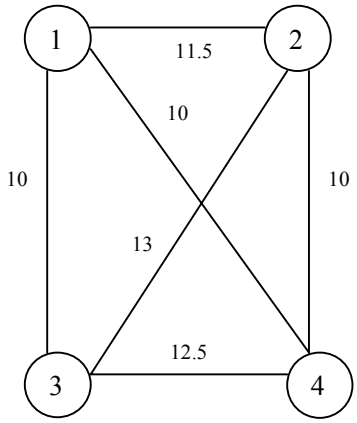

Figure 3.6. Full mesh with weights represented by *min*+*max*](*a1* * delay + *a2* * cost)/2

Case 3: weight of link = [*min*+*max*](*a1* * delay + *a2* * cost)/2

The weight of the logical link between nodes 1 and 2 can be computed as:

weight of link = [*min*+*max*]((1 * 8 + 1 * 2), (1 * 6 + 1 * 7))/2

Here *min* = 10 and *max* = 13, hence weight of link = (10+13)/2 = 11.5

The full mesh with weights using the *(min+max)/2* function is shown in figure 3.6.

Non-Linear Path Length-Based Spanning Trees

This set of trees is generated by computing the weights for the logical links using the formula:

weight of link = *function*(*min*(delay/*a1*, cost/*a2*))

Here again we have three cases based on the function used. a*1* and *a2* are user inputs. This approach is motivated by the concept of non-linear path lengths [10] that are commonly used in QoS routing. As defined in [10], given the QoS constraint (a1, a2), the non-linear length of a link is defined as min(delay/a1, cost/a2). This concept is used to convert multiple QoS metrics into one so that traditional algorithms that work on a single metric can be applied to solve multi-constrained QoS routing problems.

Case 1: weight of link = *min*(*min*(delay/*a1*, cost/*a2*))

The weight of the link between nodes 1 and 2 assuming *a1* = 100 and *a2* = 100, can be computed as:

weight of link = *min*(*min*(8/100, 2/100), *min*(6/100, 7/100))

= *min*(0.02,0.06) = 0.02

The full mesh with weights computed using this function is shown in figure 3.7.

Case 2: weight of link = *max*(*min*(delay/*a1*, cost/*a2*))

The weight of the link between nodes 1 and 2 can be computed as:

weight of link = *max*(*min*(8/100, 2/100), *min*(6/100, 7/100))

= *max*(0.02,0.06)

= 0.06

The full mesh with weights computed using this function is shown in figure 3.8.

Case 3: weight of link = [*min*+*max*](*min*(delay/*a1*, cost/*a2*))/2

The weight of the link between nodes 1 and 2 can be computed as:

weight of link = [*min*+*max*](*min*(8/100, 2/100), *min*(6/100, 7/100))/2

= [*min*+*max*](0.02,0.06)/2

= 0.04

The full mesh with weights computed using this function is shown in figure 3.9.

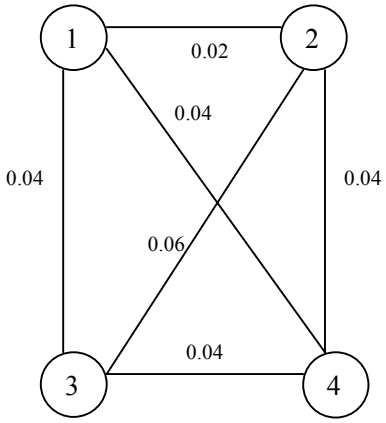

Figure 3.7. Full wesh with weights represented by *min*(*min*(delay/*a1*, cost/*a2*))

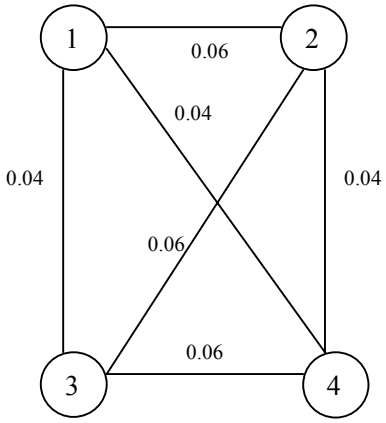

Figure 3.8. Full mesh with weights represented by *max*(*min*(delay/*a1*, cost/*a2*))

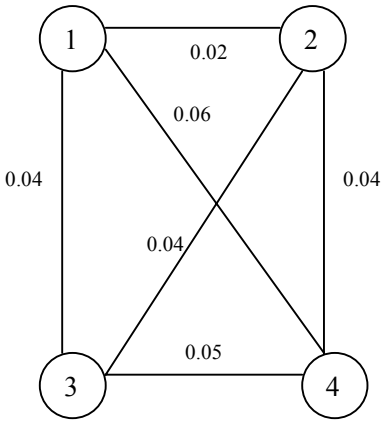

Figure 3.9. Full mesh with weights represented by [*min*+*max*](*min*(delay/*a1*, cost/*a2*))/2

