Multiple Constraint Based Routing and Probing in Data Networks

By
Prabhdeep Grewal

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of
Master of Science in Information Systems Science

School of Computer Science
Carleton University
Ottawa, Ontario
Canada

September 2006

Your file Votre référence
ISBN: 978-0-494-18352-6
Our file Notre référence
ISBN: 978-0-494-18352-6

Abstract

Resource intensive and time sensitive applications delivered over data networks often require explicit routes in the network that satisfy the resource need of the applications. Such requirements have escalated the research for finding novel route selection techniques that optimize the performance of networks. Reliability of data networks using such techniques can be further enhanced with alternate path routing techniques. One such solution that used constraint based routing with alternate paths is proposed in this thesis.

Route selection techniques rely on topology information for finding optimal routes in the networks. Keeping the topology information up-to-date, however, is associated with high communication overhead. So these two factors – optimality and network capacity – counteract each other so that making one factor better makes the other go worse. A probing technique was proposed to address this issue, especially in low resource networks. Actual simulation performed on the proposed design demonstrated slight improvement.

Acknowledgments

I express my deep sense of gratitude to my supervisor Late Professor Sivarama Dandamudi for giving me an opportunity to work on this research topic and providing guidance and inspiration. I have lost my guru, mentor and a very knowledgeable guide.

I would also like to very sincerely thank Professor Chung-Horng Lung for his very able guidance and all the help in completing my thesis when I needed it the most.

I sincerely thank Dr. Pramod Dhakal for his technical vision and guidance during the course of research at Eion Inc. I also acknowledge the research assistance provided through EION Inc, CITO and PRECARN during the course of the research reported in this thesis.

I take this opportunity to thank my family members and friends for their encouragement and support. I am also very thankful to my wife, Bipan, for her patience, encouragement and help.

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Full text
RoutProt	Routing Protocol
AIR	Adaptive Intelligent Router
BGP	Border Gateway Protocol
CBR	Constraint Based Routing
CI	Confidence Interval
CLI	Command Line Interface
CP	Certainty Probability
DataColl	Data Collection module
DCC	Delay Cost Constrained
DCLC	Delay Constrained Least Cost
DRC	Distributed Recursive Computation
ETBR	Enhanced Ticket Based Routing
ExcpHndl	Exception Handler
GUI	Graphical User Interface
H_MCOP	Heuristic based Multiple Constrained Optimal Path
IGP	Interior Gateway Protocol
LARAC	Lagrange Relaxation based Aggregated Cost
LCC	Least Combined Cost Routing
LEF	Linear Energy Function
MANET	Mobile Ad-hoc Networks
MCP	Multi Constrained Path
MPLS	Multi-Protocol Label Switching
MPMP	Multi Prepaths Multi Postpaths
NM	Normalized Margin
NS2	Network Simulator
NtfCmpl	Notify Completion
OSPF	Open Shortest Path First
OSI	Open Systems Interconnection
PathCalc	Path Calculation module
ProbHndl	Probe Handler module
ProbInit	Probe Initiator module
QoS	Quality of Service
RIP	Routing Information Protocol
RSVP	Resource Reservation Protocol
RSVP TE	Resource Reservation Protocol Traffic Engineering
TAMCRA	Tunable Accuracy Multiple Constraints Routing Algorithm
TBR	Ticket Based Routing

TblMngr	Table Manager
TopoDb	Topology Database
VPN	Virtual Private Network

Chapter 1

Introduction

With the increased usage of multimedia applications, video conferencing and online meetings, all network traffic can no longer be treated equally. It has become desirable to prioritize the traffic types and give differential treatment so as to ensure that the quality of service (QoS) provided over the network is acceptable for each application. The impetus for having a constrained based routing (CBR) has, therefore, grown stronger. The quality of service (QoS) satisfaction is becoming a requirement rather than an option in the networking industry.

Today, most of the network traffic is channeled using an unconstrained routing, which is also known as the best effort routing. The packets with best effort routing often experience delays and cannot provide speedy passage to time and mission critical traffic when the network bandwidth is exhausted along the most utilized paths even though part of the network may be underutilized.

The constraint based routing (CBR) refers to the path selection criteria based on multiple constraints satisfaction and finding the optimal path in the network. Another task to be accomplished by the constraint based routing is the proper network resource utilization.

The constraint based routing is more complicated than simple routing as it has to satisfy the constraints in addition to the normal path finding task.

Internet routing protocols like OSPF [Moy98a] use the shortest paths to find the optimal paths for routing. The only constraint being satisfied by the OSPF is the number of hops. Now the paths selected by the OSPF as the shortest paths may not be the optimal path for routing as it does not consider the resource optimization during path selection. The path selected by the OSPF routing algorithm may be congested or may not have enough bandwidth to satisfy the network traffic requirements. In case the paths are congested, the traffic may experience unacceptable level of jitter, loss and delays. This traffic could have been sent on the other paths with no loss and much less delay if the routing algorithm had knowledge about those optimal paths. Finding these optimal paths in the network to maximize resource utilization is the main objective of constraint based routing.

1.1 Routing Techniques

Based on when and how the routing paths are populated in the routing table, routing techniques can be broadly categorized into two groups: pro-active routing and reactive routing [You03]. They are briefly introduced below.

1. **Pro-active routing**: According to this approach, the routing algorithms find the routes for all possible source-destination combinations. The paths are stored in the tables and hence these techniques have fast response times as the paths are already cached in the tables by the routing mechanism. This scheme however has high storage and processing overhead. The storage overhead is further increased with the need for the alternative paths to support load balancing and fault tolerant services.

2. **Reactive routing**: In this technique, as compared to the previous one, the paths are computed only when the source has to send packets on the network to some destination. This technique has smaller overhead as compared to the previous technique but has slower response times as the paths are computed on the fly when required.

Another classification of routing techniques can be done on the basis of the information stored in the network to calculate the paths. The local path storage contains the information about the intra-domain routing and the global path storage contains the information about the inter-domain routing.

1. **Hop by hop routing**: The path computation is scattered among the nodes in the network and the nodes exchange the routing information about all other nodes in the network with their neighbors. The information exchanged is mostly the next

hop and the distance vector (distance from the source to the destination). As the nodes only have the information about the neighbors, so this type of routing suffers from slow convergence and routing loops. An example of such a routing algorithm is RIP (Routing Information Protocol). [Mal94].

2. **Link state routing**: In this technique, as compared to the previous one, the local state information is exchanged throughout the network. This state information is used to calculate the paths locally at the nodes. The global knowledge avoids the loops and provides more accurate path calculation. An example is the OSPF (Open Shortest Path First) protocol [Moy98b].

Yet another classification can be done based on how the routing information is stored and the search of the optimal paths is done in the network [Liu06] [Che98a].

1. **Source routing**: According to this approach, each node in the network has the global state information and uses this information to find the paths from the source itself. This information is kept updated by some link state algorithm like OSPF. This has the advantages same as that of the link state routing techniques of simplicity and being free of loops. The disadvantages are the computational and the communication overheads, and security concerns. Scalability can also be a problem because of the increased computations.

2. **Distributed routing:** The path calculation mechanism is distributed among all the nodes in the network. The routing decision is then made on the hop by hop basis. The advantage of this technique is faster response time and better scalability. The disadvantage is the lack of the global state information and hence looping may occur.

3. **Hierarchical routing**: In this technique, the nodes of the network are grouped into different levels. All the nodes belonging to the same level have the routing information about all other nodes in that level. The routing information about other levels need not be stored at the nodes of that particular level. This reduces the storage overheads in the network. This technique lies mid way of the two techniques discussed above and has the good points of both the techniques. This technique is highly scalable and the number of levels can be changed as the number of nodes in the network increases.

In hierarchical routing, inter-domain and intra-domain routing information is processed differently. The intra domain routing mechanism works within an autonomous system and these routing protocols are also known as the Interior Gateway Protocols (IGPs). The typical example of the intra-domain routing is OSPF which uses the link state algorithm (Dijkstra algorithm [Sta04]) to find the optimal paths for the traffic. For the inter-domain routing, BGP (Border Gateway Protocol) is an example which uses the distance vector routing technique (Bellman Ford algorithm [Sta04]) to find the optimal paths.

1.2 Constraint Based Routing

A constraint based routing involves more than one constraint in path calculations for optimal paths in the network. The constraint based routing can be classified as "online CBR" and "offline CBR" [You03]. The offline CBR, as implied by the name, finds the optimal paths using the mechanisms outside the network using the input as topology information and given constraint values. The advantage is the low computational overhead and disadvantage is low adaptability to the network changes and the pre requisite of the topology information of the network. The online CBR, on the other hand, operates within the network with the given traffic demands and the updated topology information. This technique adapts quicker to the network resource availability changes. The disadvantage is that it requires network resources to do the computations for the path calculations and hence suffers from high computation overheads.

Placing the CBR in the present network architecture, we see that the CBR lies in layer 3, the network layer, of the OSI model. Constraint based routing interacts with the MPLS (Multi-Protocol Label Switching) and the resource availability database to calculate the paths [Gra05]. MPLS is a protocol running over MPLS-enabled IP routers in a carrier-based core network. MPLS networks allow explicit predefined paths through networks as opposed to routes selected on hop-by-hop basis. OSPF determines these predefined paths and populate the topology database with these paths for all the source-destination pairs in the network. MPLS is flexible enough to define paths based on various constraints such

as available bandwidth. The CBR can find the optimal paths based on the constraint values specific to the traffic needs as compared to the shortest path routes found by the OSPF. The routes based on the path calculation module of CBR considering user specific needs will be more specific and will utilize the resources on network more efficiently.

MPLS integrates Layer 2 information about the network resources with Layer 3 and is also known as Layer 2.5 protocol. MPLS makes the network capable of routing the traffic around link failures or any congestion on the links in the network. The MPLS attaches fixed length labels to the packets in the network. The forwarding in the network is done based on these labels. The ingress node attaches the label to the packets. The intermediate nodes swap the incoming labels with outgoing labels and forward the packets based on the outgoing labels. Finally, the egress node removes the label from the packet and forwards it to the next node using the IP forwarding mechanisms. The CBR finds the optimal paths based on the requested constraint values for the traffic but does not actually reserve the resources on these paths.

To reserve the resources on the path calculated by the CBR, the RSVP-TE (Resource Reservation Protocol Traffic Engineering) [Awd01] is used. RSVP-TE is a set of traffic engineering extensions to RSVP (Resource Reservation Protocol). RSVP-TE extensions enable RSVP to be used for traffic engineering in MPLS environments. The CBR returns a full path from the source to destination which is used by the RSVP-TE to reserve the

resources on the found optimal paths. The reserved resources remain allocated that way until the routing protocol finds a new optimal path for the same source destination pair. If a new optimal path is found, the RSVP-TE will release the resources on the previous path and reserve resources on the new optimal path.

The CBR runs on the network and finds the optimal paths satisfying some constraints or optimizing some network resources. The resources in the network can be termed as the link metrics as they denote the quality of the links and the constraints have to find the paths with metric values satisfying the constraints. The network can be depicted as a graph with the nodes as the vertices of the network and the edges as the links between the nodes. The graph shown in Figure 1 depicts a network with seven nodes. The links are depicted as the edges of the nodes depicted as the vertices of the graph.

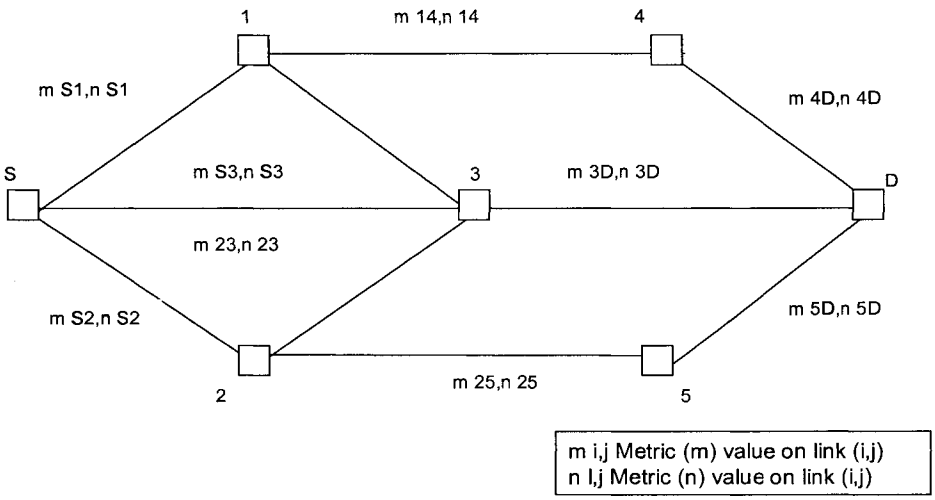

Figure 1 Graph Depiction of a Network

To see the importance of CBR, the authors of [Mal04] have discussed this technique which uses the qualities of satellite orbits to find an efficient routing mechanism for the future space Internet. The authors discuss the successful implementation of IP in the space Internet and the conversion of the multiple constraints to a single constraint using the weights which define the importance of the constraints.

1.3 Constraint types

In the graph shown in Figure 1, the nodes of the graph are numbered from 1 to 5 and two nodes denoted as source and destination. The edges are shown as non-directional lines joining the two nodes on the network graph. This means that the links are bidirectional and have same resources available for bidirectional traffic on the links. The link resources are depicted as m (node1, node2) and n (node1, node2) link markers to represent the metric value of type m or n, respectively, on the link joining node1 and node2.

There are basically three different types of constraints according to the objectives. The first objective of the routing algorithm is to find a path from source S to destination D with added value of the link metrics less than a pre-defined value. This type of constraint is known as the *additive constraint.* These are also known as the path metrics as the metric values are added for all the links along the path and should satisfy the constraint value. The sum of all the metric values on the links of the path to be selected should satisfy the constraint say c1. The other objective of the CBR is to optimize the second metric such that the value of metric on all the links is less than the constraint value, say c2. This type of constraint is called the *min/max constraint.* This type is also known as the link constraint as the link metrics along the path are not added up along the path whereas should individually be less than the constraint value. The third type of constraint for which the metric value multiplied for all the links on the path should satisfy the constraint value, such a constraint is called the *multiplicative constraint.*

The metrics or the resource values of the links used in the path calculations in the network are defined as follows:

1. **Bandwidth**: - This metric is an example of min/max constraint. The routing objective would be to find the path which has all the links with bandwidth value greater than the constraint value.

2. **Delay**: - This metric is an example of additive constraint. The routing objective would be to find the path which has added value on the links along the whole path less than the constraint value. This denotes the delay experienced by the packets on the links in the network.

3. **Hop count**: - This metric is of additive type and the value of the metric is considered to be equal for all the links and the added value has to be optimized or has to be less than the constraint value. The OSPF optimizes this metric value to find the optimal paths in the network. This denotes the distance from the source to destination in the network.

4. **Loss probability**: - This metric is of multiplicative type and the value of this metric is multiplied for all the links on the paths in the network and the final value has to be less than the constraint value. The metric denotes the acceptable loss rates on the links with normal congestion rates on the network.

5. **Cost**: - This metric has different definitions according to the network and denotes the cost of using that link. This can be a measure to indicate the resource usage on that link. This is a configurable metric and can denote various resources of the links.

The CBR mechanism, when has to find the optimal paths for more than one constraint, becomes more complicated compared to simple routing mechanisms. Considering more than one additive constraint in the path calculations makes the problem becomes NP hard [Cor02]. The min/max constraints are easy to handle as the paths that have the links not satisfying the constraint value can be pruned. For example, in Figure 1, we have second constraint to be bandwidth value not exceeding 20 units. Now the links in the network having less bandwidth than the constraint value can be taken off from the network and the rest of the topology can be input to the routing algorithm. There are different ways to solve the multi-constrained problem which are discussed later in the thesis.

1.4 Contributions

The path calculation mechanism of CBR relies on the accuracy of topology for validity of the optimal route selection. The topology accuracy is based on the information packet flooding mechanism in the network. It depends on the frequency of information exchange among all other neighbors in the network. The flooding can be done more frequently in

the network to improve the accuracy of the topology information. Increase in the frequency incurs higher communication overhead. However, if the frequency of the flooded information is decreased, the communication overhead can be reduced. Having lower frequency of topology updates will reduce the accuracy of the topology information and therefore will reduce the accuracy of path calculation module. This is due to the fact that the topology may not reflect the actual status of nodes/links at a given time for path calculations. To improve the accuracy of the topology information while not increasing the communication overhead, we introduce the probing mechanism.

Probing mechanism is used to probe nodes in the network to find the topology information and update the topology database accordingly. However, the probing mechanism will lead to high communication overhead in the network. This overhead will be further increased for larger networks where more probes and replies are required. The research motivation is to reduce this communication overhead and increase the accuracy of route selection module using the topology information updated using the probe replies. In order to achieve this, selective probing mechanism is proposed, according to which a small percentage of all nodes in the network are probed. The mechanism then updates the topology database using the probe replies.

Selective probing was carefully studied and tested for its accuracy in the area of path calculation. This thesis discusses how the selective probing technique would fit in the constraint based routing protocol architecture in the networks. Chapter 5 shows the

simulation setup to simulate the probing mechanism. It verifies the claim that probing small percentages of nodes in the network can increase the efficiency of optimal path selection significantly. Selective probing will be very useful in accurately calculating the optimal paths in the limited resource networks like the ad hoc sensor networks. Another factor for sensor networks to be best suited for the probing based routing protocol implementation is the highly dynamic network structure as the network topology is constantly changing in the ad hoc sensor networks.

1.5 Outline of the Thesis

In the Chapter 2 of this thesis, a background survey report on constraint based routing in the networks is discussed. The techniques presented in existing research work are discussed along with the background in probing. Chapter 3 discusses why the problem of multi-constraint routing is an important issue to be addressed and solved using the probing technique. In Chapter 4 design of a routing algorithm for the networks is discussed and how probing will fit and interact with the other modules of the routing protocol. In Chapter 5 the simulation setup and results for the study of the affect of probing on the path calculations are studied.. In Chapter 6 a summary of thesis is given and a conclusion is drawn from the study and the results.

Chapter 2

Background

2.1 Introduction

Constraint based routing, as discussed in the Chapter 1, is used to efficiently utilize the resources in a network based on customer-specific needs. Increased usage of high resource utilizing applications in the networks like video conferencing and video-on-demand, online gaming and resource sharing has catalyzed the need to have these routing mechanisms. Various researchers have come with various solutions for specific constraint needs to address this problem. In this chapter some of these techniques from various research resources are discussed. These mechanisms are more complex than the simple shortest path routing techniques such as the OSPF algorithm.

The route selection algorithms base their route selection decisions on the topology information stored in the topology database. As discussed later in the thesis, the topology database information has to be up-to-date so that the route selection algorithms have the correct information to select the optimal paths in the network. The nodes in the network usually transfer their state information in the network to keep this information updated. However, the frequency of these updates ensures the correctness of the topology database and hence the routes. On the other hand, frequent updates increase the communication overhead and use the available bandwidth for data transfer. To address this problem,

probing is discussed, which is being used in distributed systems [Eag86] to find the lightly loaded node. Probing selected nodes in the network ensures the communication overhead is not increased in the network and also provides up-to-date information. The efficiency of probing will be evaluated in the simulation Chapter 5.

2.2 QoS routing and Policy Based Routing

Two flavors of the CBR that have been in the discussions so far, namely policy based routing [Kar98] and QoS routing [Lui06]. The policy based routing mechanism involves the administrative constraints for the networks. The routing decisions in policy routing are not based on the source and destination. These decisions are based on the properties like the packet size, protocol used or any other network topology independent property. The policy routing mechanism is applied to all the traffic flows in the system as opposed to per flow routing techniques in the QoS routing. The policy based routing mechanism can be used by the administrators to restrict the malicious access by the unauthorized users. Other objectives of the policy routing are cost savings and load balancing [Lui06]. The policy based routing is applied along with underlying routing protocols. The policy based routing is applied before the QoS routing as policy based routing has more restrictive constraints as compared to the QoS routing constraints. The policy based routing constraints are either manually set up by the administrator or are exchanged by the nodes in the network. The policy based routing has been defined in [Kar98] as the

ideal for the VPNs (Virtual Private Networks). The use of policy based routing is found in Cisco Inter network Operating System (Cisco IOS^{TM}) Software Release 11.0 [Kar98].

