QoS-Based Unicast Routing Algorithms for High Speed Internetworks

by

KIA SEONG TENG

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

in the Department of Electrical and Computer Engineering

University of Manitoba

QoS-Based Unicast Routing Algorithms for High Speed Internetworks

BY

Kia Seong Teng

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of

Manitoba in partial fulfillment of the requirement of the degree

Of

Master of Science

Supervisor: Dr. Muthucumaru Maheswaran

Abstract

Thanks to the continuing research and advances in telecommunication equipment, multimedia applications such as web conferencing and real-time data streaming have become a reality and gained popularity over the years. Real-time multimedia applications required a connection-oriented data-transmission path from source to destination with a guaranteed quality of service. This requirement is different from non-real-time multimedia applications — such as File Transfer Protocol (FTP) — that do not require the data-transmission path to guarantee a certain level of quality of service.

To ensure the co-existence of these two types of service in one network, many new routing protocols have been introduced. The new routing protocols will govern the way the connection for real-time multimedia applications should be established, with minimum impact on the existing network data-transmission services. This thesis shows work that has been done in this area and proposes three new routing protocols that use the enhanced hybrid routing table. Extensive simulation results demonstrate that the three new routing algorithms achieve the requirement mentioned above with an improved average number of probes sent per connection and acceptance rate measurement.

Examiners:

Prof. Muthucumaru Maheswaran, Supervisor,
School of Computer Science, McGill University

Prof. Ekram Hossain, External Examiner,
Dept. of Electrical and Computer, Engineering, University of Manitoba

Prof. Rasit Eskicioglu, External Examiner,
Dept. of Computer Science, University of Manitoba

Contents

List of Figures

List of Tables

List of Abbreviations

ACK	Acknowledgement Message
ANP	Average Number of Probes
AR	Acceptance Ratio
BF	Bounded Flooding
DF	Directional Flooding
FTP	File Transfer Protocol
HOPARAM	Hop Parameter
OPsAR	Off-Piste QoS-aware Routing Protocol
LSD	Limited Scope Domain
LSHP	Limited Scope Hop-by-Hop Probing
LSP	Limited Scope Probing
MANET	Mobile Ad Hoc Network
MBN	Mobile Backbone Network
MOSPD	Multicast Open Shortest Path First
OSPF	Open Shortest Path First
PP	Parallel Probing
QoS	Quality of Service
SF-DCLC	Selective-Function-based Delay-Constrained Least-Cost
SP	Selective Probing
SR	Scout Routing
SRDE	Source Routing Destination Expansion
VPN	Virtual Private Network

Acknowledgements

Many thanks to my supervisor Professor Muthucumaru Maheswaran for his support, guidance, and patience during this research work. His supervision and experience enabled my ideas to be tested, improved, and validated to become practical algorithms. I also thank Professor Rasit Eskicioglu and Professor Ekram Hossain for serving on my examination committee.

Dedication

To my eight months old son and my wife for their indirect and direct support.

Chapter 1

Introduction

Routing is a process that involves finding a path with given constraints from one point to another point in a network. Minimizing some measure of distance is the usual constraint associated with routing. Emerging multimedia applications such as video conferencing and video-on-demand require the network infrastructure to support more enhanced routing capabilities than that provided by traditional routing protocols [8]. For example, in a practical network setting, a longer path may actually have the "quality" necessary for the application. Therefore, it is not enough to just route based on a "generic" measurement of distance, such as shortest path first. The requirements of the application should be incorporated into the routing process so that the path selected is best suited for the application. The *quality of service* (QoS) represents the quantitative measurement for the performance contract between the service provider and the user applications [6]. Examples of the quantitative measurement mentioned above are bandwidth, delay, and cost of a link or a node in the network; they are also referred to as QoS metrics throughout this thesis. Incorporating QoS constraints into the routing process is one way of fitting the application requirements into routing.

The QoS-based routing adds more constraints to the "path finding" process. One way of providing QoS is to over-engineer the network with extra capacity. This feasible approach also results in increased end-user cost for the services rendered. Furthermore, over-engineering the network does not always reduce congestion that might occur due to "flash crowds". As a result, QoS-routing is essential and particularly necessary in over-subscribed networks where only a few applications require QoS and most are content with "best effort". Integrated services [10, 17] and differentiated services [12] are two approaches currently available for deploying QoS. Integrated services depend on the ability to reserve resources [22]. The QoS-based routing also relies on some form of resource reservation to provide QoS. One of the advantages of the QoS-based routing approach is to reduce the "reservation flapping" that can occur with reservations and QoS-unaware routing schemes. The QoS-based routing algorithms use "network state" information to discover and determine a path with the given performance constraints. Since the network state information is highly dynamic, the distance from the source of the state information affects the precision of the state information (i.e., as one moves further away from a network element, its state information becomes imprecise due to the ineligible propagation delay from one node to the other). Handling the network state information presents the following challenges to QoS-based routing:

- Estimating the network state information as precisely as possible
- Performing the estimation with very low overhead in a scalable manner

Localizing the state information to the individual nodes in a network is a popular approach to QoS-based routing. It reduces the overhead in collecting and maintaining the network state information. Nevertheless, a node does not have any information regarding the other nodes. Therefore, the QoS-routing algorithm should perform additional work during "path discovery". Typically, the path-discovery process floods the network with probes searching for a feasible path [4]. While flooding is a robust technique, it can cause scalability and congestion problems by injecting too many probes into the network. One of the traditional routing protocols deployed widely is the *Open Shortest Path First* (OSPF) algorithm [16].

This thesis proposes three QoS-based routing algorithms that can directly use the information maintained by a QoS-enabled version of the OSPF or any "link state" algorithm to perform their routing process. These three routing protocols are: (a) *directional flooding* (DF), (b) *limited scope probing* (LSP), and, (c) *limited scope hop-by-hop probing* (LSHP).

The DF uses the topology information maintained by the OSPF algorithm to flood only along "feasible" paths. It generates enhanced routing tables and then routes the probes, which are flooded into the network in a distributive pattern by using the network topology information. In DF, the probes are not propagated along infeasible paths.

The "limited scope" algorithms use the network state information maintained by the OSPF to build a "precise" QoS-based routing table for nodes within a "small" neighborhood of a node, which is also referred to as a root node. The state information is more precise because each node is maintaining the state information of other nodes in the small vicinity. In LSP and LSHP, the network node floods probes towards the "edge" of the neighborhood (unless the destination is within the neighborhood) with the state information maintained at the node.

Enhanced (multi-path) routing tables and the three distributed QoS-based routing protocols can meet the challenges to QoS-based routing while maintaining the high acceptance ratio in the distributed routing and reducing the overall message overhead per connection.

This research was completed in 2000. Therefore, most of the papers referenced were published between 1997 and 1999. Although this research involved mostly wired networks, QoS research today focuses mostly on the mobile wireless ad hoc network, MANET.

Two main streams of provisioning QoS in MANET are IntServ and DiffServ. IntServ tends to reserve the resources along the established path between source node and the destination node(s). DiffServ does not implicitly reserve the resources along the path, and therefore does not maintain per-flow or connection-state information for each real-time connection established. DiffServ mainly differentiates the higher priority real-time flows, which require certain QoS metrics, from best-effort traffic.

The next chapter reviews some existing unicast QoS-based routing approaches; Chapter 3 discusses the QoS-based routing algorithms and the developed algorithms; Chapter 4 examines the results from the simulations performed to compare the three proposed algorithms, DF, LSP and LSHP, with an existing unicast QoS-based routing algorithm; the final chapter summarizes the thesis and presents directions for future work.

Chapter 2

Related Work

2.1 Overview

There have been many proposals for new QoS-based routing algorithms. Almost all of the current QoS-based routing algorithms can be categorized into two main streams: source-based routing and distributed routing. A new type of QoS-based routing algorithm called hierarchical routing is also discussed along with the two main-stream routing algorithms. The objective of QoS-based unicast routing is to establish a connection for an application between a source node and a destination node that satisfies the QoS metric(s) required by the application. In this thesis QoS-based routing refers to the QoS-based unicast routing unless otherwise specified. This section discusses various types of routing algorithms, related work, and differences between the proposed QoS-based routing algorithms.

2.2 Source Routing

Source routing requires each node in the network to maintain a snapshot of the global network states and network topology information through link-state or distance-vector algorithms. Based on the collected topology and network state information, the source node determines the path that the probe or reservation request should travel and attempts to request or reserve the required QoS metric(s).

Source routing greatly reduces the number of probes required to establish a connection between source and destination node, thereby producing minimum message overhead to the network. However, source routing requires each node to maintain a precise network state. Since it is impossible to have a node that obtains the precise state information of each node in a dynamic network in addition to the non-negligible propagation delay, the overall connection acceptance rate of source routing is very low.

2.3 Distributed Routing

In contrast to the source-routing algorithms, distributed-routing algorithms do not require each node in the network to maintain global network state information. However, each node is required to know the QoS metrics of each of its neighboring nodes (i.e., local state) and the topology information of the network. The distributed routing algorithms then collectively use the local state to route the QoS request from source node to destination node. Since each node in the network does not have the network state information, request messages from the source node are sent to the neighbors which satisfy the requested QoS metrics and this process continues.