CHAPTER 4
PERFORMANCE EVALUATION

In this section, we present the results obtained by performing simulations on topologies with 100, 200, 300 and 400 nodes in one domain. These topologies were generated by using the Georgia Tech-Internetwork Topology Models (GT-ITM) graph generation package [12]. This package supports the generation of structured network topologies that comprise of multiple administrative domains. In the simulation, the weights for each link, delay and cost, are randomly generated in the range [1-100].

4.1 Evaluation of Full Mesh Summary

As discussed in Section 1, a full mesh summary of a domain is the most accurate representation. However, since we use a heuristic to construct the full mesh representation, we must make sure that the full mesh representation accurately represents the original network. This section reports the results of the evaluation of the full mesh representation.

We performed experiments on domains of various sizes with different limited path heuristic to find the degree of distortion present in the full mesh summary generated. The degree of distortion in an aggregation can be found by comparing the area covered between any two border nodes with limited path heuristic set to *x* against the area covered by them when an exhaustive search is done, that is, limited path heuristic is set to ∞.

The domains under consideration were of sizes 100 nodes with 20 border nodes, 200 nodes with 40 border nodes, 300 nodes with 60 border nodes and 400 nodes with 80 border nodes.

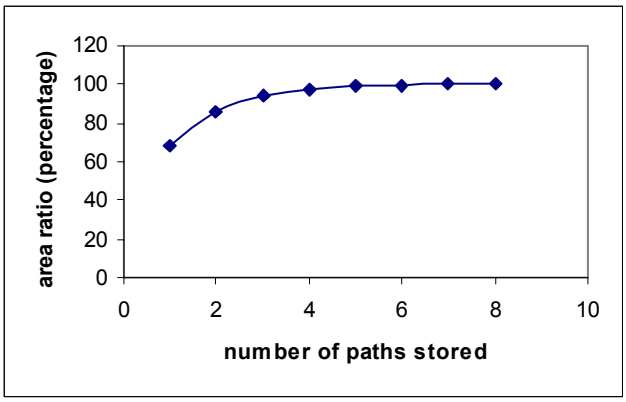

Figure 4.1. Limited path heuristic study for domain with 100 nodes and 20 border nodes

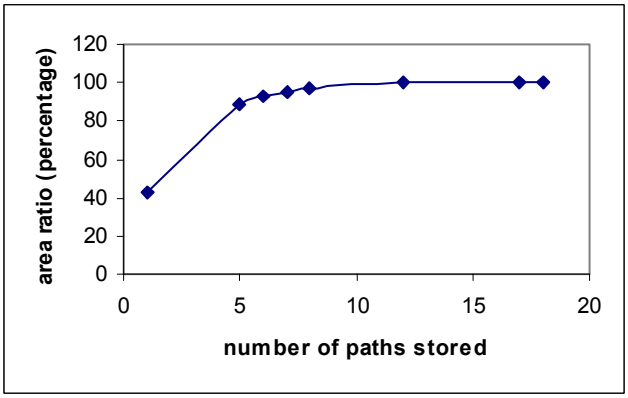

Figure 4.2. Limited path heuristic study for domain with 200 nodes and 40 border nodes

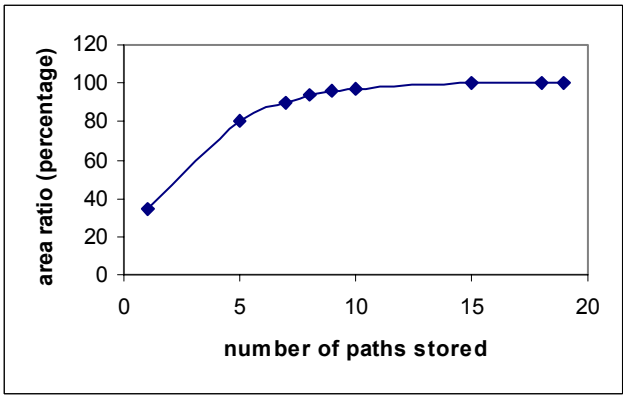

Figure 4.3. Limited path heuristic study for domain with 300 nodes and 60 border nodes