The QoS routing, on the other hand, selects the optimal paths from the network based either on the constraints as required by the user or for optimizing the network resource usage. QoS routing has to satisfy one or multiple constraints for path selection and has to propagate the information throughout the network, which makes it complicated. Solving the multi-constrained problem with more than one additive constraint is a NP hard problem and hence cannot be solved in polynomial time. There are many heuristic algorithms proposing solution to this multi-constrained problem which are discussed later in this thesis. The QoS routing can associate priorities to the constraints as presented by the authors in [Mal04] and find the different paths accordingly. However the terms, CBR, QoS routing and policy based routing can be used interchangeably as they all consider some constraints in the path calculation mechanisms for the network.

2.3 Overview of Path Calculation in CBR

The most popular IGP, OSPF [Moy98a][Rab00], computes the shortest paths for the source destination pairs. The extensions to OSPF are discussed in [Apo98] to enable QoS with normal routing techniques. The link state algorithm is run on the network and the information of the resources is sent to the neighbors by all nodes in the network. It is known that the constraint based routing has become essential mechanism in order to

support applications requiring QoS support. Path calculations until now have included not more than one constraint. The multiplicative constraints can be changed into additive constraints, for example, by using the logarithmic function and hence are considered to fall in the same category and are known as the path constraints. The values of these resources for all the links get accumulated on the considered paths. The min/max constraints are called the link constraints as the values of the resources for the links do not add up during path calculation mechanisms. The link constraints like the bandwidth are easy to deal with as the links not having enough resources can be pruned from the path calculation topology [Ma97][Goy99].

2.3.1 CBR with two constraints

Two different scenarios are there for two constraints problem, the first one with both the constraints being additive (or one additive and the other multiplicative) and the second one having one additive and one min/max constraint. The second scenario as discussed before is pretty simple as we just prune the links with not enough resources for the min/max constraint and use the remaining topology to find the optimal path using the single constraint. The authors of [Jia04] have used the second scenario with two constraints (bandwidth and hop count) to devise a scalable routing mechanism providing alternate routes.

The authors of [Jia04] discuss the tradeoff between the high and low frequency of state updates in the link state algorithms used today. Lower frequencies of updates can

misguide the routing mechanisms while higher frequencies lead to communication overhead in the network. The authors propose an alternate path scheme that deals with the inaccuracy of information due to low frequency of updates. The routing mechanism proposed will calculate K paths that will either be shortest paths or would be the paths with highest minimum bandwidth on the links. The algorithm proposed in [Jia04] calculates these paths based on the major and minor weight terminology. If the path is the best according to the hop count metric (that is with lowest hop count) the major weight is the hop count and the minor weight is the bandwidth and hence is a hop count optimizing algorithm. The roles of major and minor weights are interchanged in case of the bandwidth optimizing algorithm. The path selection can then be done from the paths returned by the algorithm and the solution has one primary path and K-1 alternate paths from source to destination. Another proposal given by [Liu01] also gives K paths but the number of constraints can be more than two.

2.3.1.1 Two additive constraints

What is discussed in the last section was a problem with one additive and one min/max constraint which is not that difficult to deal with. However, now the problem with two additive constraints is discussed. The two additive constraints make the path calculation a NP hard problem [Cor02].

The first technique we will discuss in this section is proposed in the [Che98b] and is a heuristic algorithm for the multi-constraint based routing problem. The proposed idea is

to first convert the NP hard problem to a polynomial time solvable problem and find the solution for the new defined problem. Two additive constraints considered in this proposal are cost and delay. The multi-constrained path (MCP) problem in a graph depiction G (V, E) is defined in [Che98b] as MCP (G, s, d, m, n, c1, c2) and the path selected has to satisfy both constraints (c1, c2) such that 'm' added along the path from source (s) to destination (d) is less than c1 and 'n' added along the path from source (s) to destination (d) is less than c2.

To convert this NP hard the problem into a polynomial time solvable problem a new metric function is created, that converts the metrics m (or n) into a new metric m' (or n') using the function as defined in [Che98b]

$$n' = n * x / c2 \quad \text{.....(1)}$$

where x is a positive integer. We then have the simpler problem to solve defined as MCP (G, s, d, m, n', c1, x). The value x is an evaluating factor for the time complexity of the routing algorithm. With an increase in the value of x, the probability of finding an optimal path from the source to destination can be increased. This is explained by the fact that as the value of x (that is the new second constraint), is increased, we are loosening the bound on the metric value and hence the probability of path to be found increases.

The authors have also proved that the solution space for MCP (G, s, d, m, n', c1, x) is also the solution space for MCP (G, s, d, m, n, c1, c2) and not lose anything with the conversion described above. The converse of the statement, however, may not be true. The extended Bellman-Ford and Dijkstra algorithms are described in [Che98b]. Similarly a new metric m' can be defined

m' = m * x / c1 ....(2)

The above conversion is similar to the conversion in (1). The problem can be then defined as MCP (G, s, d, m', n, x, c2). Similar to the above discussion, this too contains all the solutions for MCP (G, s, d, m, n, c1, c2). Hence the proposal is to find the solution for the MCP (G, s, d, m, n', c1, x) using the extended Bellman-Ford and Dijkstra algorithms. If a solution for the problem with metric n' is found then return the solution; otherwise, find the solution for the problem MCP (G, s, d, m', n, x, c2). The value of x as defined by the authors is taken to be x = c * d where c is some positive integer and d is the distance between the source and destination.

Another technique for two additive constraints is the one described in the paper written by the authors of [Guo99]. The technique discusses an algorithm which takes delay and cost as the two additive constraints for the CBR. The problem, defined by the authors as DCCR (Delay Cost Constrained Routing), is to find the paths satisfying the delay constraint and optimizing the cost. This means that for a pair of source and destination,

the objective is that the delay added up on all the links for the chosen path should be less than the constraint value and the cost added up for all the links on that path should be less than the accumulated cost for all the candidate paths for that source and destination pair. The authors present an improvement for the algorithm at the end of the paper called SSR (Search Space Reduction) + DCCR (Delay Constrained Least Cost) that uses the search space reduction techniques. The authors propose to find the solution to the DCLC problem using the SSR + DCCR algorithm that has the same time complexity as the simple Dijkstra algorithm.

The DCLC problem is also called the Constrained Optimization problem as it finds cost optimized paths that have to satisfy the delay constraint. The objective of the DCCR algorithm, however, is to solve the DCC (Delay Cost Constrained) problem. The authors discuss that these problems (DCC and DCLC) are both NP hard. The essence of the technique is to convert the DCC problem into DCLC problem and find a solution for the DCLC problem. This is done as finding the solution for DCLC problem is easier than DCC and the DCC can be converted to DCLC using a cost bound for DCC. This cost bound is such that the solution to the DCLC problem is a solution for the DCC problem too. To accomplish this task the cost bound is kept loose to admit more solution paths for the problem. The algorithm to solve the DCC problem, first finds a solution path with least accumulated delay on the paths. The cost of this path is used as the cost bound for the next path finding procedures. If the algorithm can not find any path for the cost

constraint it returns the minimum delay path back that it found in the start of the algorithm.

To use the DCC problem approach we have to find the paths satisfying the delay and the cost constraints for the optimal path solution. As discussed earlier that the time complexity of the algorithm proposed in [Guo99] is the same as the Dijkstra single metric algorithm, the authors propose a conversion function that converts the two metrics into a single metric. In [Nev98], the proposed TAMCRA algorithm uses a nonlinear conversion function justified by the fact that the linear function does not do justice to the two constraints. The authors of [Guo99] discuss that this conversion function works well for non-bounded path calculations. The problem definition for DCC in [Guo99] differs from the TAMCRA and hence the authors propose a new metric conversion function that gives priority to the lower cost functions. The weight function W definition used for solving the DCC problem is defined as: -

If D (path[i][u]) ≤d and C (path[i][u]) ≤c

Then W (path[i][u]) = D (path[i][u]) / (1 – C (path[i][u]))/ c

Else

W (path[i][u]) = infinity

The D (path[i][u]) denotes the accumulated delay on the path from node i to u and C (path[i][u]) denoted the cost accumulated on the path from the node i to u. The c and d

denote the cost and delay constraints, respectively, as for the DCC problem. With this conversion function the weight grows exponentially with the growth in cost and linearly with the delay. Using this conversion function along with the path finding algorithm the approach finds a solution to the DCC problem.

Another technique presented in [Jut01] uses the Lagrange relaxation based method to find solution for the CBR with two constraints. The same delay constraint and cost optimization problem is formulated as the DCLC (Delay Constrained Least Cost) problem. Lagrange relaxation is used for calculating lower bounds, and finding solutions for this problem in [Jut01]. Lagrange relaxation method is used to solve the Traveling Salesman Problem in [Hel70]. The objective of the technique presented in [Ju01] is to keep the cost minimized while keeping the accumulated delay below the constraint value for the optimal path solution. The weight assignment problem has itself been an area of research for a long time. The conversion functions are used to convert more than one metric to a single metric as it is easier to solve routing problems with single constraint within polynomial time. One of the proposals for the single metric for more than one constraint is denoted by the bandwidth value divided by the product of the delay and the loss probability when these three constraints are considered for the CBR. Doing this however, we lose the individuality of the constraints as we can no longer ensure that the path minimizing the combined metric will satisfy the constraints individually. The problem is similar to the DCLC problem discussed in the previous technique. The authors in [Jut01] propose an algorithm that uses the modified cost function $c' = c + x * d$.

The algorithm proposed by the authors is called the LARAC (Lagrange Relaxation based Aggregated Cost) algorithm that minimizes the cost function c' defined in the previous paragraph. The basic idea for this algorithm is similar to the DCCR algorithm discussed before. The algorithm first uses the original cost c and finds the optimal path and checks whether it satisfies the delay constraint. If the delay constraint is satisfied, the algorithm stores the path and the stops the search and returns this path as the optimal path for the problem. However, if the path does not satisfy the delay constraint, the algorithm continues to find the solution for the modified cost function (using modified x value) as defined above as c' after storing the path as the best path not satisfying the delay constraint.

A similar approach is defined in one of the two techniques described in [Nor02] that uses the Least Combined Cost Routing (LCC) algorithm which finds the path that optimizes this combined cost. The combined cost metric includes multiple metrics, for example, the link utilization, bottleneck bandwidth on the path, etc. The combined cost metric is chosen such that it represents the load on the links on the path. The heavily loaded paths are avoided even if they have enough resources for the traffic. This combined cost metric is defined by the authors in [Nor02]. The second technique discussed by the authors uses a parallel probe algorithm which avoids the drawback of previous technique requiring the nodes to keep the statistical information about the whole network. The parallel probe technique sends out the probes in the network that traverse the whole network and collect

the routing information. The parallel probes are sent on the pre-determined paths for the source-destination pairs in the network. The probes sent in parallel have the field indicating the number of probes sent for the source-destination pair and the last hop collects all the probes and selects the best path based on the probe information. The number of probes indicated in the field is used to ensure that all probes reach the destination. In case some of the probes do not reach the destination, it uses the timeout mechanism and selects the best among the ones that reached it.

The authors also give a comparison study of the two techniques they discuss in their paper. The LCC technique has longer call setup time as it finds the paths on demand; whereas, the parallel probe technique uses the pre-computed paths and uses these paths to reach the destination and therefore has lower setup time. However, considering the message overhead it is evident that the parallel probe technique sends out more than one probe in the network. The LCC, on the other hand, requires just one reservation packet to be sent for the source to the destination. The main advantage of the parallel probe technique is the alternate path availability for each source-destination pair. The set of pre-computed paths on which the probes are to be sent form a set of paths from which the one optimal path is chosen and rest of them form the backup paths. In the LCC and the similar techniques discussed above, on the other hand, we have to recompute the paths from the source to destination in case of failure of the optimal path. In the same paper, the authors have presented the parallel probe technique for the inter domain routing too.

The probes are then divided in two groups, one sent within the domain called the micro probes and the ones sent to the other domains called the macro probes.

Another technique to handle these two constraints is the ticket based routing (TBR) that is quite similar to the parallel probe technique. The delay-constrained least-cost routing problem as defined before also is to find the least cost path satisfying delay constraint. The authors of [Che98c] propose the TBR technique. The TBR technique does not optimize the ticket forwarding technique which was pointed out by the authors of [Xia02a]. The authors propose Enhanced Ticket-based Routing (ETBR) algorithm. The ETBR algorithm tries to improve the efficiency of ticket probing by two techniques. The first technique uses color based ticket distribution which uses the green and yellow tickets to flood the network. The tracing information of green tickets and yellow tickets is kept separate by the algorithm. The second technique discussed uses historical probing results to optimize ticket probing. The authors present the simulation results to show that the performance of the ETBR is much higher than the TBR and has similar message overhead and path finding probabilities.

2.3.2 CBR with multiple constraints

The previous techniques discussed were dealing with two additive constraints. In this section multiple constraints are considered for the CBR. The techniques discussed in this section do not depend on the mathematical properties of the constraints and the number

of constraints considered. We start with the flooding based routing technique which is proposed in [Pun99][Son00][Dai02]. The technique discussed in these papers resolves the over reservation problem faced by most of the multi-constraint routing techniques. The over reservation is caused by the probes which traverse the network and reserve the resources on their paths from source to destination. These flooding algorithms were proposed by many authors including the authors of [Kwe99]. However, in [Son00], the problem of over reservation of the resources was being addressed and resolved. The comparison of [Kwe99] and [Son00] shows the better performance of technique discussed by authors of [Son00], because of the smaller blocking probability, lower setup time and better scalability.

The flood and prune algorithm defined by the authors of [Son00] uses two reservation states called 'soft-state' and 'hard-state'. When a probe reaches a node, the node reserves the resources for that flow in the soft state. The node converts the state from soft to hard when it receives the acknowledgment from the last node. The soft state reservation allows the best effort traffic to use the resources on those links; however it does not allow the other probes to reserve the resources on that link. The algorithm uses three techniques to avoid the problem of over reservation on the links in the network.

- Quick pruning: release the resources reserved in excess
- Destination sends the acknowledgement after receiving the very first reservation request

- Prune forward: release resources on the non-QoS routes.

To understand how the technique uses the above three steps to avoid over reservations in the network, four functions to be performed by the nodes are discussed, as defined in [Son00].

1. The node, when receives a reservation request, reserves the resources in the soft state and forwards the packets on the ports that meet the QoS requirements of the request.
2. The node rejects the duplicated copies of the requests and releases a packet upstream indicating to release the resources reserved by that duplicated request.
3. If the node receives the rejection from all the neighbors to which it had forwarded the reservation request, it immediately releases the resources and sends back the signal to the upstream node to release the resources they had reserved in the soft state.
4. When the node receives the acknowledgement for the downstream node, it reserves the resources in hard sate for this connection and performs two additional tasks:
    - It transmits the confirm signal to the upstream nodes from which it received the request; and secondly

- Sends the prune forward packets to the other downstream nodes to which it had sent the reservation requests in the network to avoid the over reservation problem.

A distributed routing technique is proposed by the authors in [Che99]. The source routing has several disadvantages like frequent update flooding in the network increasing the communication overhead and secondly the scalability problem. The distributed routing technique is called DRC (Distributed Recursive Computation) and is described by the authors [Che99]. According to this technique, each node does its computation locally with the local state information and spawns the child processes at some of the neighbor nodes. The authors also talk about the information reuse methodology. Each node carries out its local computation and keeps this information along with the timestamp. Whenever the parent node invokes the child process for that node, the child node checks for the timestamp for the computations it has already done and stored results. If the information is stale, that is the information is older than the timeout interval, the child node recomputes the path before returning the path to the parent. Otherwise, if the timeout is not yet being observed for the timestamp, it returns the path it had previously computed. This information reuse reduces the computation complexity of the algorithm and the overhead of this algorithm depends on the value of this timeout interval.

Another technique that was presented in the paper by authors of [Fan99] discusses a solution for multiple constraints based routing with inaccurate state information. The

authors debate that the accurate state information which forms the basis of many algorithms is not possible in the distributed environment. The basic idea of the algorithm is that it uses this uncertainty to device a new factor called certainty probability on which the algorithm is based. The uncertainty is unavoidable in the distributed network environment as the state updates can not be frequent enough to cope with the highly dynamic state changes in the network. The algorithm aims to find the solution paths with low information update frequency without losing the efficiency. The information update is flooded in the network when the metric value of the link increases relatively more than a threshold value. The certainty probability (CP), for a requested value of metric say x, is defined as the probability that the link has metric value more than x. The CPs are multiplied for all the links along the path when path calculations are being done. The paths having larger value of CP accumulated should be preferred to the paths with lower CP value, as the ones with larger value tend to have higher chances to cater to the needs of the user.

Another technique to handle multiple constraints is discussed in the papers [Xia02b] [Kui03] that are based on the dominance of the links of the paths. In the paper [Xia02b] a limited path Dijkstra algorithm is used that computes a limited number of non-dominated solutions for each node in the network. The concept of dominance is discussed in detail by the authors of [Kui03]. The authors explain the importance of non-dominance of the constraints and the constraint values in the solvability of the multi-constrained routing problem. The dominance can be defined by taking a network scenario example with two

constraints. The path, say A, has the metric values which are both less than the metric values for path B for the same source-destination pair. The path B is called dominated by path A. So we know that path B cannot form the optimal solution for the source-destination pair in consideration. So to be a part of the solution space the path should not be dominated by any other path for the same source-destination pair.

The second factor for the solvability of the CBR is the values of constraints themselves. The constraint value being too strict will reduce the work space and complexity as well as the probability to find the solution for the problem. The constraint value being loose will allow more paths to form the candidate paths and increase the complexity along with the increase in the probability of finding solution to the problem. The second affecting property of the link metrics is the link correlation. The links should display negative correlation to be one of the candidate solutions. Two links A and B are said to have positive correlation if one of the links has all the metric values less than the metric values of the other link. If the links display positive correlation, the links are dominated and hence would not be included in the solution space. Hence the links should display negative correlation to be included in the solution space.

Another approach to solving the multi-constrained problem is defined by the authors in paper [Cui03] that uses the linear energy functions. The assumption is that all the nodes in the network have the complete global state information. The algorithm is based on one of the techniques discussed for the two additive constraints before. The algorithm uses

the linear energy functions (LEF) to convert the multiple path metrics to a single combined metric and then computes the optimal path using this converted metric. The metrics are converted into a single energy function using a multi-dimensional vector for which the sum of all the components is equal to one. The number of components is equal to the number of constraints considered for the problem. So the basic idea is to assign weights to all the metrics and calculate a single combined metric value for all the metrics and use that combined metric to find the optimal solution. A fixed number of solutions are found using different possible combination of energy values for the network. So the components of the vector denote the importance associated with the metrics in the path calculation. The problem with these types of solutions is loss of the individuality of the metrics and injustice done to each individual constraint.

To do justice to each constraint for the network, there have been a number of proposals [Nev98][Kor01][Kor99][Shi02]. Before discussing the techniques the concept of pre paths and post paths should be known. When the source has to route a packet to the destination, then at any given time when the packet is at a node, say n, then the path from the source to n is called the pre path and path from n to the destination is called the post path of the path. The authors in [Nev98] calculate a pre defined number of pre paths for each node in the network. These fixed number of pre paths are considered to have higher probability to form the optimal path from source to destination. The drawback of this technique is that it does not consider the post path that will get attached to this pre path. Therefore, whenever the algorithm finds a new pre path with shorter length, it replaces

the already stored pre path with this path as this newly found pre path is considered to be a better candidate for forming the optimal path solution. This replacement did not consider the post path that will attach with the new pre path. The post path associated with the new pre path may not form as optimal path as the previous pre path would have formed.