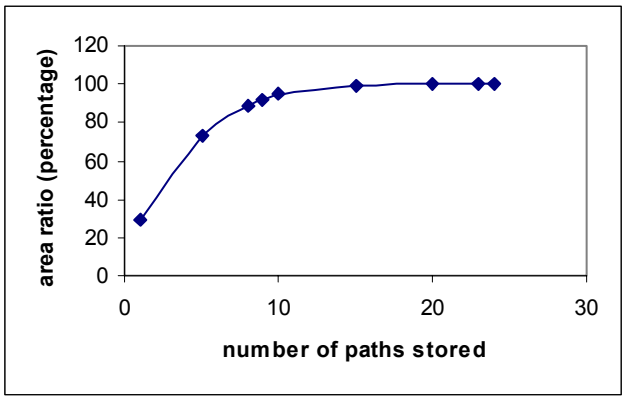

Figure 4.4. Limited path heuristic study for domain with 400 nodes and 80 border nodes

Table 4.1 – Performance Evaluation of Full Mesh Summary

Number of Nodes	Number of Border Nodes	Limited Path Heuristic		
		Distortion < 10%	Distortion < 5%	Distortion = 0
100	20	3	4	8
200	40	6	7	18
300	60	7	9	19
400	80	9	10	24

The information represented by the graphs in figures 4.1, 4.2, 4.3 and 4.4 is summarized in table 4.1. It is observed from this study that 8 paths stored between a pair of border nodes for a 100-node domain with 20 border nodes, 18 paths for a 200-node domain with 40 border nodes, 19 paths for a 300-node domain with 60 border nodes and 24 paths for a 400-node domain with 80 border nodes will yield a full mesh summary that has zero distortion. As low as 3, 6, 7 and 9 paths respectively can be stored for faster performance of the EBFA in networks where a distortion of less than 10% is acceptable. Figure 4.5 shows the distortion histogram for the 100-node domain

when the limited path heuristic was set to 3. It is obvious from the histogram that the number of links with zero distortion is very high.

In summary, these results indicate that the limited path heuristic indeed results in accurate full mesh representation by maintaining a relative small number of paths (e.g. 10) in each node even for a relative large network. In the next section, we will evaluate the schemes to further reduce the size of the summary.

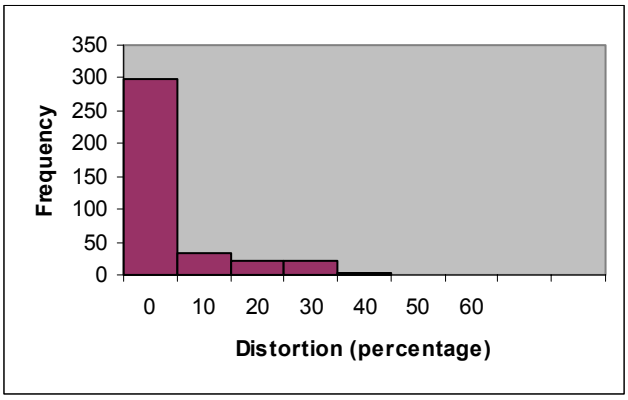

Figure 4.5. Distortion histogram for a 100-node domain with 20 border nodes and limited path heuristic = 3

4.2 Evaluation of Spanning Tree Summary

To evaluate the performance of the spanning tree aggregation, a number of MSTs, RSTs, and a combination of MSTs and RSTs were generated. The goal of the experiments is to compare the performance of the trees and decide which one maps the original topology of the domain with the least amount of distortion. The first simulation was done on a 100-node domain with 20 border nodes. We did a comparative study between the performance of one MST and one RST. Each MST and RST was constructed from the full mesh of the domain with the weights assigned according to the different methods described in Section 3.The values for *a1* and *a2* for the construction of sum-based trees were set to 1 and their values for the construction

of the non linear path-length based trees were set to 100. Table 4.2 shows the results obtained.

Table 4.2. Performance Comparison of MST vs. RST for a 100-node Domain

100 Nodes with 20 Border Nodes		Average Difference Ratio	Standard Deviation
1 MST	Area Based	1.15	1.83
	Sum Based Case 1	1.89	3.77
	Sum Based Case 2	1.15	1.83
	Sum Based Case 3	1.29	1.92
	Path Length Based Case 1	1.99	2.94
	Path Length Based Case 2	1.39	1.92
	Path Length Based Case 3	1.58	2.39
1 RST	Area Based	64.48	270.90
	Sum Based Case 1	147.85	785.53
	Sum Based Case 2	131.38	804.32
	Sum Based Case 3	88.55	333.66
	Path Length Based Case 1	157.70	754.97
	Path Length Based Case 2	87.94	471.60
	Path Length Based Case 3	112.11	485.59

Two concepts *Average Difference Ratio* and *Standard Deviation* are used to compare the performances. Difference Ratio is obtained by finding the difference in the area covered by a logical link in the full mesh and the area covered by it in the spanning tree scheme and dividing this difference by the area represented by the link

in the full mesh. We use the average difference ratio to describe the amount of distortion present in an aggregation. Standard deviation has its universal meaning.

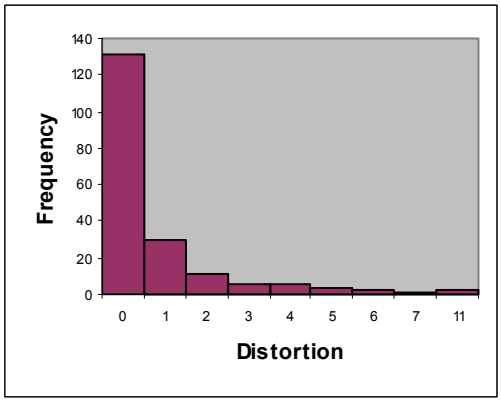

Figure 4.6. Distortion Histogram for area based MST

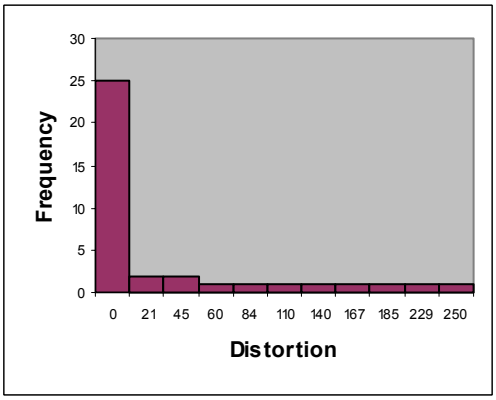

Figure 4.7. Distortion Histogram for area based RST

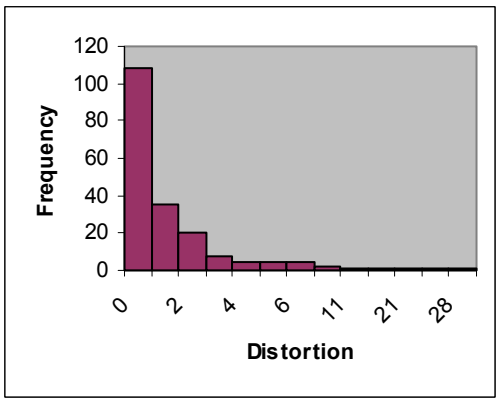

Figure 4.8. Distortion Histogram sum-based MST with function min(a1*delay + a2*cost)

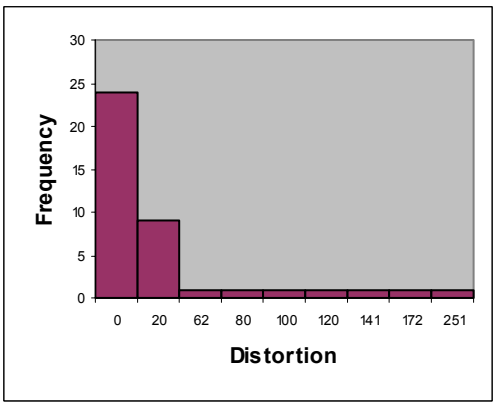

Figure 4.9. Distortion Histogram sum-based RST with function min(a1*delay + a2*cost)

Figure 4.10. Distortion Histogram sum-based MST with function max(a1*delay + a2*cost)

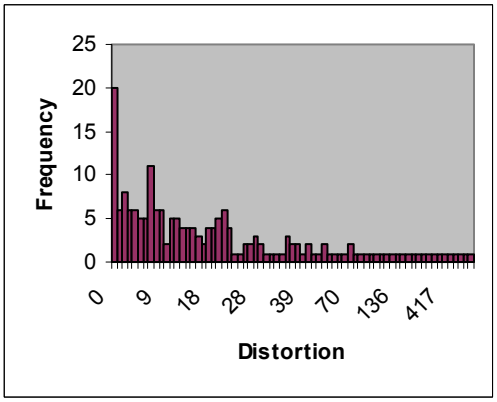

Figure 4.11. Distortion Histogram sum-based RST with function max(a1*delay + a2*cost)