To overcome the drawback of the technique discussed above, the authors of [Kor01] propose a technique H_MCOP (Heuristic based Multiple Constrained Optimal Path) that associates a single post path with all the nodes. The post path for the nodes is considered to be a good option for forming the complete optimal path. The H_MCOP uses this post path to select the pre paths for all the nodes. The selection of a wrong single post path on which the selection of the pre paths is based can result in bad selection of the optimal path and hence lead to failure of the algorithm. In order to overcome the drawbacks of H_MCOP, techniques like [Kor99][Shi02] should be considered that compute more than one pre path and post paths for each node in the network. The MPMP (Multi Prepaths Multi Postpaths) technique discussed in [Shi02] searches the feasible path using a modified Dijkstra algorithm. The algorithm uses a metric called the minimum normalized margin (NM). The NM is used to find the path with the highest low margined metric compared to the constraint value. This specifies the strictness of the QoS constraint in finding the optimal path solution. The lower the value the less likely that path will form part of the solution. This technique has much higher probability of finding correct

optimal paths as compared to the two similar techniques discussed in the previous paragraph.

2.4 Scalability and Alternate Path Provisioning

As discussed before in the thesis, hierarchical routing is an efficient way to handle the scalability issues in CBR. The authors of [Lee03] and [Zap97] discuss the scalability issues in CBR and provide mechanisms to handle them. In case of a link state routing algorithm, every router in the network has to maintain the routing information of the entire network topology. This restricts the scheme from being scalable as the size of network information increases with increase in network size. Thus, the hierarchical source routing as discussed in [Lee03] are considered. The routing protocols like OSPF use the domains called areas to make the routing scalable. The routing then operates at two levels: inter and intra level routing [Moy98]. The authors of [Lee03] show through simulation that the hierarchical routing techniques have better performance when large network sizes are considered. Another important property of the networks discussed in the thesis is alternate path provisioning [Zap04][Bej03]. The alternate paths are required for reliable QoS support, load balancing, cost effective QoS provisioning as discussed in [Zap04][Bar92].

2.5 Probing

Finding optimal paths in the routing protocols is based on the topology information of the network which is stored in the topology database. The nodes of the network flood the topology information in the network to keep the information accurate. In order to make the path calculation mechanism accurate this information should be flooded very frequently. However, this will lead to the resource consumption on the links of the network. So the research motivation is to optimize the topology information update mechanism so that the accuracy is kept high while keeping the communication overhead low in the network. This problem can be mapped to the problem that is being studied in the field of distributed systems [Eag86][Mun98]. As the authors in [Mun98] talk about the parallel disk storage systems for multimedia applications, they refer to the solution for the similar problem for the distributed systems in which nearly 10% of the total nodes in the system are probed for the load and the requests in the system is sent to the least loaded probed node. The authors claim that even sending a fraction of the total requests to the probed lightly loaded node in the network can give almost the same gain as could have been gained by probing all the nodes in the network.

This thesis proposes a similar solution for Path Calculation algorithm where a few links/nodes in the network are probed to update the topology database. The design presented in the Chapter 4 of thesis includes the Probing module within the Path Calculation module. The performance gains similar to the claims made by researchers in [Eag86] are expected from the probing mechanism for the path calculations. The

simulation study presented in Chapter 5 will evaluate the effectiveness of 10 % probing. This means that the probing module will probe 10% of the total nodes in the network. If the simulation results show the effectiveness of the probing mechanism, then the Probing module can be included within the Path Calculation module for the Routing Protocol. If probing is found to be effective, it can be helpful in the network protocols and be extended to sensor networks where resources are limited.

Chapter 3

Problem Statement

3.1 The Problem Addressed

With ever increasing number of resource demanding applications running in the network, the optimal resource utilization has become a major concern for routing protocols. The application specific QoS needs form the constraints on the resource values of the links in the network. For example, a video streaming application may require a specific amount of bandwidth. This forces the path selector module of a router to find paths that have sufficient bandwidth available on all the links in the selected path. This defines the constraint based routing that finds paths in the network while enforcing the constraints on resources in network. Constraint based routing introduces extra communication and computation overhead for the routing protocol. For this reason the scalability should be taken care of by the routing protocol.

The resources available on links/nodes, also called metrics, are classified in three categories based on their mathematical properties. Additive and multiplicative constraints are also known as path constraints as the metric values are accumulated (added/multiplied) on the links on a path to satisfy the constraint. Min/max constraints are not accumulated on the links for the path calculation and are called link constraints.

1. **Additive constraints**: The measures of the link properties in network are being added along the paths and should satisfy the constraint value specified. The examples of the characteristics for additive constraints are delay, power consumption, hop count and cost. These resource values are added along the paths calculated and the total value either has to be less than the constraint value or it has to be optimized depending on the requirements.

2. **Multiplicative constraints**: The measures of these link properties in the network are being multiplied on the paths. The total value either has to be optimized or has to be less than the constraint value according to the requirements. An example of these types of constraints is the loss probability.

3. **Min/Max constraints**: The measures of these types of link properties in the network are used to find the paths with the highest minimum (or lowest maximum) value along the path. For example considering the reservable bandwidth in the network we find the path that has the highest minimum value among the paths available. Another scenario could be to find the path having this metric value more than the constraint value on each link on the optimal path.

Most routing protocols use the number of hops as the only constraint for finding optimal paths in networks. However, when protocols use more than one additive constraint to find the optimal paths, the problem becomes NP hard. The Min/Max constraints are easier to

handle as the links without sufficient resources are excluded from the path calculation mechanism. To handle multiple additive constraints, two techniques can be considered. The first technique handles all the constraints individually and hence has higher computation complexity. The second technique uses weight and metric pairs to find the aggregated metric and then optimizes this metric. The computation complexity is reduced but the constraints lose individuality that means that the technique can no longer specify the constraint value for a particular metric and can just optimize this aggregated metric. The first technique, however, handles each constraint individually while increasing the computation complexity. In both techniques the min/max constraint is handled similarly, that is the links not having sufficient resources are removed from the candidate paths list of the route calculation algorithm.

Another aspect of the routing and path calculation is the accuracy of the topology information. The topology accuracy is based on the information packet flooding mechanism that is how often the topology information is shared among all other neighbors in the network. Having a shorter time interval for topology updates, to improve the accuracy of the information incurs higher communication overhead. However, if this interval is longer, the communication overhead decreases and so does the accuracy of the topology. This means the topology may not reflect the actual status of nodes/links at a given time for path calculations. To ensure the accuracy of the topology information the probing mechanism is introduced.

Probing, as indicated by the name, is a mechanism to probe nodes in the network to find the topology information and update the topology database accordingly. However, the probing mechanism will lead to high communication overhead in the network and is worsened with bigger networks leading to more probes and replies in the network. The research motivation is to reduce this communication overhead while not incurring large number of errors in routing due to inaccuracy in topology information. The proposal is to selectively probe a small percentage of all nodes in the network and update the topology database according to the replies to such probes. Later in this thesis, the probing mechanism is simulated to verify the claim that probing small percentages of nodes in the network can increase the efficiency of optimal path selection significantly.

3.2 Why is this Problem Important to Solve

To cope with the increasing demand of the efficient utilization of resources without incurring overheads, the need arises for an efficient routing solution. The constraint based routing technique should optimally select the best paths in the network. The first step is to propose a mechanism that will accurately find the optimal paths in the network while satisfying the resource needs of the traffic. Many proposals have been made by researchers using different techniques to solve this problem. In this thesis, one of these techniques that uses a single converted metric for the multi-constraint routing problem is discussed. However, the main contribution comes from the second part of thesis which

explores the concept of probing to provide a balance of accuracy and efficiency of finding the optimal paths in the network.

The technique which depends on the topology information of the network needs to be accurate enough and also should not incur heavy communication and calculation overheads. To ensure this, thesis proposes the selective probing technique to provide a balance between the two factors. Selective probing has been used in the area of distributed systems [Mun98], and was carefully studied and tested for its accuracy in the area of path calculation, which is discussed in the Chapter 5. Chapter 4 discusses about how the probing technique would fit in the path calculation protocol architecture in the networks. Later in this thesis, the simulation model is discussed that was used to evaluate the efficiency of the probing. Selective probing will prove to be very useful in accurately calculating the optimal paths in the networks with limited resources like the sensor networks where the resources have to effectively utilized.

3.3 Comparison with the Existing Approaches

Other researchers have proposed, developed, and deployed a variety of constraint based routing protocols for optimal route selection. As discussed in the survey, there are two groups of protocols: one uses a single metric which is a representation of all the metrics for the network and optimizes that metric to find the routes in the network. The other group of protocols optimizes the metrics individually to find the best route in the

network. The second group gives individuality to the metrics and hence is more accurate as compared to the first group which loses the individuality of the constraints to find the optimal paths. The first group, however, is favorable, as they allow us to give priorities to the metrics in the network and are comparatively faster than the second group in finding the paths. A lot of research has been done in this area to find the optimal solutions in the network.

The contribution of this thesis comes from the probing technique to reduce the communication overhead in the network. The routing protocols need to have information about the metrics in the network to find the best routes. This information has to be propagated in the network which is done periodically which introduces communication overhead in the routing mechanisms. This information has to accurately find the best paths according to the present state of the network entities. The information has to be propagated frequently to have up-to-date information of the resources in the network; however, this introduces communication overhead. These two are counter acting factors as improving one degrades the other. To find a solution to optimize the contribution of both factors, selective probing was found to be successful in the area of distributed systems where the processors are selectively probed to find the most lightly loaded processor in the distributed systems where a portion of the load is sent to the most lightly loaded node. This technique has not been evaluated for its effectiveness in improving the route selection procedures. Selective probing reduces the communication overhead and also increases the accuracy of the route selection mechanism of the protocols. An

extensive evaluation of this technique is conducted for different scenarios of the networks and the efficiency of the probing in the route selection mechanisms is evaluated.

Chapter 4

Architecture and Design

4.1 Functional Description

Optimal routes have to be calculated for data packets so as to route them through a path that maximizes the utilisation of resources in the network. The path calculation is an important function to be performed before routing the packets. The shortest path routing that has been done in most routing algorithms is not enough to ensure the proper resource utilisation as the shortest path in the network can be congested and hence may not be the optimal choice for routing packets. Constraint based routing uses more than one constraint to find the optimal paths. Considering more than one constraint in the path calculations is a daunting task and even more difficult is to make the routing scalable. The motivation behind the design presented in this chapter is in making the path calculation mechanism in the network scalable and to adapt it to the routing. The path calculation decisions are made on line as the first data packet for a given destination arrives in the ingress router. This is called a reactive approach as opposed to a proactive approach used in most routing algorithms.

Figure 2 shows the interaction of path calculation module with other modules in routing domain. The Path Calculation module itself comprises of the sub modules namely, the Path Calculation algorithm which uses the information from the second sub module, the

Topology Database. The Data Collection module is responsible for collecting the network metrics which are stored in the Topology Database to be used in finding the optimal paths using the Path Calculation algorithm. The Routing Protocol uses the optimal paths calculated by the Path Calculation module to send the traffic in the network.

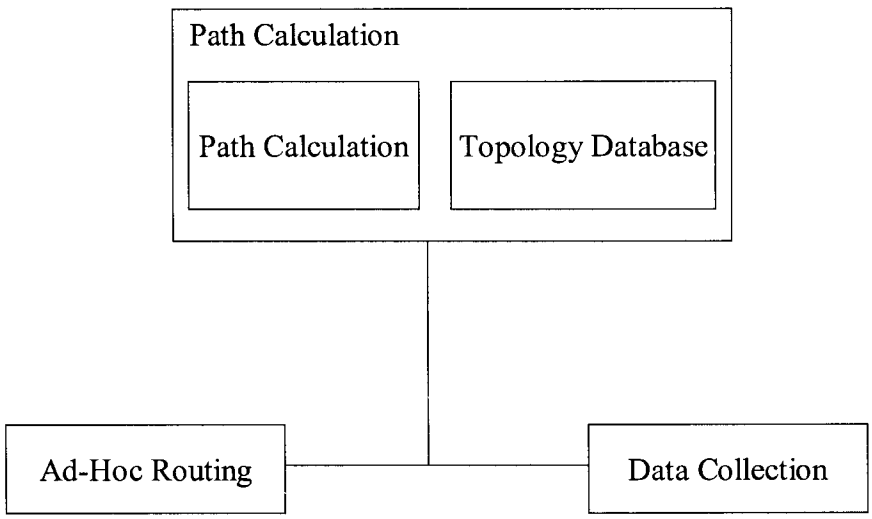

Figure 2 Architectural Design

4.2 System Deployment

4.2.1 Architecture

Figure 3 shows functional diagram of a router. It shows how the routing modules interact with each other. It shows three functional domains of a router, namely the Management

Plane, the Control Plane and the Data Plane. The focus of this research is the Control Plane. In the Control Plane, the Path Calculation module itself contains two sub modules, namely the topology database and the Probe Handler module. These modules will be discussed in detail in a later section.

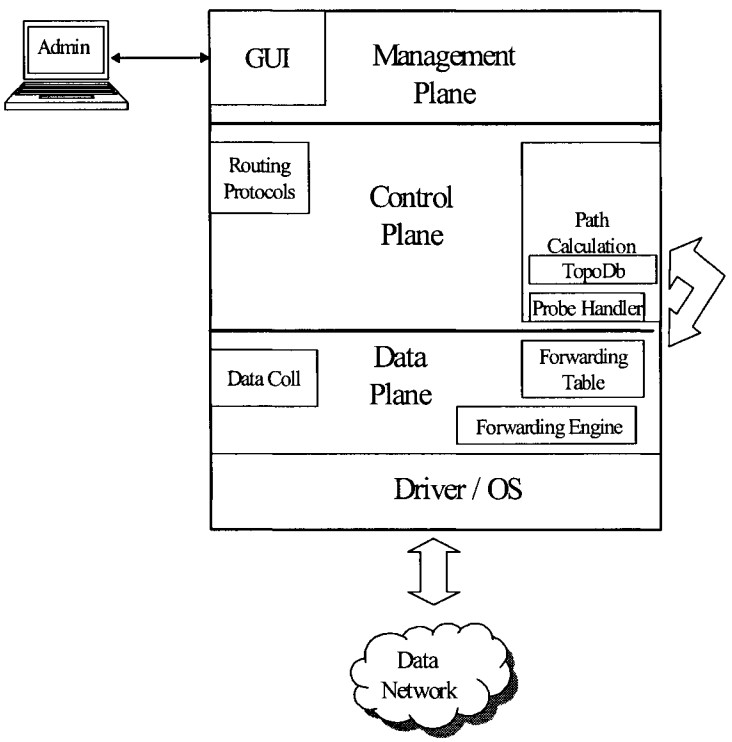

Figure 3 Functional Diagram of a Router

4.2.2 Module Configuration

The responsibility of the module configuration component includes five major tasks discussed below.

1. Display the list of constraints that can be utilized in path calculation. The following parameters must be listed:
    a. Bandwidth
    b. Jitter
    c. Loss Probability
    d. Delay
    e. Distance
    f. Hops
2. Allow the administrator to select which are configurable parameters, which are not to be considered in path calculation, and how many constraints are to be handled at most at a given time. Also, for the alternate path routing, the administrator can specify the number of paths required for each source destination pair in the network.
3. Allow the administrator to associate priorities to the metrics in the network. Based on the priorities, the metrics will be converted to a single additive metric.
4. Allow the administrator to set the nodes for probing within the network. The percentage of total nodes/links to be selected for probing can be on the previously

known path from a source to destination pair for which a path is already known to the system. Another option is to randomly select the nodes/links in the network to be probed.

5. Allow the administrator to query the TopoDb (Topology Database) and display specific information about the network. The administrator will have the following options to get information displayed from the TopoDb:
    a. All Data: all the fields of the database entries to be displayed.
    b. Subset of data: specific fields of the entries to be displayed
    c. One entry: just one entry to be displayed
    d. Many entries: more than entry to be displayed
    e. All entries: all the entries in database to be displayed

4.2.3 Dependencies

For Path Calculation to operate properly, the following dependencies must be satisfied:

The Path Calculation module has to have the interaction path set up with the Data Collection module as the database is to be filled with the information that is provided by the Data Collection module. The Path Calculation module can also send specific probes to the Data Collection module as discussed in the functional description section. Specific CLI's as discussed in the previous section should be there through which the probes can be sent by the Path Calculation module to the Data Collection module. To communicate with the Data Collection module the message ID should be known to the Path Calculation

module. The message ID is set by the administrator for the Path Calculation module to send the probe request to the Data Collection module. This is required only when the probing mechanism is enabled in the routing protocol and the Path Calculation module has to send probe requests to the Data Collection module. However, if probe requests are not sent by the Path Calculation module then it does not need the message ID for communicating with Data Collection module. These configurations can be handled using the methodologies discussed in the Module Configuration section. The Data Collection module should return the residual bandwidth on the links or any other link/node metric in the network.

4.3 Design

The purpose of the Path Calculation module is to return the optimal paths to the calling functions, which will be used to send the traffic in the network. The Routing Protocol module will use the path returned by the Path Calculation module to send the traffic in the network. The optimal path selection enhances the resource utilization in the network, as the paths chosen will be the ones with enough resources to satisfy the sender's needs. In the following subsections the graph theory is discussed to represent the network topology. Then the algorithm to find the optimal paths in the network is discussed along with an explanation of the probing mechanism. Finally the use cases for the modules are presented to explain the design of the routing protocol with the probing enabled for optimal route selection in the network.

4.3.1 Graph Theory

Graphs can be used to represent networks with the nodes as the vertices and the links as edges in the graph. The links between the vertices can be weighted and the weights can be used to represent the link metrics. Figure 4 is an example graph showing a network topology with bi-directional links. The nodes are marked a, b, c and the source as S and the destination as D. The links are marked with weights that may represent any link characteristic like residual bandwidth on the links. The link marked W<vertice1, vertice2> represent the metric values between the two vertices vertice1 and vertice2.

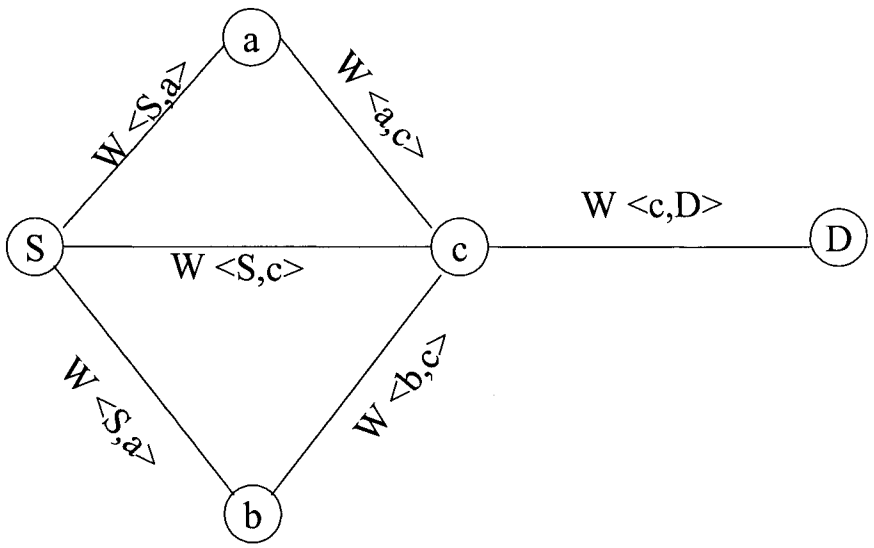

Figure 4 Network Shown as a Graph

4.3.2 Path Calculation

The algorithm used by the Path Calculation module is modified version of the **Dijkstra Algorithm** (used in the OSPF protocol)[Cor02]. The two constraints considered in the Path Calculation module will be any additive constraint like configurable cost and the residual bandwidth. When given the source, destination, cost and bandwidth constraints, the Path Calculation module uses the bandwidth constraint to eliminate the links not having enough bandwidth from the candidate paths. After pruning the links based on the constraint bandwidth value, the module applies the Dijkstra algorithm on the remaining links using the cost as the additive metric. The path found will be optimized based on the cost measure on the links in the paths. These two constraints, cost and bandwidth, are easier to deal with, as cost is an additive constraint and bandwidth is a non-additive constraint.

In the modified Dijkstra algorithm, the cost accumulated on the links from source till the node n in the graph represented network during the path calculation is denoted by r[n]. Function ExtMin(V) searches for the vertex u in the vertex set V that has the least r[] value. Function Union (S, {y}) is used to include a node y in the list of nodes denoted by S. The modifications done to the Dijkstra algorithm are:

1. The search space reduction is done by pruning the links from the graph that can not be on the optimal paths list. The links with bandwidth less than the constraint value are pruned from the list of edges.

2. The additive hop count metric used for the finding shortest paths using the Dijkstra algorithm is replaced with the additive cost metric. The weighted additive metric which is derived using the weights associated with the metrics is discussed later in this section. This weighted aggregated metric is used as the cost metric to find the optimal paths using the modified Dijkstra algorithm.

Modified Dijkstra Algorithm (G, w, s) // G: Graph, w: ..., s: ...

```
For each node x in Graph G
    r[x] := infinity
    parent[x] := null
End For
r[s] := 0                              // s is the source
S := empty set of vertices
V := set of all vertices
while V is not empty
    y := ExtMin(V)          //Extract minumum r[] valued node from V
    S: = Union(S,{y})
    For each edge (y,x) outgoing from node y
        If r[x] > r[y] + w(y,x)
            r[x] := r[y] + w(y,x)
            parent[x] := y
        End If
    End For
End While
```

In the algorithm above the search ends for the module when all the nodes are extracted from the set V, which means path for each node in the network is found. The weight w (y,x) in the algorithm is the cost value on the links joining nodes x and y. The cost accumulated so far on the link in the path calculation is denoted by the r[n] from source till the node n in the graph represented network. The objective of the algorithm is to find the optimal path with minimum cost on the paths of the network. However, this algorithm can handle more than one additive constraint by converting them into a single additive constraint and then using the above algorithm to find the paths. Also, to provide alternate path routing the path calculation module can find more than one paths and return the ranked for each source destination pair in the network. The number of paths required for each source destination pair in the network can be specified by the administrator in the configuration file.

To convert the additive constraints into a single additive constraint, the administrator can specify the weight associated with each metric. The formula for the weighted aggregated metric then uses the weights to convert the metrics to a single additive metric before applying the algorithm. These weights give the priorities to the constraints and can be changed to reflect the changes in the priority by the administrator. In (3) below WAM represents the weighted aggregated metric and the two metrics are represented by M and N for the link between any two nodes, i and j, in the network. The weights for the two metrics M and N are Wm and Wn, respectively, and specify the priority associated with each metric. The WAM can then replace the cost metric in the modified Dijkstra

algorithm to find the optimal paths in the network based on WAM instead of the metrics M and N.

$$WAM(i,j) = M(i,j) * Wm + N(i,j) * Wn \quad \text{...(3)}$$

4.3.3 Probing Methodology

The probing mechanism is used to make path calculation more accurate as it gives more up-to-date information to the Topology Database. The path calculation will then use the updated information to find optimal paths in the network. Probing is based on the last known state of the database and the probes are sent to the nodes that are known to exist in the network. Probes are sent to a very small percentage of the nodes in the network and hence reduce the communication overhead in network. The nodes to be selected can be the ones that lie on a previously known path from the same source to destination. Generally, the administrator selects 5 to 10 percent nodes in the network and probes them for the topology information updates. Increasing the probing nodes percentage to higher values diminishes the value of selective probing. The increases in the accuracy of optimal paths calculated would be evaluated using simulations and if proven efficient will noticeably change the performance of the path calculation module.

4.3.4 Assumptions

The Path Calculation module should have an access to a fully populated Topology Database (discussed in the Functional Description module). The source and destination are the input to the module and the Topology Database is the reference network on which the algorithm works to find the optimal path.

The information contained in the Topology Database may, however, not be up-to-date. The Path Calculation module sends requests for probing a small number of nodes to the Data Collection module. The Data Collection module probes the requested nodes and feeds back the collected topology information to the Topology Database. The Path Calculation module is then notified to proceed with the calculation using this updated database information. These will require the bi-directional communication enabled between the Data Collection and Path Calculation modules.

4.3.5 Use Cases

The Use Cases of Path Calculation (PathCalc), Topology Database (TopoDb) and Probe Handler (PrbHndlr) modules are being discussed in these three sub sections. In these sub-sections we briefly discuss the functionalities of these modules. The details of the Use Cases can be found in the Appendix with the Use Case Diagrams, Static Models, Message Sequence charts and the State Transition Diagrams.

4.3.5.1 Path Calculation

The Path Calculation module is an essential part of the routing protocol and does the path calculation for finding the optimal paths in the network. It interacts with other modules in the network like the topology database and the Data Collection module to find these paths. The Probing module introduced in this thesis also forms an essential part as it is used to keep the topology updated while keeping the communication overhead in the network minimal. Based on the module configuration the path calculation module can find more than one path for each source-destination pair in the network to provide alternate path routing and the load balancing. The main functionalities of the Path Calculation module are:

1. Administrator sets the port ids for the PathCalc to communicate with other modules like RoutProt (Routing Protocol) and DataColl (Data Collection).
2. Administrator sets constraints to be considered for the optimal path calculations.
3. The PathCalc module processes the path request sent by RoutProt or entered by the administrator and sends back the path. PathCalc uses the Topology Database to find the optimal path for the source destination pair satisfying the constraints.
4. PathCalc module validates the list of paths sent by the RoutProt module and returns the paths that passed the validation for the resources.
5. PathCalc module saves the last known state of the database in a configuration file and reads from it when it loses the information due to some failure.

4.3.5.2 Topology Database

The Topology Database is used to store the information about the network for the path calculation module to calculate the optimal paths in the network. The database is updated by the regular interval updates flooded by the nodes in the network. The Data Collection module is responsible for finding the metrics in the network and fill in the information in the topology database. The main functionalities of the Topology Database module are:

1. The Administrator (ADMIN) initializes the port ids that other modules use to communicate with TopoDb (Topology Database). It also initializes the storage structure to store the database entries in the Topology Database.
2. The Data Collection (DataColl) or GUI updates the topology information stored in the TopoDb by adding, deleting or updating the entries in the table.
3. The GUI or Admin can query the TopoDb to get the topology information. The following options are available to get the information displayed from the TopoDb:
    i. All Data: all the fields of the database entries are to be displayed.
    ii. Subset of data: specific fields of the entries are to be displayed
    iii. One entry: just one entry is to be displayed
    iv. Many entries: more than one entry is to be displayed
    v. All entries: all the entries in database are to be displayed

4.3.5.3 Probe Handler

The Probing module in the routing protocol is responsible to selectively probe the nodes in the network and update the topology information in the Topology Database. Based on the module configuration, the Probing module will probe selected nodes in the network to reduce the communication overhead as compared to probing all the nodes in the network. The probing is done by the Data Collection module in the network to find the link/node metrics and update the information in the Topology Database. The main functionalities of the Probe Handler module are:

1. The Administrator (ADMIN) initializes the port ids that other modules to communicate with Probe Handler.
2. The Probe Initiator can be based on a timer or the Path Calculation module in the network can initiate it when it requires a path from the source to the destination in the network.
3. The DataColl module, upon receiving the Probe Request, starts processing the probes on the nodes specified. Depending on the response of the nodes to the probes the DataColl module works on the information received as detailed in next two sub sections.
4. In case, the DataColl module does not receive any response from the nodes it probed, it will send an error message to the Probe Initiator indicating that the probing was not successfully performed in the network on the selected nodes. The

exception handler module would give the list of the nodes that failed in the probing procedure.

5. In the case that the DataColl module receives all the probe replies for the nodes to which it sent the probe requests, it sends out a signal to the Probe Initiator about the successful completion of probing.

Chapter 5

Simulation and Testing

5.1 Introduction

The design of the routing protocol discussed in the previous chapter details the interaction of the Probing module with the Path Calculation and the Topology Database. The Probing module, with the help of Data Collection module, selectively probes nodes in the network, extracts most updated metrics available through those nodes, and updates the topology database. The Path Calculation module uses this updated topology information and calculates the paths to reach the destination. The results obtained from the modified Dijkstra algorithm for Path Calculation module, and presented in this chapter, demonstrate in a simple but understandable way the correctness of the algorithm developed as the part of this research work. As discussed in Chapter 4, the Probing module attempts to considerably reduce the communication overhead and increase the efficiency of the routes by selectively probing the nodes in the network. Results of the simulation of the probing technique carried on a NS 2 Simulator [Ns06] are presented in graphs which depict the enhancements on the efficiency and accuracy achieved in updating topology information to the Path Calculation module.

5.2 Verification of Path Calculation Technique

Path Calculation module used the modified Dijkstra algorithm along with the topology database, as discussed in Chapter 4, for finding the optimal paths in the network. Because this module was one of the first to be developed during the course of this research, it was also the one that was used the most. This module has been working as intended for more than a year now and has been adopted successfully by the industrial sponsors of this research project. As part of this research work, the algorithm was implemented in C programming language that extracted the topology information from manually written input text files and calculated the paths to specified destinations. The input files were created deliberately with fixed set of expected outcomes. The results from the algorithm were then compared with the expected outcome and verified that the algorithm was able to produce the output accurately.

5.2.1 Verification Procedure

The verification was done for 20 different topology scenarios. The process of validating the correctness of the algorithm is shown in Figure 5. The topology input files were used to fill in the topology database, which was then used by the algorithm to find the optimal paths in the network. The input files had the metric information for all the links in the network. The weights were assigned statically to the metrics and the weighted aggregated metric was found by using the algorithm described in Chapter 4. Topology input files representing networks of varying size and topology were used to simulate networks and

the changes that occur in them. As the topology changes, the algorithm should have found the correct optimal paths for the given topology. The networks' topology and metrics were pre-determined so that optimal paths between any source-destination pair could be pre-determined. The optimal paths calculated by the algorithm were then compared with the expected results. Testing 20 test scenarios with different topologies demonstrated that results produced by the algorithm always matched with the expected results.

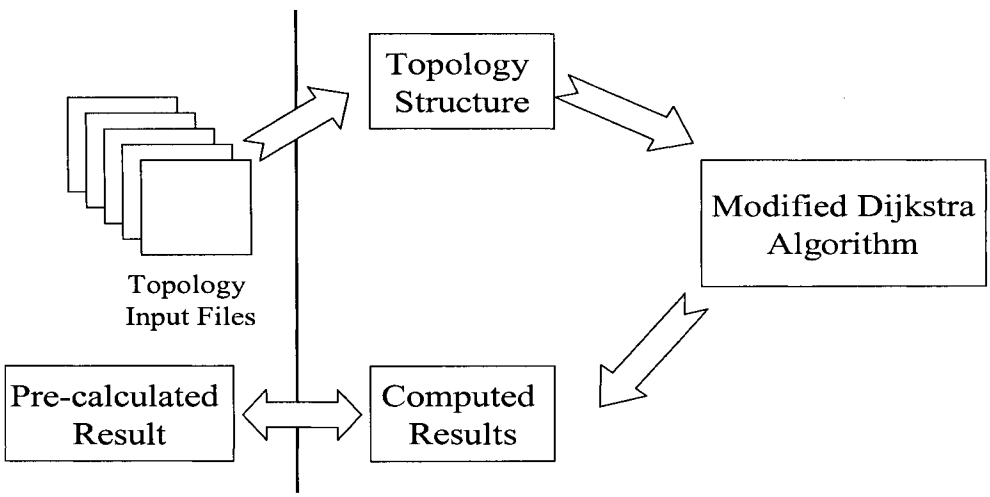

Figure 5 Verification procedure for path calculation

5.2.2 A Sample Test Scenario and Verification

In this section, one of the test network used to validate the correctness of the modified Dijkstra algorithm used in Path Calculation module, is discussed. As shown in the Figure 6, a sample network of 8 nodes was taken. The network has following parameters:

- Network has 8 nodes with 9 bi-directional links.
- Each link (i,j) in the network has two metrics (Mij and Nij).
- Weights are assumed to be 0.5 for both metrics.
- For each source destination pair two paths are found (optimal and alternate path).
- Paths are optimized for the weighted aggregated metric WMij.

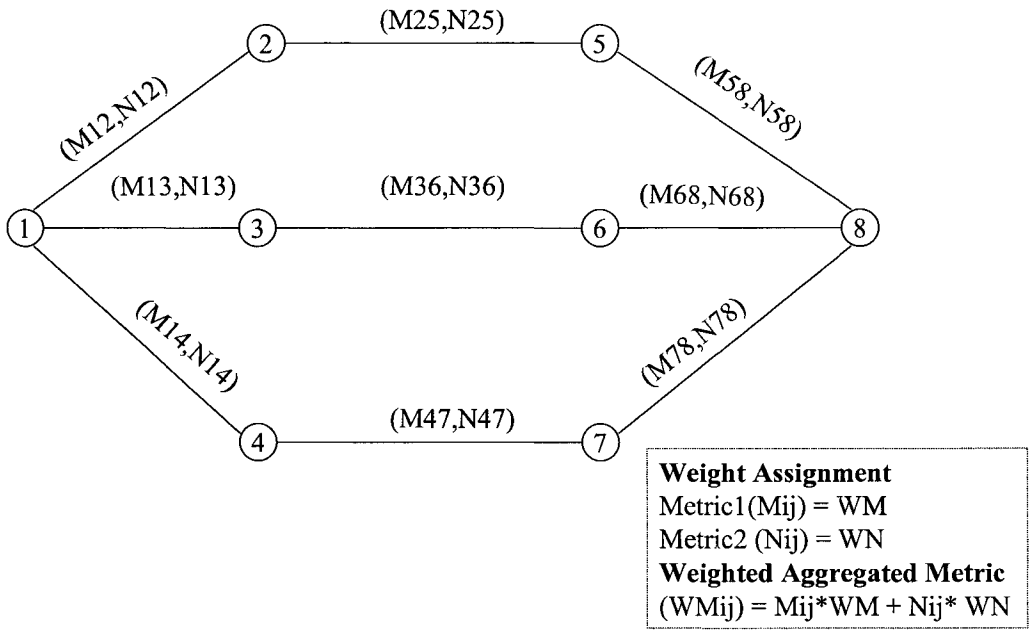

Figure 6 Network scenario for verification of modified Dijkstra algorithm

Figure 6 shows 8 nodes with bi-directional links in the network. For each source destination pair in the network, optimal paths with alternate paths were found. Input text files were used to define the metrics for all the links in network. Changes in the topology were simulated by changing the input text files with different metric values for the links and the routes were re-calculated. The results were than compared with the pre-calculated results to verify the correctness of the modified Dijkstra algorithm.

Example topology information is shown in Table 1. The links are listed in the first column and the corresponding value at time t1 and t2 are listed in column 2 and 3 respectively. At time t1, the paths were calculated according to the information in the column 1 and compared with the pre-calculated results. For example, at time t1 the optimal path returned for node 1 to node 8 by the algorithm was 1-3-6-8. The alternate path returned was 1-2-5-8. These results matched with the pre-calculated results. Also, to simulate the change in the metrics, the paths were calculated at time t2. At time t2, for the same source destination pair, the algorithm returned optimal path 1-2-5-8 and the alternate paths 1-4-7-8. The results at time t2 matched the pre-calculated results. All tests similar to these and more proved that the algorithm discussed in Chapter 4 and used in Path Calculation module produced correct result in each and every case.

Table 1 Network topology information for modified Dijkstra algorithm

Link (i-j)	Metrics at time t1(M,N)	Metrics at time t2 (M,N)
1-2	1,1	1,1
1-3	1,1	1,1
1-4	1,1	1,1
2-5	1,2	1,1
3-6	1,1	2,2
4-7	2,2	1,2
5-8	1,1	1,1
6-8	1,1	1,1
7-8	1,1	1,1

The results were obtained for 20 different test networks with large number of nodes and the results were compared with the pre-calculated results. All the tests conducted on different network scenarios with changing link metrics were found to be valid. The results in each test case were same as expected, which was validated using the pre-calculated results. The algorithm was later successfully adopted by the industrial sponsors of this research project in their product offerings.

5.3 Validation of Probing Technique

The Probing module was intended to form an important and integral part of the routing protocol if simulations results were to demonstrate that selective probing could significantly improve the accuracy of the topology information, and, therefore, improved the overall performance of the data networks. As discussed in the Chapter 4, the Data Collection module of the routing protocol, which extracted the link information from the network, updated the link metrics in the in the topology database in a response to the requests from the Probing module. The updated topology database was then used to re-calculate the paths for the destinations registered in the database.

The Probing module for simulation study was implemented in C Language, which was also the case in the modified Dijkstra algorithm that computed the optimal paths based on the cost information. Figure 7 explains the working of the modules in the routing protocol as discussed in Chapter 4. It shows how the Probing module selectively probes a few nodes (e.g. 4 and 5 in Figure 7) in the network and updates the topology information database which is used by the path calculation module to find the optimal paths in the network. Initially, based on the information in the database, the optimal path was 1-4-7-8. A few nodes/links were then probed in the network and the topology database was updated. Based on the updated topology information in the database, the Path Calculation module found the new path; 1-2-5-8, as it determined that the links/nodes used in the previous calculation did not have enough resources at the time of the new route

calculation. This scenario proved the advantage of probing a few nodes in the network and updating the topology information used for route calculations in the network.

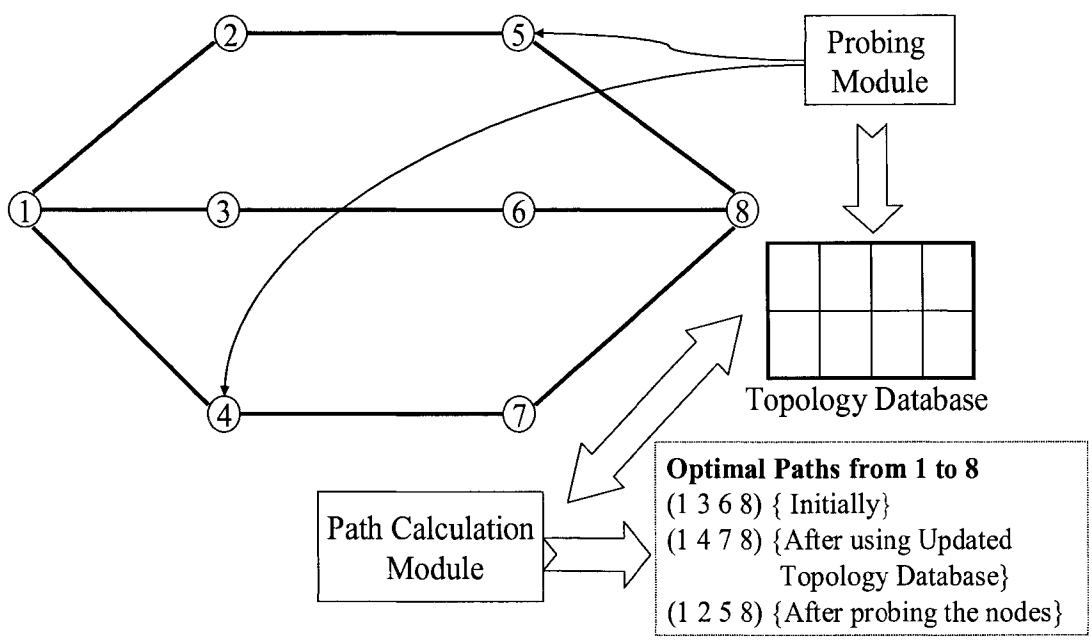

Figure 7 Probing Technique Used with Path Calculation

The NS 2 simulator was used to simulate a wired network with a topology generator model which generates a random traffic pattern in the network. The network model was pure random and the nodes in the network were connected based on the pure random model as discussed in [Ns06] for various simulation studies. The connectivity in our model was decided by the random number generators; the connection was established

between two nodes if the number generated was less than a pre-determined value ranging from 0 to 1. The larger the pre-determined number, the denser the network. Similarly, the traffic streams were established that generated exponential traffic in the network. The streams were randomly distributed in the network and were configurable in the C program written specifically for simulation study of probing, to generate the traffic.

The NS2 simulator did not have the capability to calculate paths based on the different metrics as discussed in Chapter 4. The routes could only be modified manually. To incorporate the constraint based routing decisions in the path calculation module, a special module was written in tcl script. A sub module written AWK was used in the tcl module to read the cost models generated by the functions written in the tcl file. The topology information was flooded on regular intervals and the routes were found based on the topology database after taking into consideration the costs at the regular intervals. This was the mechanism used in link state based routing algorithms like the OSPF, that use information flooded at regular intervals to create the database and use that database to calculate the paths in the network. NS2 was modified to base the path selection on the modified topology database after the regular interval flooding. To evaluate the effectiveness of the selective probing in the network, paths were found at different time intervals and costs of the routes are compared with the ones calculated at the regular intervals.