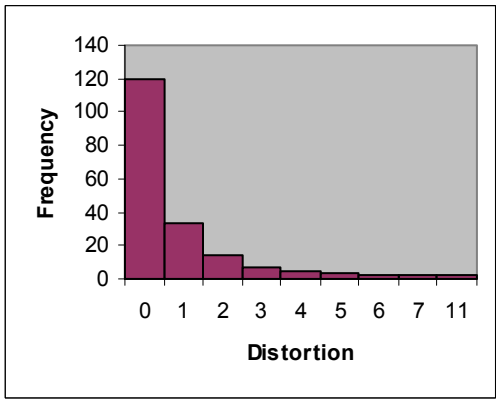

Figure 4.12. Distortion Histogram sum-based MST with function [min+max](a1*delay + a2*cost)/2

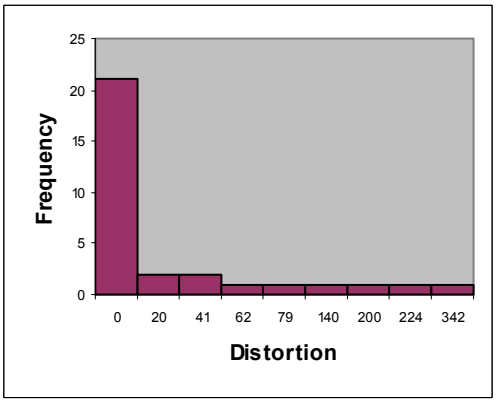

Figure 4.13. Distortion Histogram sum-based RST with function [min+max](a1*delay + a2*cost)/2

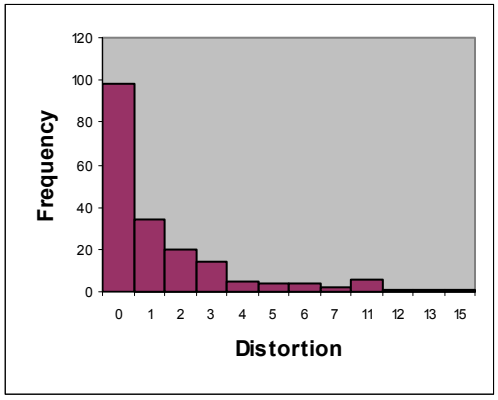

Figure 4.14. Distortion Histogram path-based MST with function min(min(delay/a1,cost/a2))

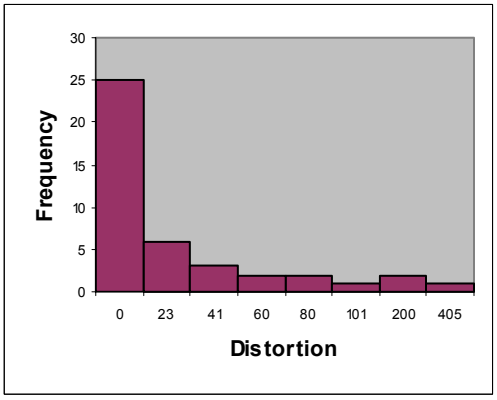

Figure 4.15. Distortion Histogram path-based RST with function min(min(delay/a1,cost/a2))

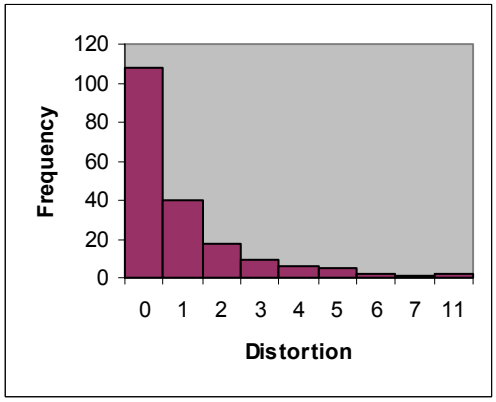

Figure 4.16. Distortion Histogram path-based MST with function max(min(delay/a1,cost/a2))

Figure 4.17. Distortion Histogram path-based RST with function max(min(delay/a1,cost/a2))

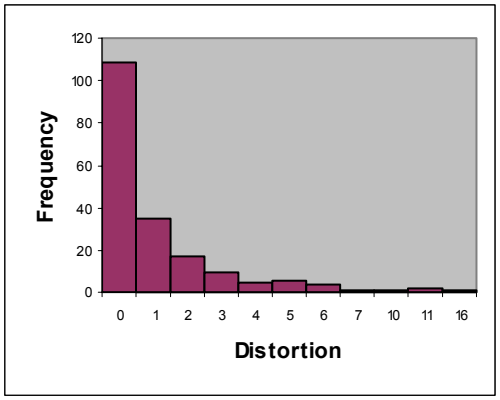

Figure 4.18. Distortion Histogram path-based MST with function [min+max](min(delay/a1,cost/a2))/2

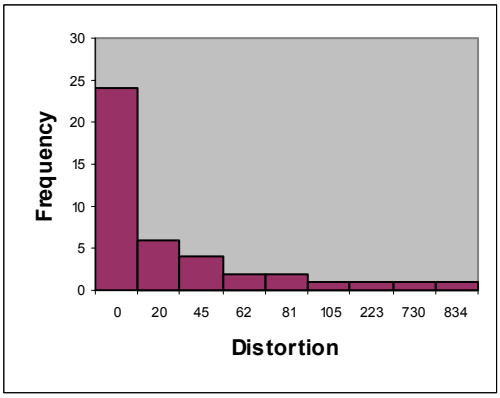

Figure 4.19. Distortion Histogram path-based RST with function [min+ max](min(delay/a1,cost/a2))/2

From table 4.2, it is clear that a MST summary of a domain performs several times better than a RST summary of that domain. Furthermore, it is shown that an area-based MST and a sum-based MST with weights assigned using the function *max*(*a1* * delay + *a2* * cost). A sum-based MST with weights assigned using the function [*min*+*max*](*a1* * delay + *a2* * cost)/2 also perform relatively good. Distortion histograms for the MST and RST computed for the 100-node domain with 20 border nodes are presented in figures 4.6 – 4.19.To further validate our results, we performed similar experiments on the 200, 300 and 400 node domains. Table 4.3, 4.4 and 4.5 summarize the results.

Table 4.3. Performance Comparison of MST vs. RST for a 200-node Domain

200 Nodes with 40 Border Nodes		Average Difference Ratio	Standard Deviation
1 MST	Area Based	4.01	5.29
	Sum Based Case 1	7.65	12.03
	Sum Based Case 2	3.54	4.30
	Sum Based Case 3	3.44	4.37
	Path Length Based Case 1	8.54	16.58
	Path Length Based Case 2	4.68	7.74
	Path Length Based Case 3	8.00	12.70
1 RST	Area Based	219.94	3857.12
	Sum Based Case 1	127.70	2251.93
	Sum Based Case 2	98.70	830.48
	Sum Based Case 3	81.14	658.54
	Path Length Based Case 1	133.27	2092.14
	Path Length Based Case 2	105.98	1431.84
	Path Length Based Case 3	99.33	987.17

Again, it is clear that a MST out-performs a RST and an area-based MST, a sum-based MST with weights assigned using the function *max*(*a1* * delay + *a2* * cost) and a sum-based MST with weights assigned using the function [*min*+*max*](*a1* * delay + *a2* * cost)/2 introduce the lowest amount of distortion in the routing information as compared to the other MSTs.

Table 4.4. Performance Comparison of MST vs. RST for a 300-node Domain

300 Nodes with 60 Border Nodes		Average Difference Ratio	Standard Deviation
1 MST	Area Based	3.45	3.96
	Sum Based Case 1	5.66	7.18
	Sum Based Case 2	3.30	3.94
	Sum Based Case 3	3.55	4.03
	Path Length Based Case 1	17.32	45.48
	Path Length Based Case 2	5.36	7.10
	Path Length Based Case 3	8.10	12.02
1 RST	Area Based	102.12	1222.37
	Sum Based Case 1	124.38	1761.48
	Sum Based Case 2	89.62	515.61
	Sum Based Case 3	123.54	1337.44
	Path Length Based Case 1	174.27	1587.37
	Path Length Based Case 2	198.92	3368.50
	Path Length Based Case 3	170.50	1507.01

Table 4.5. Performance Comparison of MST vs. RST for a 400-node Domain

400 Nodes with 80 Border Nodes		Average Difference Ratio	Standard Deviation
1 MST	Area Based	4.37	5.68
	Sum Based Case 1	8.47	10.85
	Sum Based Case 2	3.90	5.12
	Sum Based Case 3	5.08	7.05
	Path Length Based Case 1	42.57	125.58
	Path Length Based Case 2	8.92	11.15

Table 4.5-continued

400 Nodes with 80 Border Nodes		Average Difference Ratio	Standard Deviation
1 MST	Path Length Based Case 3	11.55	16.32
1 RST	Area Based	102.11	1401.25
	Sum Based Case 1	113.03	795.13
	Sum Based Case 2	147.09	1877.76
	Sum Based Case 3	157.42	3743.76
	Path Length Based Case 1	124.30	933.83
	Path Length Based Case 2	98.13	1204.14
	Path Length Based Case 3	138.85	2053.48

The next set of experiments was done to evaluate the performances of three MSTs, three RSTs and a combination of one MST and two RSTs. The results obtained for the 100-node, 200-node, 300-node, 400-node domains are summarized in table 4.6-4.9. From the results is shown that three overlapping MSTs perform much better that three RSTs as well as the combination of one MST and 2 RSTs.