5.3.1 Description of Simulation Components

A flow diagram of the models involved in the simulation is shown in Figure 8. Components of the flowchart are described in the following sections. The NS2 simulator generates the out.nam file based on the traffic generation model and the connection model. The out.nam file describes the motion of the packets in the network. An AWK program that used the content of the out.nam file was used to calculate the costs based on the Equation (4) described later in this chapter. This cost formula was used for the traffic information. The cost information filled topology files were then used by the simulator for the routing and path calculations. The topology information was used by the modified Dijkstra algorithm described in Chapter 4, to find the routes in the network on a fixed time interval. The route information for different test scenarios was then used to find the accuracy of the selective probing technique for path calculations. The Probing module was used to update the topology information based on the scenarios described later in this chapter.

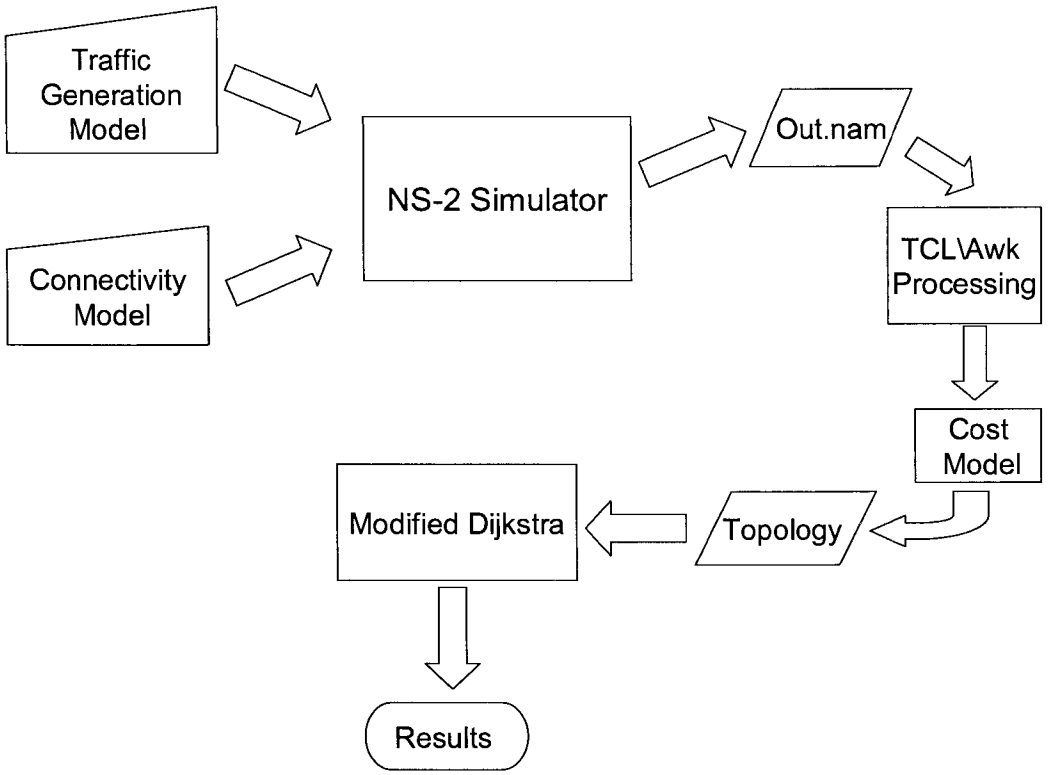

Figure 8 Flowchart of the Simulation Process

5.3.1.1 Connectivity Model

The connectivity model is based on the pure random model discussed in [Ns06]. The model describes a simple probabilistic model to define a connection between any pair of nodes in the network. For implementing this random connection model in the simulation, all the nodes were considered pair wise in the network and the connection was established based on the random number generation method. For each pair in the

network, the system generates a random number and if the generated random number is less than a predefined limiting number the connection is established between the pair of the nodes and otherwise not. This provides a simple and easy to manage connection model. To increase or decrease the density of the network for the simulation case studies the limiting number can be changed to make the network sparse or dense. The larger the limiting number, the denser the connectivity in the network. Also, there was a limit on the maximum number of connections any node in the network could have. This threshold value for the maximum connectivity for the nodes could be changed to make the network dense or sparse in terms of connectivity. Another random number was generated in the same generation model for each link in the network which decided the bandwidth capacity of the links in the network. There were three classes of links with bandwidth 1Mbps, .9Mbps and .7Mbps.

5.3.1.2 Traffic Generation Model

The traffic generation model was written in C language as a separate utility used with the simulator. The traffic generator generated exponential traffic streams in the network. The traffic generation models have been discussed in the traffic generation models for the NS2 simulator used in most simulations in [Mar06]. The traffic generation model took the parameters for the traffic generation as below:

1) Number of nodes
2) Number of connections
3) Packet size

4) Burst time

5) Idle Time

6) Rate

The number of nodes in the network is used to define the actual number of nodes to be simulated in the topology. The number of connections is the total flow streams to be set in the network among the nodes in the topology. This parameter can be used to vary the traffic loads. The packet size denotes the size of the packets sent in the flow streams of the network traffic. The burst time and the idle time are the parameters for the exponential traffic generators for the packet release in the streams. The rate is the average rate of the packets sent in the flow streams of the topology.

5.3.1.3 Cost Model

The cost model considered the reference bandwidth and calculated the cost of the links in the network based on the reference bandwidth and the available bandwidth on the links in the network. The AWK module used the out.nam file generated by the NS2 simulator for the traffic model and generated the topology information by creating the cost file that was used by the simulation to route the traffic in the network. The topology database was then used by the modified Dijkstra algorithm to find the routes in the network and the results for different cases are then compared after the simulation is over. The cost model used the Equation (4), described later in this chapter, to find the cost of the network links in

the topology. The metric was an additive metric as used by the OSPF implementation discussed in the reference [Osp06]. The cost was used in the algorithm discussed before in the thesis to find the optimal paths in the network while minimizing the total cost on the paths.

5.3.2 System Parameters

Table 2 lists the parameters that are either kept constant or varied to generate different simulation scenarios. The table also shows the default values for these parameters used for generating the simulation scenarios.

Table 2 List of Parameters for the Simulation

Parameter Name	**Default value**
Number of Nodes	20
Number of Connections	40
Limit of links per node	4
Percentage of nodes probed	10
Packet size for traffic stream (bytes)	512
Burst time for traffic stream (seconds)	2
Idle time for traffic stream (seconds)	1
Rate for traffic stream (packets/sec)	200
Reference Bandwidth (Mbps)	1
Simulation run time (seconds)	200
Route selection frequency (seconds)	0.5
Topology exchange period (seconds)	4

5.3.3 Simulation Setup and Performance Metrics

5.3.3.1 Simulation Objective

The objective of the simulation study conducted in this research was to evaluate the extent to which the probing technique, which samples the changes in metric information and updates topology of the network, can enhance the efficacy of routing in a data network. Topology database thus updated by Probing module is to provide input information to the metric aggregation module that converts the metrics in each link into a single cost metric. This cost metric is then given as an input to the modified Dijkstra algorithm discussed in Chapter 4 and applied to find the optimal paths in the network.

5.3.3.2 Simulation of Aggregate Cost Metric

In order to concentrate on the ability of the probing technique in improving the accuracy of the topology information in the topology database, the simulation studies presented in this section assume that the Probing module extracts the single cost metric when it probes the network. Although in a practical network the implementers may want the Probing module to extract the parameters individually and populate the database and aggregate their costs separately. Ultimately the aggregate cost is the metric of concern to the Path Calculation module irrespective of the timing and technique of aggregation. The study focuses on the quantitative difference in the accuracy of the output of Path Calculation module in the presence of the Probing module and in the absence of the Probing module.

The control traffic injected for the purpose of probing is not simulated and the communication overhead involved with probing is not studied by this project. Probing is simulated by selectively collecting the topology information from a specially created topology database that contains the metric values.

To simplify the study further, the single aggregate metric used in the simulation study was chosen to be a cost function derived from the available bandwidth on the links in the network as described in [Osp06]. It determined the cost of the links from the reference bandwidth, which was a constant for the simulation, and the available bandwidth on the links. For example, on a link with available bandwidth of .5 Mbps and the reference bandwidth of 1Mbps, the cost metric will be 2 using Equation (4).

Cost = reference bandwidth / available bandwidth .....(4)

5.3.3.3 Simulation Setup

The simulation study was based on a pure random network generated using the connectivity model discussed in section 5.2.1. The default values for parameters for network generation were selected to be as listed in Table 2.

Traffic streams were generated by the traffic generation model using the default values defined in Table 2. An out.nam file was filled by the traffic flow information on the links as the simulation ran. As discussed in section 5.2.2, the additive cost metric to be

optimized by the modified Dijkstra was derived from the available bandwidth information extracted from the out.nam file generated by the NS 2 and using Equation (4). As the traffic generated by the traffic generation model consumed the bandwidth on the links, the available bandwidth was changed and therefore the costs on the links in the network were also changed.

The simulation was run for 200 seconds over the network generated using the connectivity model with the traffic following the traffic generation model. Two time periods, first 40 seconds and the last 60 seconds, were purposely left out as the topology did not change much in these time periods. These were like the filling-up and draining-out time periods for the traffic. On the remaining 100 seconds, a specific percentage of the network was probed every 0.5 seconds. The path calculations were done every 0.5 seconds within this period. The results were then batched into groups of 6 resulting in 30 test cases for each simulation run. The results for real costs at the regular time interval were purposely left out as the three costs would have been same at the regular update time interval.

As the simulation ran, it populated the out.nam file with the topology information. This information was parsed within the tcl script using the AWK program code. The AWK code processed the information and filled in the topology database in a text file which was used for the path calculations by the modified Dijkstra algorithm. The algorithm found the optimal paths in the network based on the topology information in the text file

generated by the AWK code. The results from the algorithm were carefully studied and the quantitative analysis on the improvement in the accuracy of topology information achieved by the use of the Probing module was done.

5.3.4 Description of Experiments

The simulation experiments for this study are outlined below:

- Effect of percentage of probes
- Effect of traffic load in network
- Effect of density of network
- Effect of size of network

The first experiment was to study the effect of percentage of nodes probed in the network. The probe percentage was varied from values of 5, 10 and 20. As the percentage of probing increased the accuracy of the route selection algorithm should have increased. The results would have indicated the right percentage where we do not lose much due to increased communication overhead and also do not lose the accuracy. The second experiment studied the effect of traffic load on the route selection algorithm. The traffic streams can be varied by changing the system parameters in the configuration file. The third experiment was used to study the effect of the density of the network. The density of networks can be changed from the tcl script file by changing the system parameter for the limit on the links per node. The last experiment was conducted to study the effect of

the size of the network. The number of nodes for the simulation can be changed by changing the system parameter that is being set within the tcl script file.

5.3.5 Performance Metrics

The modified Dijsktra algorithm discussed in Chapter 4 was used to calculate the routes for each source destination pair in the network. The topology files generated during the simulation were used as the input for the route selection algorithm. Starting from the start of the effective simulation time, that is 40 seconds, the program calculated the routes in the network for all the source destination pairs at a regular interval of 0.5 seconds. The topology update period was 4 seconds which means that topology was updated with latest and up-to-date information every 4 seconds from the start of the simulation. At a particular time during the simulation, following listed costs for the optimal paths in the network were calculated.

- Regular interval perceived cost
- Regular interval real cost
- Probed perceived cost
- Probed real cost
- Real cost

The first cost is called the regular interval perceived cost. This cost is the sum of all the cost metrics for the paths in the network using the old information at the last update

interval. As the name specifies this cost is the perceived cost as the costs on the links have been changed from the time of the last update. Next, the real costs at the regular time interval are found using the same paths we selected using the old information but replacing all the costs with the real costs at that particular time interval from the topology information text file.

Next, the probed perceived cost is found in the network. Probed perceived cost is the cost for all the paths for selective probing updated topology information. The Real probed cost is the cost of the paths calculated using the selectively probed updated topology but using the real costs on the links from the topology information text file. The Real cost of the paths for the simulation study is calculated using the updated topology at the particular time interval. These paths are the most accurate as we have the latest information available about the topology in the database and it will be the lowest of all the real costs calculated. The selective probing real cost lies between the two real costs, the Real cost at last regular updated topology being the most inaccurate and hence the highest and the latest information based topology being the most accurate and hence the lowest.

The accuracy of selective probing was evaluated based on comparison of the three real costs calculated for each scenario discussed later in this chapter. The Real cost at the time of path calculation would be the lowest, and the one with last regular interval updated topology would be the highest. The Real cost with selective probing would lie between these two values and the closer the value is to the Real cost at the time interval, the more

accurate selective probing is. The idea is to establish that with reduced communication overhead by selectively probing, accurate routes can be found almost as accurate as would have found with fully probing the network (that is the Real cost at the particular time interval).

The 30 test cases described in Section 5.4.2 give us the three real cost values being averaged for the corresponding 6 readings for each test case. Then, the average of the results for all the 30 cases and the standard deviation are found. Next, the formula for finding the confidence of interval for each case study from [Ban01] is used.

95% C.I. = M +/- (1.96 * S.E) .....(5)

S.E = standard deviation / n .....(6)

In the formula above, M is the mean of the distribution of means. The 30 test cases consist of 6 readings each. Average of each test case was taken. Then the average of all the 30 averaged values was found and was used for M in the formula. S.E. denotes the standard error of the distribution. S.E. can be found for the distribution by dividing the standard deviation for the distribution by the number of test cases (30 in our case).

The percentage improvement is used in the simulation results. The formula for the calculation of the percentage improvement is given below.

Actual Improvement = (Regular interval real cost) – (Probed real cost) .....(7)

Ideal Improvement = (Regular interval real cost) – (Real cost) .....(8)

Percentage Improvement = (Actual Improvement / Ideal Improvement) * 100 .....(9)

The performance metrics can now be described for this simulation study as they are based on the three real costs discussed in the previous sections and using the formulas (5) and (6) and (9). Experiments discussed before in this chapter are to be performed to see their effect on simulation study. With increase in percentage of nodes probed in the network, intuitively, the accuracy of the path calculation mechanism should increase. The percentage improvement should be close to the Ideal Improvement at any time of route selection as expected from the claims made by authors in [Eag86]. But the limit of probe percentage is to be found till where the probing percentage can be increased while gaining accuracy without losing the advantage by the increased communication overhead in the network. Similarly, for other variations in the system parameters and the experiments conducted near optimal results for the routes selected are anticipated.

5.3.6 Results and Interpretation

5.3.6.1 Impact of the percentage of probes

In the simulation, first the effect of the percentage of nodes probed in the network for the topology updates was studied. The percentage of nodes probed can be varied by changing variable in the modified Dijkstra algorithm implementation that uses the topology

database updated by the probes in the network. So, with increase in the percentage of nodes probed in the network, intuitively, the accuracy of the path calculation mechanism should increase. But a limit had to be found till where the probing percentage can be increased and gain in accuracy is still achieved without losing the advantage by the increased communication overhead in the network. The results for average link real costs are listed in Table 3.

Table 3 Average costs comparison to verify the effect of probe percentage

	5% Probe	**10% Probe**	**20% Probe**
Regular interval real cost	4.8862	4.8862	4.8862
Probed real cost	4.8676	4.8402	4.7951
Actual real cost	3.5295	3.5295	3.5295

The results from the simulation are plotted in Figure 9 as the average of the percentage gain in accuracy of the algorithm in terms of total cost of all the optimal paths for each source destination pair in network. The averages for Confidence Interval of 95% are also plotted on the plot shown in Figure 9. As evident from the plot, slight improvement, but not significant, was achieved with probing even up to 20% nodes in the network. However, nearly linear increase in the accuracy of the path calculations was observed with increase in the percentage of nodes probed in the network. From Figure 9 and the

interpretation of the results of the simulation, it can be safely said that probing did not help with the accuracy of the path calculation as expected. The effect of other parameters on the accuracy of probing are studied in following sections to see if it might help in some other scenario.

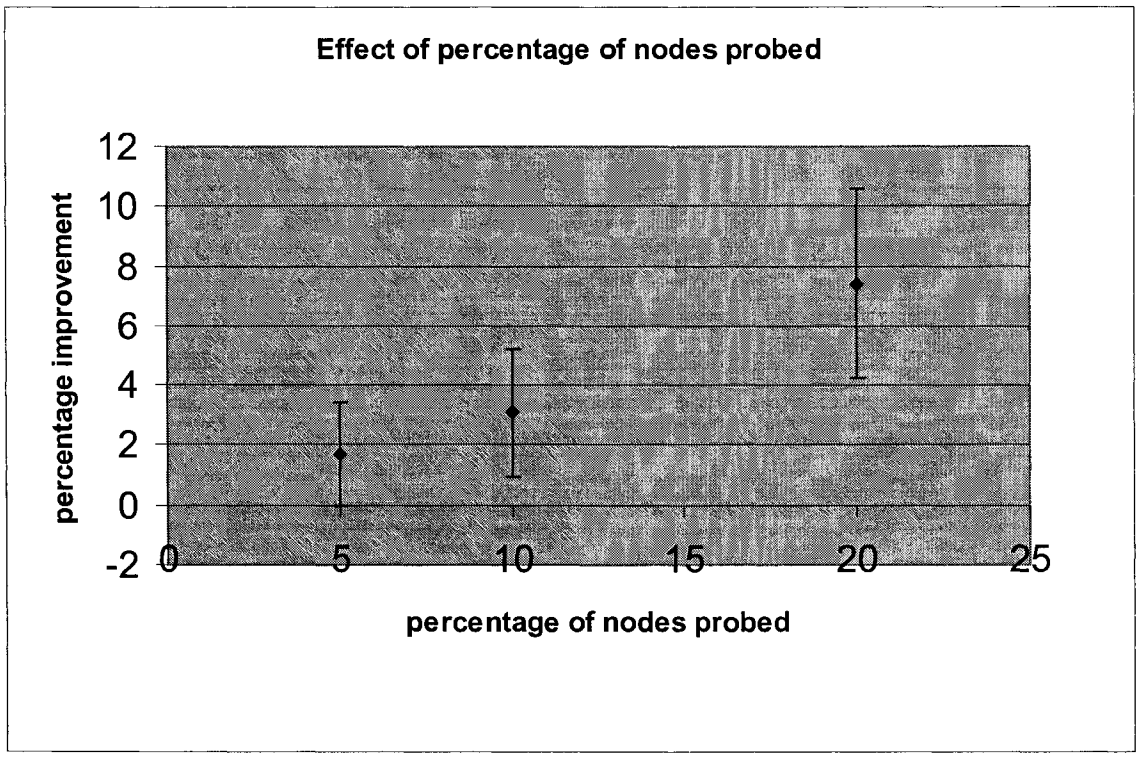

Figure 9 Results for Simulation to verify effect of percentage of nodes probed

5.3.6.2 Impact of traffic load in the network

As discussed in the previous sub-section 5.3.6.1, the probing did not help much with the accuracy of the optimal paths calculated by the routing algorithm with probing enabled. So, the effect of the traffic load on the accuracy improvement of path calculation mechanism is studied in this section. The input parameters can be changed for the traffic

generator to see the effect of the traffic load on the accuracy. The traffic in network was reduced by reducing the number of traffic streams in the network from 40 from 20. Next, the number of streams was increased to 80 to simulate the highly loaded network. Also, results were obtained for the real costs in the network for the normal load using number of streams to be 40. The results for average link real costs are listed in Table 4.

Table 4 Average costs comparison to verify the effect of traffic load

	Low	**Normal**	**Heavy**
Regular interval real cost	1.8973	4.8862	33.0728
Probed real cost	1.8991	4.8402	33.7847
Actual real cost	1.8954	3.5295	9.1237

The plot in Figure 10 shows the average of all the three cases with 95% Confidence Interval. Similar to the previous section discussions, not much improvement was observed in any scenario by changing the network loads. From the plot in Figure 10, it is evident that increasing the number of traffic streams does not change the performance improvement among the three cases. This proves that the traffic load changes do not change the efficiency of the path calculation mechanism with probing in the network. In the next section, the effect of connectivity of the nodes in the network is studied.

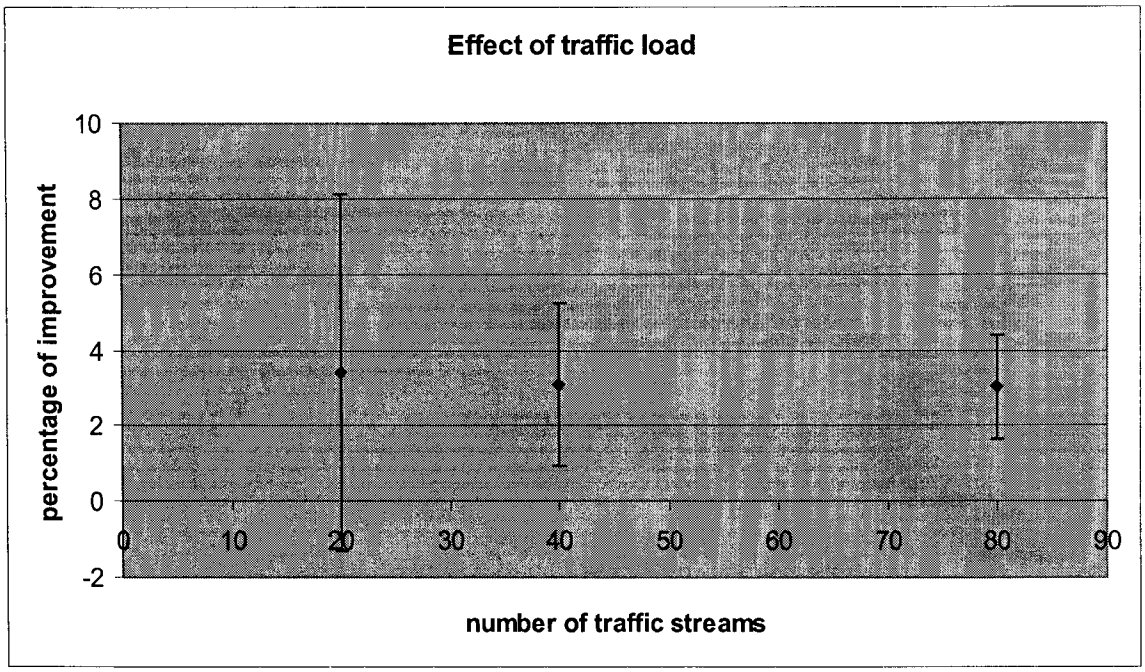

Figure 10 Results for Simulation to verify effect of traffic load in network

5.3.6.3 Impact of the density of the network

From the results and the discussions in the two previous sections, it is evident that probing till now has not proven to be as helpful as expected in either case. So, in this section the effect of change in the density of the network is studied. This experiment was to see if the density of the network improves the efficiency of probing in the path calculation mechanism for the networks. Three scenarios were simulated by changing the connectivity of the nodes in the network. In the .tcl file for the simulation the threshold value for the maximum allowed number of connections per node was changed to change the density of the network. For simulating a sparse network scenario the threshold value was reduced to 2. Next in the experiment the threshold value was increased to 6 to

simulate a dense network. For the average network, the threshold value was set to be 4. The results for average link real costs are listed in Table 5.

Table 5 Average costs comparison to verify effect of density of network

	Sparse	**Normal**	**Dense**
Regular interval real cost	10.4603	4.8862	5.3354
Probed real cost	10.4950	4.8402	5.4196
Actual real cost	5.4389	3.5295	3.3158

Figure 11 shows the results for the averages of all three cases with 95% C.I. Quite evident from Figure 11 is the fact that this variation as the previous ones could not establish the usefulness of probing in the path calculation mechanisms. However, in the sparse network the probing was least effective as it had lesser options for the paths and hence less variation. However, when the connectivity was increased to 6, not much difference was found in the improvement as compared to the normal connectivity network. The effect of the number of nodes in the accuracy of the probing technique for path calculations in the network is studied in the next section.

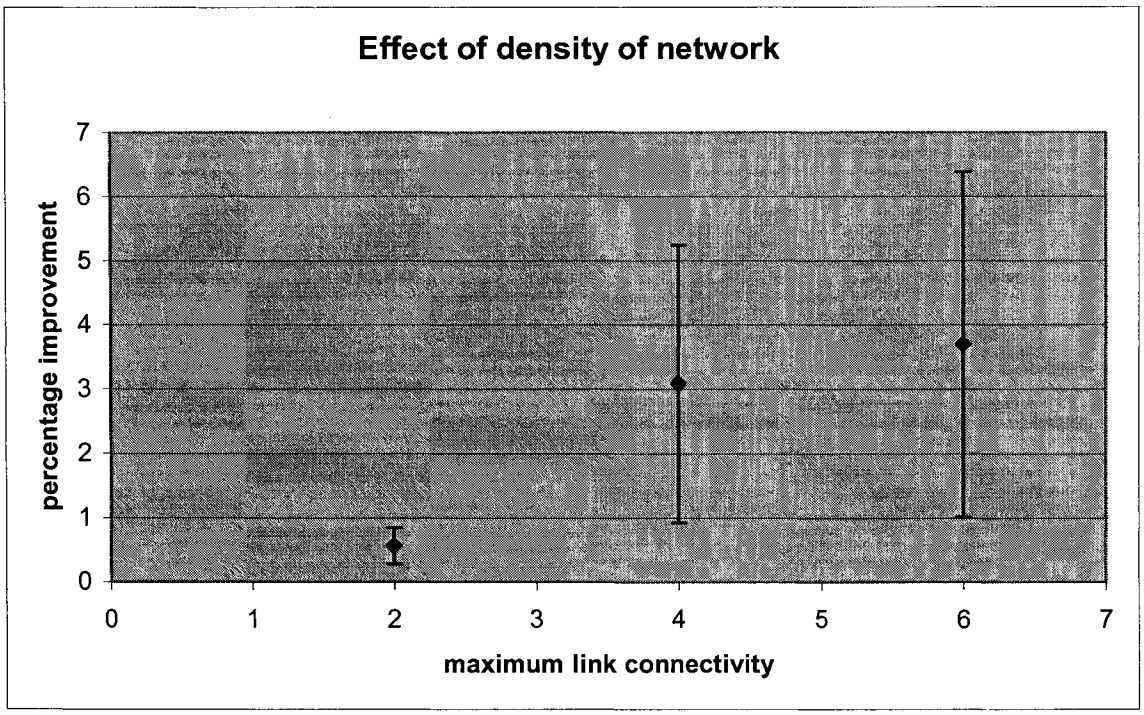

Figure 11 Results for Simulation to verify effect of density of network

5.3.6.4 Impact of the size of the network

In this section the effect of size of the network is studied. The size of the network was changed by changing the number of nodes in the simulated network. The other parameters for the network like the link connectivity are unchanged and the traffic load was changed so as to make it proportional to an average case. For the study of 50 nodes, for example, number of streams in the network was changed to 100 as compared to 40 for 20 nodes. The number of nodes in the network can be changed from the .tcl file used for the simulations. The results for average link real costs are listed in Table 6.

Table 6 Average costs comparison to verify effect of network size

	50 Nodes	**20 Nodes**
Regular interval real cost	10.2624	4.8862
Probed real cost	10.3298	4.8402
Actual real cost	8.6927	3.5295

As shown in Figure 12, a 50-node network was used for simulation study and the results are plotted along with the average sized network of 20 nodes. The performance improvements were almost negligible, even with more than 100% increase in the size of the network. So, increasing the size of the network further would be not an effective to establish the accuracy of probing mechanism. As discussed in the previous sections, and also in this section, probing is not helpful in improving accuracy of the path calculation mechanism significantly in any case with any variation of the network parameters.

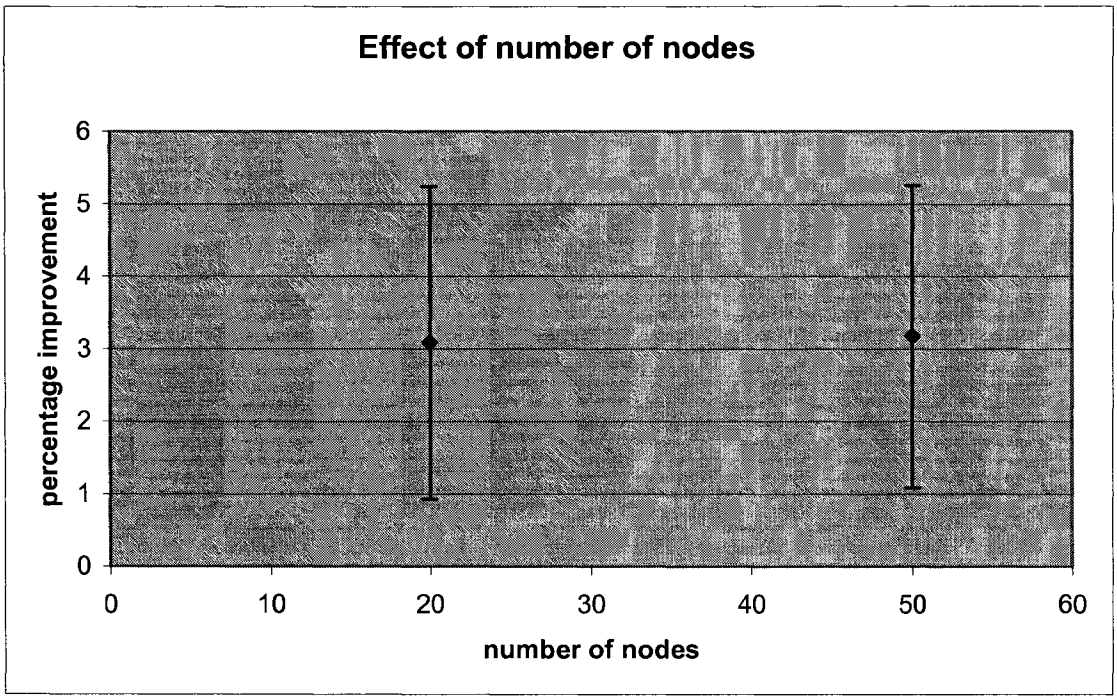

Figure 12 Results for Simulation to verify effect of size of the network

Chapter 6

Conclusions

6.1 Summary

As discussed in Chapter 2 of thesis, out of the available proposals for solving the multi-constrained routing problem in the network, design of a proposed solution is discussed in the Chapter 4 of this thesis. To increase the accuracy of the routing protocols, the selective routing technique is proposed. Out of the two types of routing protocols described in Chapter 2, one type of routing protocol where we aggregate the metrics and use a single metric to find the optimal routes in the network, is discussed in Chapter 4. This technique has less computation overhead as compared to the technique where each constraint is handled individually. However, on the down side, with this technique the constraints lose the individuality and we cannot specifically limit the resources in the network as could be done with the second technique.

From the results in Chapter 5 for the simulation for the probing technique efficiency evaluation, it can be safely said that the simulation has proven the probing only provides slight performance gains for some scenarios and no improvement for others. The performance gains were much lower than expected, as other researchers in [Eag86] have reported significant increase in performance with about 10% processors being probed in the distributed systems. The simulation results in Chapter 5 discuss results for 10% node

probing in the network, which showed slight performance gains. The main difference between their results and ours is likely due to the fact that our problem is dealing with a path which consists of multiple links rather than just a single resource.

The selective probing does not help much in increasing the accuracy of the path calculation mechanism. Even with increase in the probing percentage to 20% of the nodes in the network, not much significant or considerable improvement in the accuracy of the route selection algorithm was seen. Increasing the probe percentage beyond this value would increase the communication overhead in the network and diminish the effect of the accuracy improvements for the results. Probing was initially used in the case of multi-processors in distributed systems where we probe the available processors to find the most lightly loaded processor and send the load to it.

Also, different parameters were varied in the network to see their effect on probing, none of which helped to validate the performance improvement claims made earlier in thesis for probing in the path calculation mechanism. The four parameters were changed and the effects being discussed in the previous sections. What was observed from the results and the plots was that changing the network size would not affect the route finding mechanism's accuracy at all. Changing the percentage of nodes probed almost linearly changed the accuracy but still was not significant. Similarly, for the traffic load and the connectivity of the nodes in the network showed a bit of influence on the performance improvement but again, not much improvement was observed. As discussed in Chapter 4,

the Probing module would have been incorporated in the actual implementation of the routing mechanism if selective probing had established the accuracy improvement claims. The selective probing would have been very helpful in networks with limited resources, like sensor networks by providing up-to-date topology information for route selection without using much the limited resources available on the network.

6.2 Contributions and Future Work

Chapter 2 of this thesis discusses what the researchers have proposed for solving the multi-constrained routing problem. The multi constraint routing problem is NP Hard problem which cannot be solved in polynomial time and there are many heuristic based solutions proposed by the researchers. Two types of solutions to solve these problems have been discussed in the Chapter 2, out of which one design solution for one type is presented. The solution implements the type of protocols that convert the multi constraint problem to a single additive constraint problem and then use the extended Dijkstra algorithm to find the optimal paths in the network.

The essence of this thesis comes from the Probing module of the proposed routing protocol that probes the nodes in the network to get the updated information about the resources in the network on which the routing decisions are based. This module will form an integral part of the routing protocol and will be very helpful in reducing the communication overhead while increasing the efficiency of the routing mechanism, if

found to be effective. This technique has been implemented in the distributed systems where the processors are selectively probed to find the lightly loaded nodes in the system. However, simulations study has evaluated the technique in the routing protocols for the networks and this technique is not found to be efficient in the routing protocols for the networks.

Till now, the type of networks referenced are the wired networks, however, in ad-hoc networks where nodes have limited resources particularly the sensor networks, the thesis would have found its real deployment if proved to be efficient. Advances in wireless network technologies have created tremendous opportunities in personal area networks and wireless local area networks. An intelligent wireless network can provide access to information anytime and anywhere by collecting, processing, analyzing and distributing data. Such a network can be used by a variety of applications, such as military, detecting environmental hazards (sensor networks), monitoring remote terrains, collecting network statistics (network usage per customer for billing purposes), medical applications, etc. In this thesis some solutions for the multi constrained routing problem with selective probing technique are discussed to increase the accuracy of the routes while minimizing the associated communication overhead. The selective probing technique was not found to be efficient enough and hence is not recommended to be used in the routing protocols. So, in order to select the routes with latest and accurate information in the network while not increasing the communication overhead considerably, specially in the low resourced

sensor networks, more research should be done to find some other technique to realize this goal.

References

[Apo98] G. Apostolopoulos et al., "QoS Routing Mechanisms and OSPF Extensions," RFC 2676, August 1998, http://www.ietf.org/rfc.html, Last accessed on 10 June 2006.

[Awd01] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC 3209, December 2001, http://www.ietf.org/rfc.html, Last accessed on 05 November 2005.

[Ban01] J. Banks, J. S. Carson, B. L. Nelson and D. M. Nicol, *Discrete Event System Simulation*, 3rd edition, Prentice-Hall, Inc., Upper Saddle River, New Jersey 07548, U.S.A, 2001

[Bar92] M. Barezzani, E. Pedrinelli and M. Gerla, "Protection Planning in Transmission Networks," *Proc. of Int'l Conf on Communications*, 1992, pp. 420–424.

[Bej03] Y. Bejerano, Y. Breitbart, A. Orda, R. Rastogi and A. Sprintson, "Algorithms for Computing QoS Paths with Restoration," *Proc. of INFOCOM*, April 2003.

[Che98a] S. Chen and K. Nahrstedt, "An overview of Quality of Service Routing for Next –Generation High-Speed Networks: Problems and Solutions," *IEEE Network*, November/December 1998, Vol. 12, No. 6, pp. 64–79.

[Che98b] S. Chen and K. Nahrstedt, "On Finding Multi-Constrained Paths," *Proc. of IEEE International Conference on Communications*, Jun. 1998, pp. 874-879.

[Che98c] S. Chen and K. Nahrstedt, "Distributed QoS Routing with Imprecise State Information," *Proc. of IEEE International Conference on Computer, Communications and Networks,* October 1998, pp. 614–621.

[Che99] S. Chen and K. Nahrstedt, "Routing by Distributed Recursive Computation and Information Reuse," *Proc. of 18th IEEE International Performance, Computing and Communications Conference (IPCCC)*, 1999, pp. 393–403.

[Cor02] T.H. Cormen, C. E. Leiserson, R.L. Rivest and C. Stein, *Introduction to Algorithms*, MIT Press/McGraw-Hill Book Company, 2002.

[Cui03] Y. Cui, K. Xu and J. Wu, "Precomputation for Multi-Constrained QoS Routing in High-Speed Networks," *Proc. of IEEE INFOCOM 2003,* March 2003, pp. 1414-1424.

[Dai02] J. Dai, H. Pung and T. Angchuan, "A Multicast routing protocol supporting multiple QoS constraints," *Proc. of 10th International Conference on Networks*, 2002, pp. 31–36.

[Eag86] D.L. Eager, E.D. Lazowska and J. Zahorjan, ''Adaptive load sharing in homogeneous distributed systems,'' *IEEE Transactions on Software Eng*ineering, 1986, Vol. 12, pp. 662 – 675.

[Fan99] Z. Fan and E.S. Lee, "Multiple QoS Constrained Routing with Inaccurate State Information," *Electronics Letters*, 1999, Vol. 35, No. 21, pp. 1807-1808.

[Goy99] P. Goyal and G. Hjalmtyssony, "QoS Routing for Best-effort Flows," *Proc. of 8th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 98)*, 1998.

[Gra05] E. Grampin and J. Serrat, "Cooperation of Control and Management Plane for Provisioning in MPLS Networks," *Proc. of 9th IEEE International Symposium on Integrated Network Management,* 2005, pp 281-294.

[Guo99] L. Guo and I. Matta, "Search Space Reduction in QoS Routing," *Proc. of IEEE International Conference on Distributed Computing Systems,* 1999, pp. 142–149.

[Hel70] M. Held and R. Karp, 'The traveling salesman problem and minimum spanning trees', *Operations Research 18*, 1970, pp.1138-1162.

[Jia04] Y. Jia, I. Nikolaidis and P. Gburzynski, "Scalable QoS Routing Using Alternative Paths," *International Journal of Communication Systems*, 2004, Vol. 17, No. 1, pp. 1–26.

[Jut01] A. Juttner, B. Szviatovszki, I. Mecs, and Z. Rajko, "Lagrange Relaxation Based Method for the QoS Routing Problem," *Proc. of IEEE INFOCOM*, 2001, pp. 782–791.

[Kar98] O. Kartau, "Policy-Based Routing," Available from http://www.netlab.hut.fi/opetus/s38130/s98/routers/policy-routing.html, Last accessed on 16 December 2005.

[Kor99] T. Korkmaz and M. Krunz, "A Randomized Algorithm for Finding a Path Subject to Multiple QoS Constraints," *IEEE GLOBECOM '99*, pp. 1694–1698.

[Kor01] T. Korkmaz and M. Krunz, "Multi-Constrained Optimal Path Selection," *IEEE INFOCOM 2001*, Vol. 2, pp. 834–843.

[Kui03] F.A. Kuipers and P. Van Mieghem, "The Impact of Correlated Link Weights on QoS Routing," *IEEE INFOCOM 2003*, Vol. 2, pp. 1425-1434.

[Kwe99] S.-K. Kweon and K. G. Shin, "Distributed QoS Routing Using Bounded Flooding," University of Michigan, Ann Arbor, CSE Tech. Rep. CSETR-388-99, 1999.

[Lee03] S. S. Lee, S. Das, G. Pau, and M. Gerla, "A Hierarchical Multipath Approach to QoS Routing: Performance and Cost Evaluation," *ICC 2003*, Vol. 1, pp. 625-630.

[Liu01] G. Liu, K. G. Ramakrishnan, "A*prune: An Algorithm for Finding K Shortest Paths Subject to Multiple Constraints," *Proc. of IEEE INFOCOM*, 2001, pp. 743–749.

[Liu06] H. Liu, M. Zheng and D. Bin, QoS Routing, http://www.ee.vt.edu/~l**dasilva**/6504/Routing.pdf, Last accessed on 06 June 2006.