Between the three RSTs and the combination of one MST and two RSTs, the combination performs much better. It can now be stated that a Random Tree aggregation for a domain introduces the greatest degree of distortion and it is not advisable to use this representation else routing will suffer greatly. Furthermore it becomes obvious that the best way to assign the weights to the logical links is to use the area representation or the sum-based representation with case 2 or case 3 functions.

Table 4.6. Performance Comparison of 3 MSTs vs. 3 RSTs vs. 1 MST + 2 RSTs for a 100-node Domain

100 Nodes with 20 Border Nodes		Average Difference Ratio	Standard Deviation
3 MST	Area Based	0.36	0.61
	Sum Based Case 1	0.48	0.93
	Sum Based Case 2	0.28	0.43
	Sum Based Case 3	0.33	0.56
	Path Length Based Case 1	0.99	1.70
	Path Length Based Case 2	0.32	0.54
	Path Length Based Case 3	0.48	1.05
3 RST	Area Based	7.48	52.9
	Sum Based Case 1	9.31	50.15
	Sum Based Case 2	12.41	67.81
	Sum Based Case 3	17.38	128.20
	Path Length Based Case 1	8.55	58.05
	Path Length Based Case 2	11.87	58.53
	Path Length Based Case 3	13.25	84.65
1 MST + 2 RST	Area Based	1.21	2.01
	Sum Based Case 1	1.15	2.87
	Sum Based Case 2	0.67	1.12
	Sum Based Case 3	0.97	1.65
	Path Length Based Case 1	0.89	2.20
	Path Length Based Case 2	0.76	1.23
	Path Length Based Case 3	0.97	1.81

Table 4.7. Performance Comparison of 3 MSTs vs. 3 RSTs vs. 1 MST + 2 RSTs for a 200-node Domain

200 Nodes with 40 Border Nodes		Average Difference Ratio	Standard Deviation
3 MST	Area Based	0.94	1.21
	Sum Based Case 1	0.85	1.18
	Sum Based Case 2	0.99	1.17
	Sum Based Case 3	0.80	0.96
	Path Length Based Case 1	2.33	7.56
	Path Length Based Case 2	0.83	0.99
	Path Length Based Case 3	0.93	1.39
3 RST	Area Based	17.09	254.57
	Sum Based Case 1	18.11	202.98
	Sum Based Case 2	9.48	43.83
	Sum Based Case 3	9.60	56.41
	Path Length Based Case 1	9.17	47.67
	Path Length Based Case 2	19.01	274.92
	Path Length Based Case 3	18.17	34.23
1 MST + 2 RST	Area Based	2.27	3.38
	Sum Based Case 1	1.95	3.55
	Sum Based Case 2	2.04	2.96
	Sum Based Case 3	2.12	3.29
	Path Length Based Case 1	2.77	6.35
	Path Length Based Case 2	1.98	3.54
	Path Length Based Case 3	2.34	1.91

Table 4.8. Performance Comparison of 3 MSTs vs. 3 RSTs vs. 1 MST + 2 RSTs for a 300-node Domain

300 Nodes with 60 Border Nodes		Average Difference Ratio	Standard Deviation
3 MST	Area Based	1.35	1.47
	Sum Based Case 1	1.41	1.63
	Sum Based Case 2	1.42	1.52
	Sum Based Case 3	1.30	1.40
	Path Length Based Case 1	2.48	6.82
	Path Length Based Case 2	1.70	1.89
	Path Length Based Case 3	1.56	2.00
3 RST	Area Based	18.23	239.80
	Sum Based Case 1	15.25	195.60
	Sum Based Case 2	21.48	281.56
	Sum Based Case 3	19.94	245.46
	Path Length Based Case 1	20.96	269.95
	Path Length Based Case 2	18.30	196.24
	Path Length Based Case 3	15.82	188.22
1 MST + 2 RST	Area Based	2.55	3.13
	Sum Based Case 1	2.09	2.80
	Sum Based Case 2	2.25	2.57
	Sum Based Case 3	2.24	2.42
	Path Length Based Case 1	3.77	8.59
	Path Length Based Case 2	2.25	2.73
	Path Length Based Case 3	2.63	4.03

Table 4.9. Performance Comparison of 3 MSTs vs. 3 RSTs vs. 1 MST + 2 RSTs for a 400-node Domain