[Lui06] K. S. Lui, "QoS Routing," http://cairo.cs.uiuc.edu/qosrouting/qos-routing.html, Last accessed on 12 January 2006.

[Ma97] Q. Ma and P. Steenkiste, "Quality-of-Service Routing for Traffic with Performance Guarantees," *IFIP Fifth International Workshop on Quality of Service*, May 1997, pp. 115-126.

[Mal94] G. Malkin, "RIP Version 2," RFC 1723, November 1994, http://www.ietf.org/rfc.html, Last accessed on 12 June 06.

[Mal04] B. Malakooti, M. Griffith, T. Linsky, V. Liberatore and N. Malakooti, "Multi-Criteria Space Based Internet Routing," *Proc. of the Industrial Engineering Research Conference (IERC),* May 2004.

[Mar06] Marc Greis' Tutorial for the UCB/LBNL/VINT Network Simulator "ns", http://www.isi.edu/nsnam/ns/tutorial/index.html, Last accessed on 21 July 2006.

[Moy98a] J. Moy, "OSPF Version 2," RFC 2328, April 1998, http://www.ietf.org/rfc.html, Last accessed on 12 June 2006.

[Moy98b] J. T. Moy, *OSPF Anatomy of An Internet Routing Protocol*, Addison-Wesley, 1998.

[Mun98] R. Muntz, J.R. Santos and S. Berson, "A Parallel Disk Storage System for Real-Time Multimedia Applications," *International Journal for Intelligent Systems,* 1998, Vol. 13, pp. 1137-1174.

[Nev98] H. De Neve and P. Van Mieghem, "A Multiple Quality of Service Routing Algorithm for PNNI," *Proc. of the IEEE ATM Workshop*, May 1998, pp. 324–328.

[Nor02] S. Norden and J. Turner, "Inter-Domain QoS routing Algorithms," Applied Research Lab, Department of Computer Science, Technical Report WUCS-02-03, Washington University, Saint Louis, 2002.

[Ns06] The Network Simulator ns-2: Topology Generation, http://www.isi.edu/nsnam/ns/ns-topogen.html#gt-itm, Last accessed on 20 July 2006.

[Osp06] OSPF Frequently Asked Questions [IP Routing], http://www.cisco.com/en/US/tech/tk365/technologies_q_and_a_item09186a0080094704.shtml#q3, Last accessed on 09 August 2006.

[Pun99] H. K. Pung, J. Song and L. Jacob, "Fast and Efficient Flooding Based QoS Routing Algorithm," *Proc. Of International Conference on Computer Communications and Networks* (ICCCN), October 1999, pp. 298–303.

[Rab00] R. Rabbat, K. P. Laberteaux, N. Modi and J. Kenney, "Traffic Engineering Algorithms Using MPLS for Service Differentiation," *Proc. of IEEE International Conference on Communications (ICC),* June 2000, pp. 791–795.

[Shi02] D. Shin, E. K. P. Chong, and H. J. Siegel, "A Multiconstraint QoS Routing Scheme Using a Modified Dijkstra Algorithm," Networks: *Proceedings of the Joint*

International Conference on Wireless LANs and Home Networks (ICWLHN) and Networking (ICN), August 2002, pp. 65–76.

[Son00] J. Song, H. K. Pung and L. Jacob, "A Multi-Constrained Distributed QoS Routing Algorithm," *Proc. of IEEE International Conference on Networks (ICON),* 2000, IEEE Computer Society, pp. 165-171.

[Sta04] William Stallings, *Data and Computer Communications,* Pearson Prentice Hall, 2004.

[Xia02a] L. Xiao, J. Wang and K. Nahrstedt, "The Enhanced Ticket-based Routing Algorithm," *Proc. IEEE International Conference on Communications (ICC)*, 2002, pp. 2222–2226.

[Xia02b] W. Xiao, Y. Luo, B. H. Soong, K. V. Ling, H. C. Chua and C. L. Law, "An Efficient Heuristic Algorithm for Multi-constrained Path Problems," *Proc. of IEEE VTC (Vehicular Technology Conference)* , 2002, pp. 1317–1321.

[You03] O. Younis and S. Fahmy, "Constraint-Based Routing in the Internet: Basic Principles and Recent Research," *IEEE Communications Surveys & Tutorials*, 2003, pp. 2–13.

[Zap97] D. Zappala, D. Estrin, and S. Shenker,"Alternate Path Routing and Pinning for Interdomain Multicast Routing," Technical Report USC CS TR 97-655, University of South California, 1997.

[Zap04] D. Zappala, "Alternate Path Routing for Multicast," *IEEE/ACM Transactions on Networking,* February 2004, Vol. 12, pp. 30-43.

Appendix

1. Path Calculation use case

The Path Calculation module use cases are shown in the Figure 13. The use cases are described in detail in the following section.

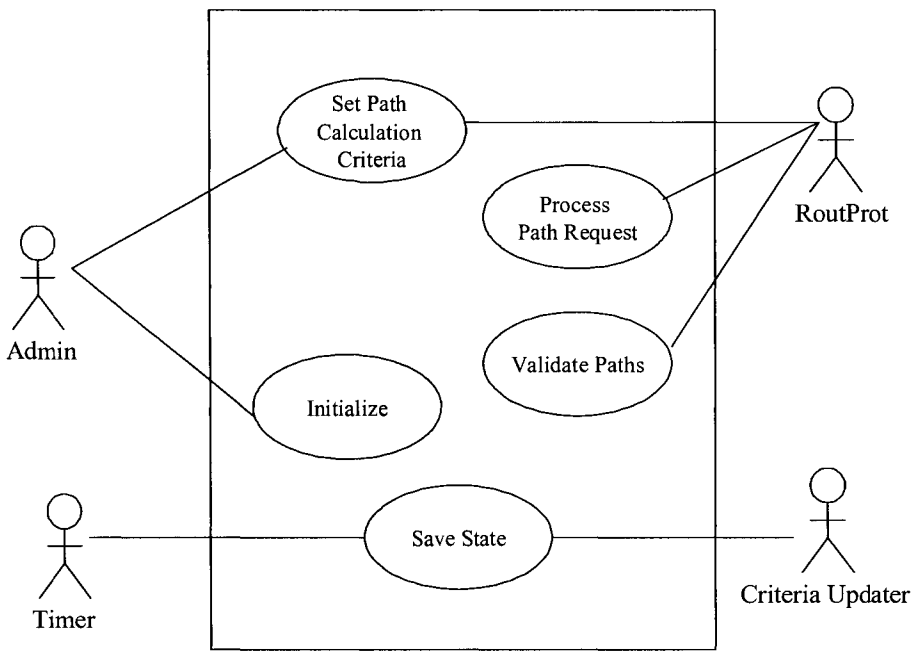

Figure 13 Use Case Diagram for PathCalc

UC1: Initialize

Summary

The Administrator (ADMIN) initializes the port ids for PathCalc to communicate with other modules.

Actor

ADMIN

Preconditions

None

Description

Administrator sets the port ids for the PathCalc to communicate with other modules like RoutProt (Routing Protocol) and DataColl (Data Collection).

UC2: Set Path Calculation Criteria

Summary

The Administrator (ADMIN) sets the criteria for optimal path selection.

Actor

ADMIN or RoutProt

Preconditions

None

Description

1. Administrator or RoutProt reads a set of constraints from the available options to be considered for the optimal path calculations.

2. Save the constraints to be used by the Path Calculation module.

3. Ensure that the constraints entered by ADMIN or RoutProt provide a valid combination.

UC3: Process Path Request

Summary

The PathCalc module processes the path request sent by RoutProt or entered by the administrator and sends back the path. The path request contains the source, destination and constraint values.

Actor

RoutProt

Preconditions

The port should be initialized for communication with PathCalc.

The path selection criteria should be known to the PathCalc module (i.e. must be fed in the configuration file).

Description

1. RoutProt sends a route a request message to the PathCalc module with the source address, destination address, and constraints as parameters using the port id set up by the administrator.

2. PathCalc uses the Topology Database and the parameters sent in the message to find the optimal path for the source destination pair satisfying the constraints.

3. PathCalc sends back the optimal path to the RoutProt if one is found; otherwise, it sends back a message indicating that no path was found.

UC4: Validate Paths

Summary

PathCalc module validates the list of paths sent by the RoutProt module and returns the paths that passed the validation for the resources.

Actor

RoutProt

Preconditions

None

Description

1. PathCalc module receives a set of paths sent by the RoutProt to be validated.

2. PathCalc module validates the paths using the TopoDb.

3. PathCalc sends back the set of paths that had enough resources to the RoutProt module.

UC5: Save State

Summary

PathCalc module saves the last known state of the database in a configuration file and reads from it when it loses the information due to some failure.

Actor

Timer and Criteria Updater

Preconditions

None.

Description

Path Calculation module saves the current snapshot of the TopoDb when triggered to do so by the Timer or the Criteria Updater.

Static Model

The following diagram depicts the static model of the Path Calculation module. It shows how different modules interact with PathCalc and the relationships between the modules.

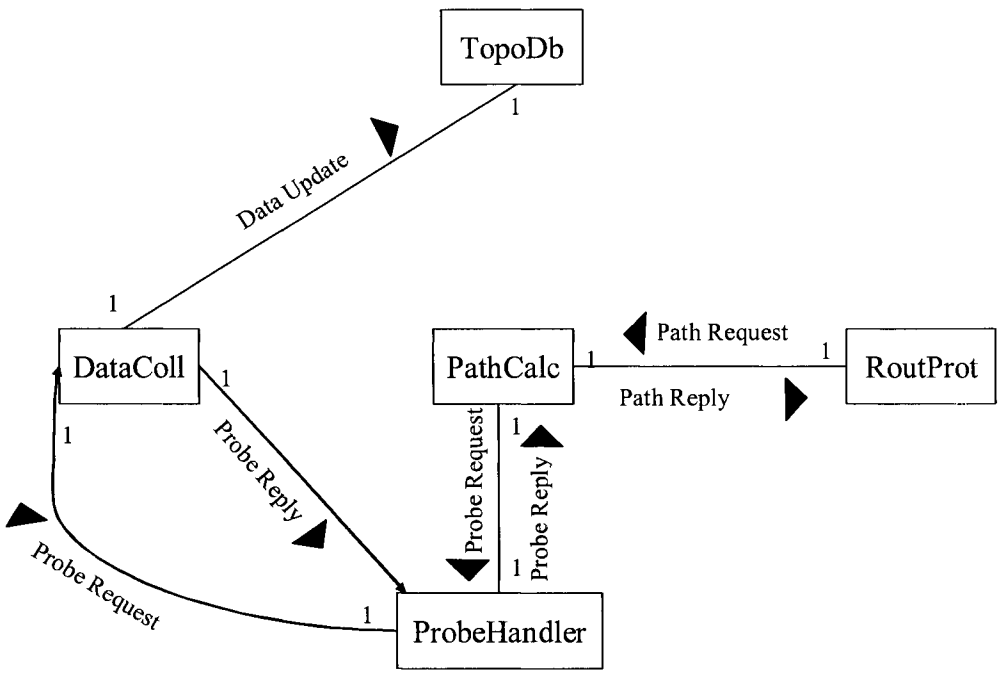

Figure 14 PathCalc Static Model

Message Sequence

The following two figures (Figure 15 and Figure 16) show the message sequence charts of two use cases, UC3 and UC4, of the PathCalc module.

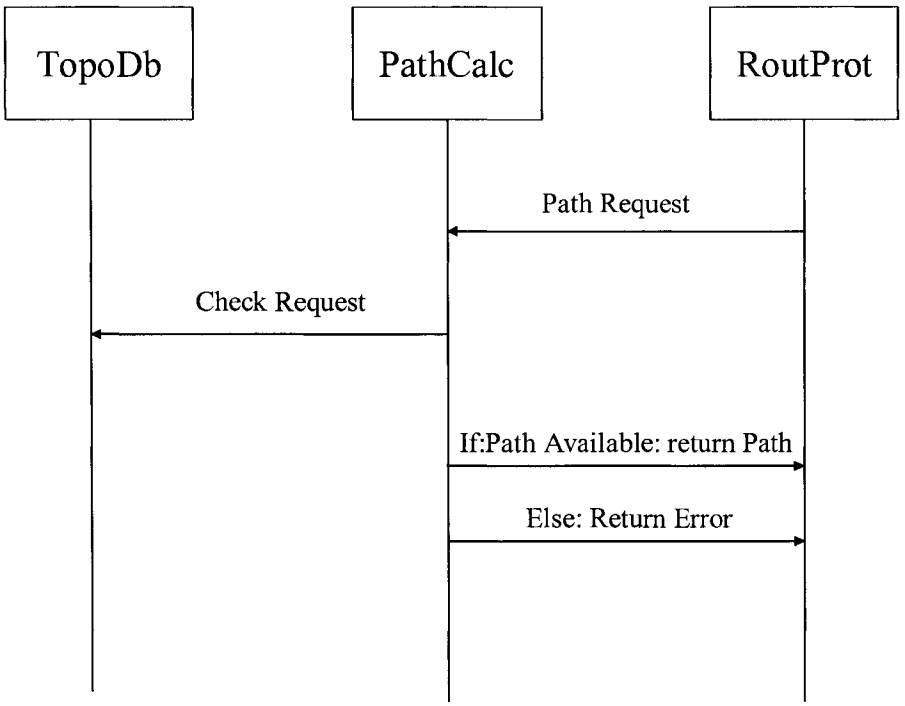

Figure 15 Message Sequence of UC3: Process Path Request

In the message sequence diagram of the Process Path Request use case shown above, the RoutProt sends a path request to the PathCalc module. The PathCalc module checks for the resources required by the request in the TopoDb. If the resources are available for path between source and destination satisfying the constraints in the request, the PathCalc returns the found path; otherwise, it sends an error message.

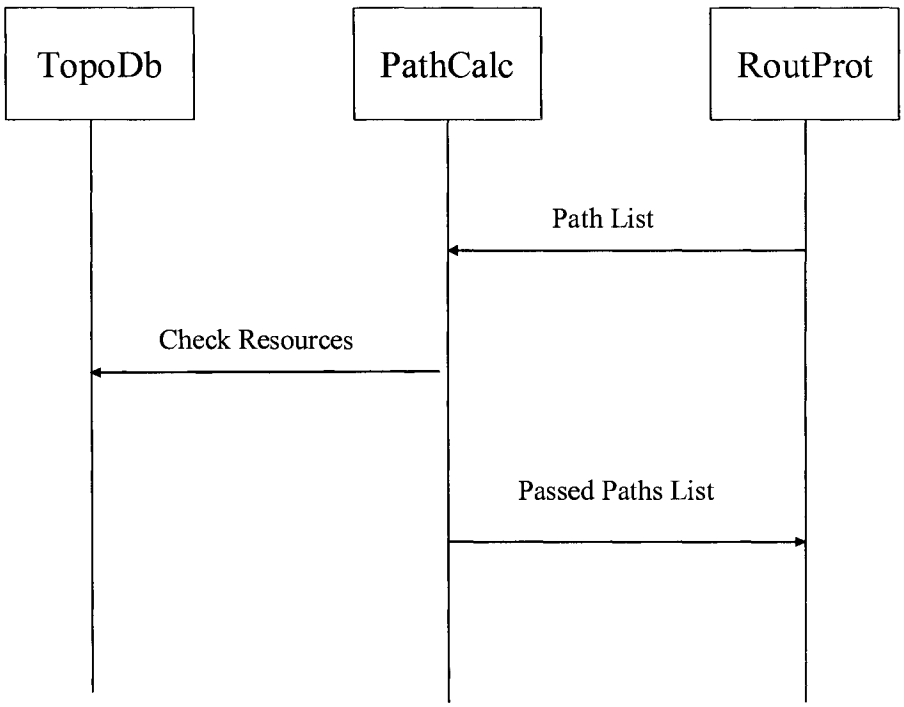

Figure 16 Message Sequence of UC4: Validate Paths

In Figure 16, the RoutProt sends a list of paths to be verified to the PathCalc module. The PathCalc module, on receiving the list of paths, checks for the resources from the TopoDb entries. It selects the passed paths and sorts them according to the resource availability on each and sends back the list to the RoutProt module.

State Transition

Figure 17 shows the state chart of the Path Calculation module. The module reads the configuration file to get into the Ready state from the Init state and receives the path request from the RoutProt to enter the Probing state where it sends the probe request to the DataColl module before finding the optimal path. After sending the probe requests the module enters the Waiting state while waiting for the response form the DataColl. It may have alternate techniques for path calculations and if the results for all techniques do not match then the path is calculated again. After processing the path requests by calculating the optimal paths, the module communicates the result to the Routing Protocol and enters the Ready state.

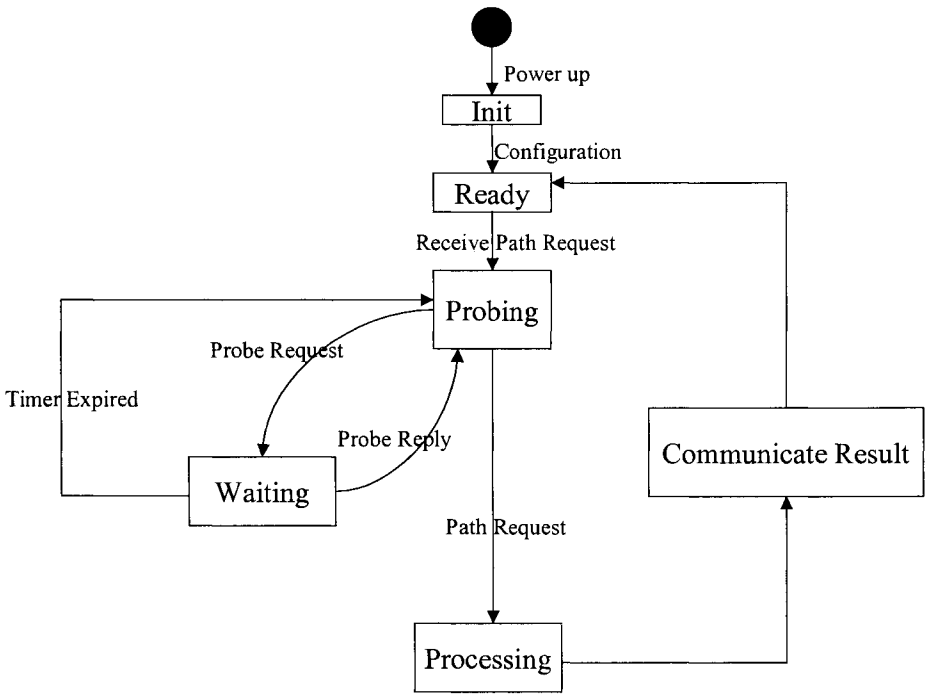

Figure 17 State Transition chart of PathCalc

2. Topology Database use cases

The Topology Database module has six use cases which are shown in Figure 18. The use cases are described in detail in the following section.

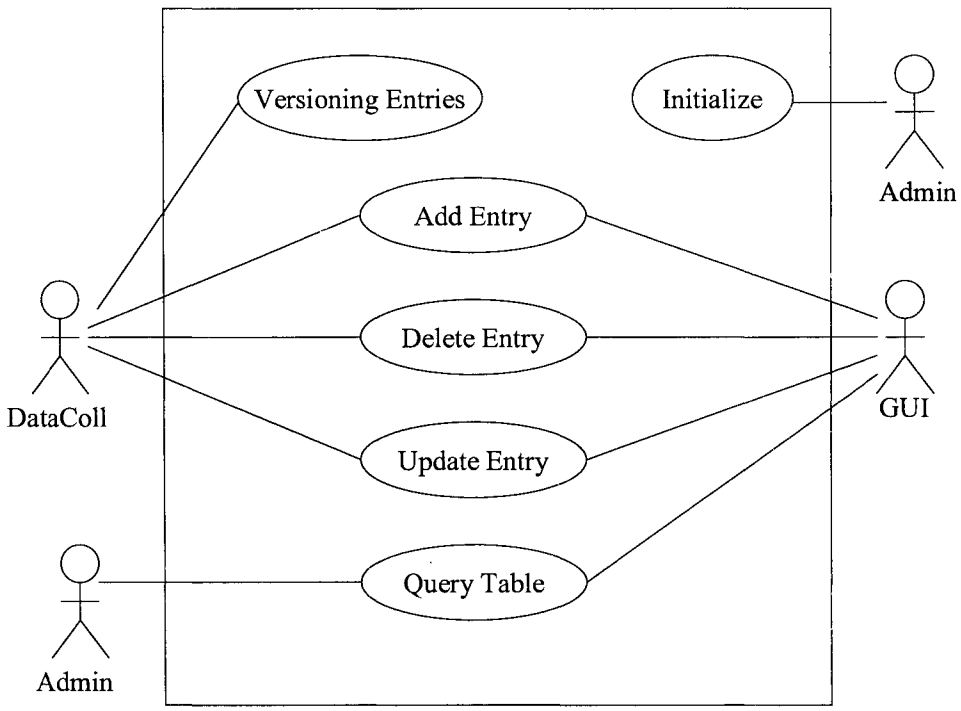

Figure 18 Use Case Diagram for TopoDB

UC1: Initialize

Summary

The Administrator (ADMIN) initializes the port ids that other modules to communicate with TopoDb (Topology Database). It also initializes the storage structure to store the database entries in the Topology Database.