400 Nodes with 80 Border Nodes		Average Difference Ratio	Standard Deviation
3 MST	Area Based	1.12	1.11
	Sum Based Case 1	1.56	1.87
	Sum Based Case 2	1.16	1.11
	Sum Based Case 3	1.08	1.01
	Path Length Based Case 1	2.96	6.97
	Path Length Based Case 2	1.67	1.71
	Path Length Based Case 3	1.42	1.56
3 RST	Area Based	83.93	3212.87
	Sum Based Case 1	67.43	2174.08
	Sum Based Case 2	120.42	4572.15
	Sum Based Case 3	94.16	4377.51
	Path Length Based Case 1	143.77	6267.20
	Path Length Based Case 2	29.87	578.33
	Path Length Based Case 3	123.37	5284.19
1 MST + 2 RST	Area Based	2.37	2.86
	Sum Based Case 1	2.52	3.96
	Sum Based Case 2	2.58	3.22
	Sum Based Case 3	2.54	3.06
	Path Length Based Case 1	4.78	12.95
	Path Length Based Case 2	2.50	3.64
	Path Length Based Case 3	2.51	3.86

From all the spanning tree simulation results, we can conclude that to summarize the topology of domains with 100-400 nodes, three overlapping MSTs when used for the aggregation perform much better than a single MST and have a low degree of distortion.

CHAPTER 5
CONCLUSION

We have proposed in this paper a method to compute the full mesh summary with a limited path heuristic and demonstrated that distortion degree in the scheme when this heuristic is used is low. An area-differences based performance evaluation scheme is proposed to evaluate the distortion degree in different aggregations. Finally, we have computed several MSTs and RSTs and evaluated their performances. We have shown through simulation results that MSTs perform much better than RSTs and three overlapping MSTs aggregate a domain much more accurately than one single MST. It was also shown that when the weights assigned to a logical link use the area representation or the sum based representations with functions $max(a1 * m1 + a2 * m2)$ or $[min+max](a1 * m1 + a2 * m2)/2$, where m1 and m2 are any two additive metrics associated with a link, the spanning trees perform better than when other methods of weight assignment is used.

REFERENCES

[1] Fang Hao and Ellen W. Zegura, “On Scalable QoS Routing: Performance Evaluation of Topology Aggregation”, *IEEE Proceedings of the INFOCOM’00*, March 2000.

[2] Xin Yuan, “Heuristic Algorithms for Multi-Constrained Quality of Service Routing”, *IEEE/ACM Transactions on Networking (TON)*, volume 10, issue 2, pages 244-256, April 2002.

[3] Fang Hao and Ellen W. Zegura, “Scalability Techniques in QoS Routing”, Technical Report, GIT-CC-99-16, Georgia Tech University, 1999.

[4] The ATM Forum. Private network-to-network interface specification version 1.0 (pnni 1.0), March 1996. f-pnni-0055.000.

[5] King-Shan Lui and Klara Nahrstedt, “Topology Aggregation and Routing in Bandwidth-Delay Sensitive Networks”, *IEEE Proceedings of IEEE Globecom’00*, November-December 2000.

[6] Whay Chiou Lee, “Spanning Tree Method for Link State Aggregation in Large Communication Networks”, *IEEE Proceedings of the INFOCOM’95*, pages 297-302, 1995.

[7] J.M. Jaffe, “Algorithms for Finding Paths with Multiple Constraints”, *Networks*, Volume 14, pages 95-116, 1984.

[8] E. Sullivan and R. Callon, “P-NNI Draft Specification”, *ATM Forum*, July 1994.

[9] K. Sivarajan and W. Lee, “Issues in the Aggregation of Link State Parameters in Hierarchical P-NNI Networks”, *ATM Forum*, July 1994.

[10] R. Vogel, R. Herrtwich, W. Kalfa, H. Wittig and L. Wolf, “QoS-Based Routing of Multimedia Streams in Computer Networks”, *IEEE Journal. on Selected Areas in Communications*, volume 14, no. 7, pages 1235-1244, 1996.

[11] T. H. Cormen, C. E. Leiserson and R. L. Rivest, “Introduction to Algorithms”, *The MIT Press*, 1990.

[12] Kenneth Calvert, Matthew Doar and Ellen Zegura, “Modeling Internet Topology”, *IEEE Communications Magazine*, June 1997.

BIOGRAPHICAL SKETCH

Almas Ansari

Almas Ansari was born and brought up in Bombay, India. She obtained her Bachelor of Technology degree in Computer Engineering from Dr. Babasaheb Ambedkar Technological University, India in 1999. After working as a Research Assistant to Dr. Gopal Shevare in the Department of Aerospace Engineering at the Indian Institute of Technology, Bombay, India, for two years, Almas has been working towards the completion of the Master's Degree in Computer Science at the Florida State University.