Actor

ADMIN

Preconditions

None

Description

1. The Administrator sets the port ids for the other modules to communicate with TopoDb.

2. The Administrator initializes the storage structure for the topology information storage for the network.

UC2: Add Entries

Summary

The Data Collection (DataColl) or GUI updates the topology information stored in the TopoDb by adding the entries in the table.

Actor

GUI or DataColl

Preconditions

The port ids have been set for the communication with the database and the database storage structure has been initialized.

Description

1. The GUI can be used to add entries in the database.

2. The DataColl module adds new entries in the database.

UC3: Delete Entries

Summary

The Data Collection or GUI updates the topology information stored in the TopoDb by deleting the entries in the table.

Actor

GUI or DataColl

Preconditions

The port ids have been set for the communication with the database and the database storage structure has been initialized.

Description

1. The GUI can be used to delete the entries in the database.

2. The DataColl module deletes entries in the database.

UC4: Update Entries

Summary

The Data Collection or GUI updates the topology information stored in the TopoDb by updating the entries in the table.

Actor

GUI or DataColl

Preconditions

The port ids have been set for the communication with the database and the database storage structure has been initialized.

Description

1. The GUI can be used to update the entries in the database.

2. The DataColl module updates the existing entries in the database.

UC5: Query Table

Summary

The Admin or GUI queries the table to get topology information from the TopoDb.

Actor

GUI or Admin

Preconditions

The port ids have been set for the communication with the database and the database storage structure has been initialized.

Description

The GUI or Admin can query the TopoDb to get the topology information.

The following options are available to get the information displayed from the TopoDb:

a) All Data: all the fields of the database entries are to be displayed.

b) Subset of data: specific fields of the entries are to be displayed

c) One entry: just one entry is to be displayed

d) Many entries: more than one entry is to be displayed

e) All entries: all the entries in database are to be displayed

UC6: Versioning of entries

Summary

The Data Collection module is responsible for versioning the entries in the TopoDb.

Actor

DataColl

Preconditions

The port ids have been set for the communication with the database and the database storage structure has been initialized.

Description

1. The DataColl module switches the entries to be in Up or Down state. When the nodes are inaccessible the state is changed to Down (initially all the entries are Up).

2. The existing entries are used in path calculations only if the state is UP. The Down state entries are kept in the database in order to reduce the overhead of updating the information as the states change from Down to Up again.

Static Model

Figure 19 shows the static model of the Topology Database. The interactions of other modules with the Topology Database along with their relationships are depicted.

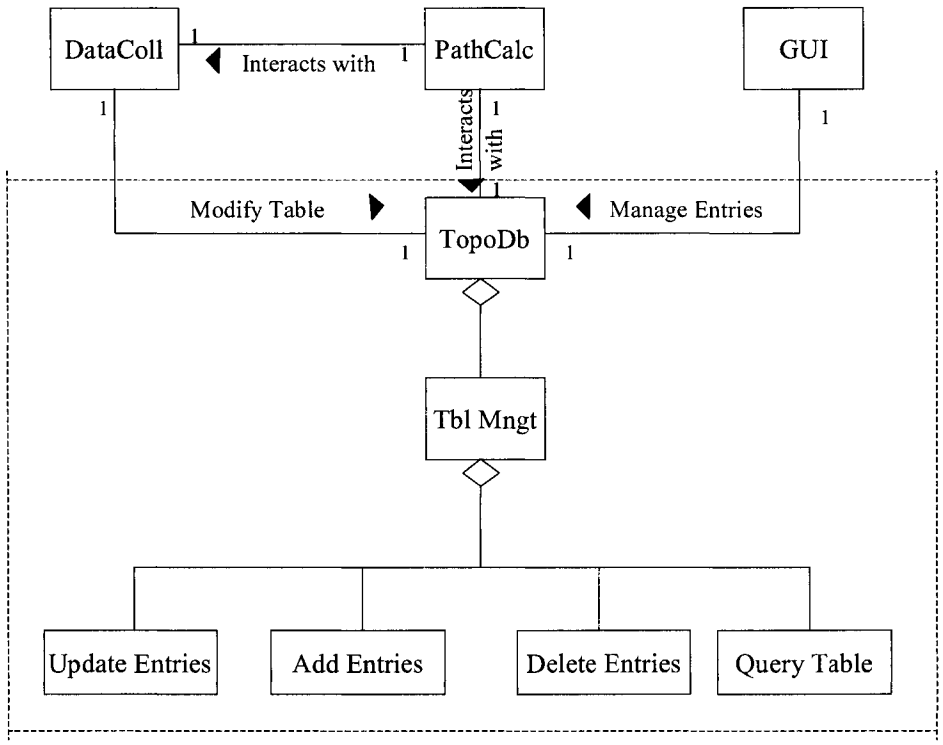

Figure 19 TopoDB Static Model

Message Sequence

Figure 20 shows the message sequence chart of the use case UC4 of the TopoDb module as others are quite similar with similar interactions. The PathCalc module will have the nodes selected for the probing in the network. It sends a probe request to the DataColl module, which probes the nodes in the network and updates the entries through Tbl Mngr (Table Manager) in the TopoDb.

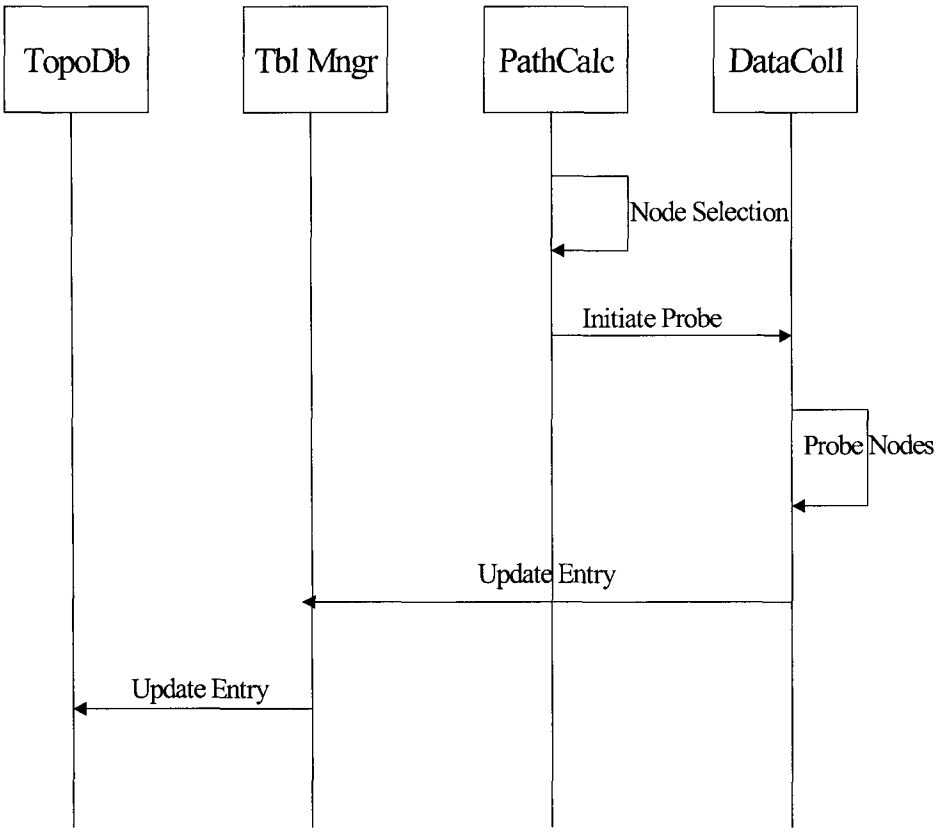

Figure 20 Message Sequence for UC4: Update Entries

3. Probe Handler Database use cases

The Probe Handler module has five use cases and they are shown in Figure 21. The use cases are described in detail in the following section.

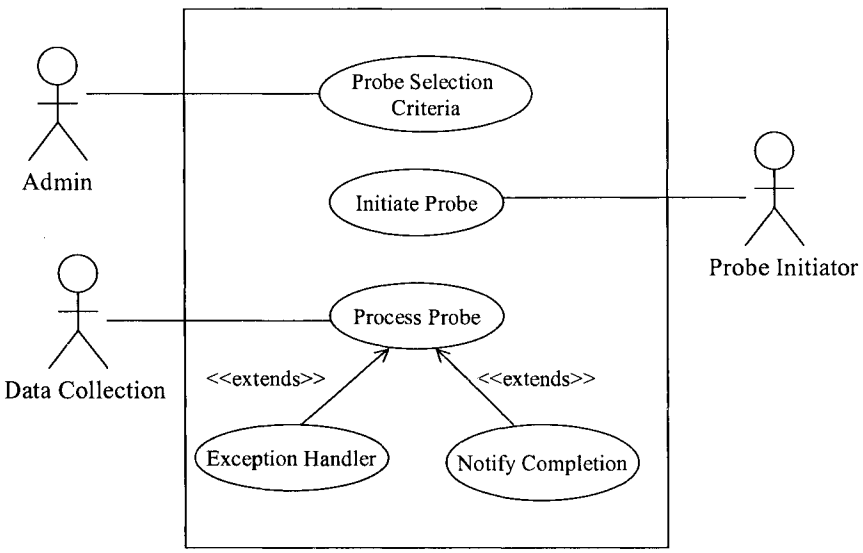

Figure 21 Use Case Diagram for PrbHndlr

UC1: Probe Selection Criteria

Summary

The Admin sets the probe selection criteria for the Probe Handler module.

Actor

Admin

Preconditions

None

Description

The probe nodes selection criteria are set by the Admin. One possible selection criterion to select the nodes on the path between the source and the destination is to select all the nodes on the path.

UC2: Initiate Probe

Summary

Probe initiator triggers the probing mechanism in the network.

Actor

Probe initiator

Preconditions

The probe selection criteria should be set by the Admin for the network.

Description

1. The Probe Initiator can be based on a timer or the Path Calculation module in the network can initiate it when it requires a path from the source to the destination in the network.

2. The Probe Initiator will trigger the Probe Handler to initiate the probing mechanism in the network.

3. The Probe Handler sends the probe request to the DataColl module.

UC3: Process Probe

Summary

DataColl processes the probe request sent by the Probe Handler.

Actor

DataColl

Preconditions

The Probe Initiator sends a Probe Request to the DataColl.

Description

The DataColl module, upon receiving the Probe Request, starts processing the probes on the nodes specified. Depending on the response of the nodes to the probes the DataColl module works on the information received as detailed in the next two use cases.

UC4: Exception Handler

Summary

The Exception Handler module is used by the DataColl to send error messages to the Probe Initiator in case the probed nodes do not respond or respond incorrectly.

Actor

DataColl

Preconditions

Probing was unsuccessful in the network.

Description

In case, the DataColl module does not receive any response from the nodes it probed, it will send an error message to the Probe Initiator indicating that the probing was not successfully performed in the network on the selected nodes. The exception handler module would give the list of the nodes that failed in the probing procedure.

UC5: Notify Completion

Summary

The Notify Completion module notifies the Probe Initiator about the success of the probing.

Actor

DataColl

Preconditions

Probing is performed successfully in the network.

Description

In the case that the DataColl module receives all the probe replies for the nodes to which it sent the probe requests, it sends out a signal to the Probe Initiator about the successful completion of probing. In case the PathCalc module is the initiator, upon receiving the probe completion notification, it starts the path calculation mechanism using the new information that might be updated by the DataColl in the TopoDb.

Static Model

The static model of the Probe Handler module is shown in Figure 22. The Admin can configure the Probe Handler module. The Probe Initiator triggers the probe procedure in the network. The Probe Handler will give the probe request to the DataColl module. The DataColl module updates the database entries in the TopoDb after probing the nodes in the network.

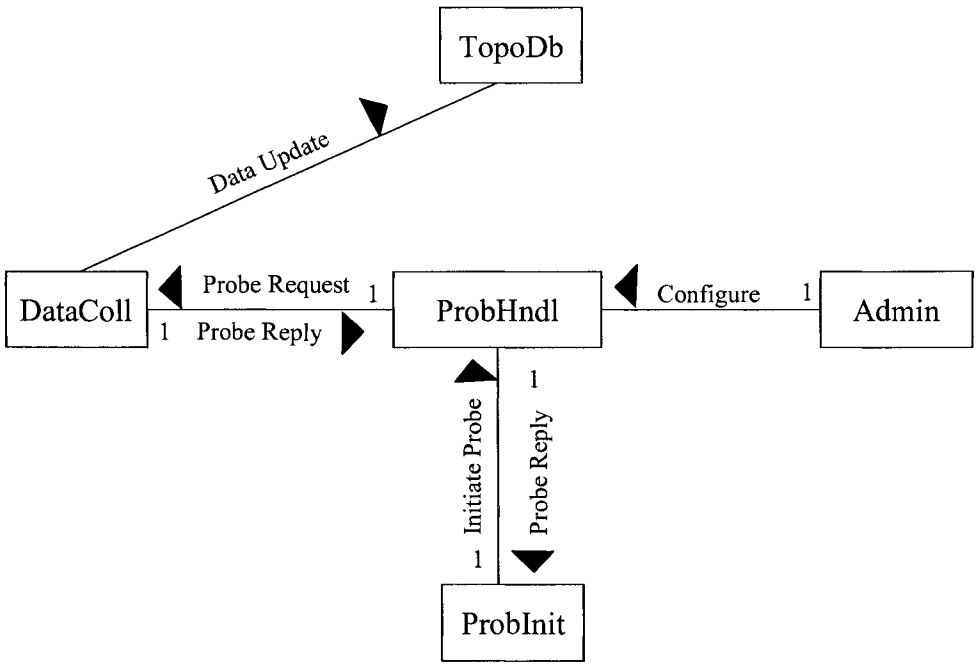

Figure 22 Static Model for PrbHndlr

Message Sequence

The message sequence chart for the Process Probe use case is shown in Figure 23, which extends the two use cases namely, Exception Handler and Notify Completion (NtfCmpl). Upon the receipt of the probe request the Data Collection probes the nodes and sends back the reply to probe handler. If the Probe Handler receives all the replies from the probed nodes, it notifies the Probe Initiator; otherwise, Probe Handler uses the ExcpHndl module to notify about the unsuccessful probes in the network with the list of the failed nodes which may result in another Probe Request being sent.

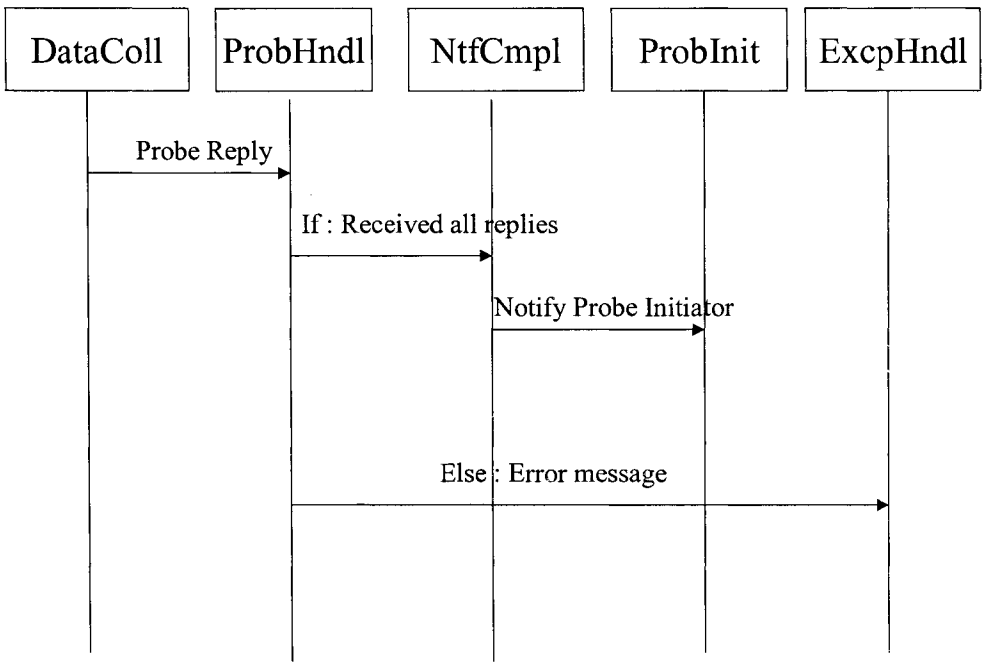

Figure 23 Message Sequence of UC3: Process Probe

State Transition

Figure 24 shows the state transition diagram for the Probe Handler module. The module powers up to be in the Init state and gets its configuration settings to be in Ready state. The Probe Initiator triggers the probing mechanism and the Probe Handler goes in the Probing state. The Probe Handler sends the probe requests to the DataColl module and goes in Sleep state to wait for the results to hear from the DataColl about the probing.

From the Sleep state it goes to Wakeup state if the probing was successful for all the nodes and then it notifies the probe completion to the triggering module. If the probing is not completed and the timer for probe replies times out, the module sends an error message notification and goes to probing state again.

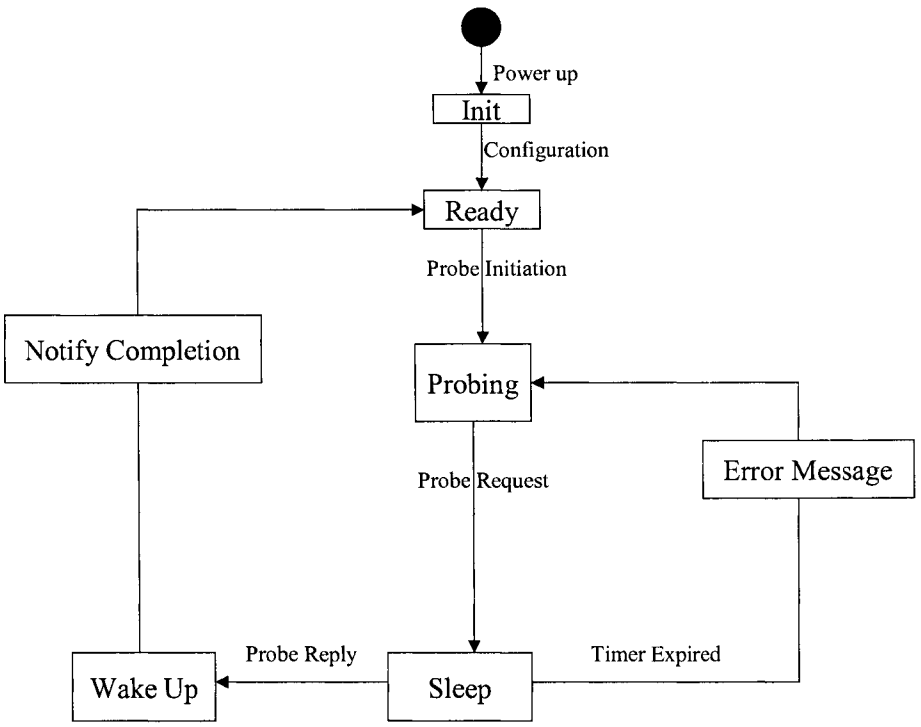

Figure 24 State Transition Chart of PrbHndlr