Eventually the destination node receives one or more request messages from its neighbor(s), and then the destination node evaluates the accumulated information carried by the request probes. Based on the evaluation result, the destination node may decide to send an acknowledgement message to reserve the QoS metrics requested along the path discovered by the request probe. By flooding the request messages to the network, distributed routing algorithms cause a high average of network message overhead per connection. However, distributed routing algorithms such as flooding methods are known to have a high average of connection acceptance rate.

2.4 Hierarchical Routing

Another type of QoS-based routing algorithms is hierarchical routing. Hierarchical routing algorithms require the network to be grouped or partitioned into multiple clusters. Each cluster contains a group of nodes that usually have identical prefix IP addresses. Since each cluster of the network can be seen as a logical node representing a group of nodes having one QoS metric, their QoS metrics must be aggregated. Aggregating the QoS metrics increases the degree of network state imprecision. Hierarchical routing, therefore, shares the disadvantages of source routing. Several distributed routing algorithms proposed recently are discussed in more detail below.

2.5 Selective Probing

Selective probing (SP) is one approach for QoS-based routing [4]. In SP, every node is required to maintain just its local state. When a connection request arrives at a node, the node sends a probe to one or more neighbors that are connected by links without violating the given QoS metrics constraints. The nodes also prevent the probes from circulating in loops by forwarding the same probe once per node. When the probes reach the destination, the destination selects the probe with the best-path attributes and sends an *acknowledgement message* (ACK) toward the source node. The ACK attempts to reserve the resources required to deliver the requested QoS along the path as it propagates. If the ACK fails due to the lack of resources, a failure message is sent back to the nodes, from which the ACK was received, so that the resources reserved previously will be released. [4] proposed several optimization techniques such as recording more than one probe that reaches a node, trying to reserve other paths learned from other probes recorded when a failure message is received, and iterative routing. SP algorithm attempts to reduce the high degree of network message overhead by not sending probes to neighbor(s) that do not satisfy the QoS requested.

A well-connected and lightly loaded network defeats this objective and SP will behave like a flooding algorithm. The iterative routing algorithm works by first setting the Time to Live or age of the probe to be the shortest path from the source node to the destination node then selectively flooding the network with the aged probe from the source node to the destination node. If the first attempt fails, the source node sets the age of the second probe to infinity and starts the selective flooding again. Iterative routing works well in lightly loaded networks by reducing the overall network message overhead. However, when the network is heavily loaded, the second iteration is usually invoked. The second iteration further increases the overall message overhead per connection and the load of the network by injecting more probes. In the worst case, iterative routing requires twice as much routing response time and higher network message overhead to discover that there is no feasible path in a heavily loaded network. Our proposed algorithms will only send probes to the feasible path by using both the topology and accurate link states information. Therefore, we expect proposed algorithms will have a lower network message overhead and have similar acceptance ratios compared to SP.

2.6 Parallel Probing

Parallel probing (PP) is another approach to path establishment for real-time channel QoS-routing [15]. The PP attempts to combine two existing distributed routing approaches — preferred neighbor and flooding approaches — but requires a set of intermediate nodes to be assigned as specific nodes in the network before the QoS-based routing. The specific nodes are responsible for probing several paths simultaneously and employing different heuristics on each path. The two heuristics employed in their simulation work are shortest path first and lightly loaded link first.

The PP achieves a higher average acceptance ratio than the flooding algorithm and an average connection setup time closer to the flooding approach. Also, the average route distance achieved by this algorithm is closer to the flooding algorithm. However, it requires a network administrator to carefully assign the intermediate nodes in the network. The specific nodes are crucial for the performance of the algorithm. PP also requires every successful probe to reserve the resources in advance and later release them if inferior to the other heuristic. In the worst case, a well-connected network link's resources will be reserved at least once and most of them will be released. This involves excessive back tracking, which is expensive and causes link state fluctuation. PP is also unfair to other traffic that is also attempting to reserve the same resources. In a dynamic network, it is difficult to foresee which heuristic approach is best for the links ahead. Furthermore, only specific nodes can have probes with different heuristics to be launched; therefore, carelessness in assigning the specific nodes may result in performance degradation. In the proposed algorithms, the edge nodes located at H hops away from the source node are the specific nodes by default; therefore there is no extra effort needed to allocate the specific node to ensure the performance. The proposed algorithms will not reserve the resources during the probing phase, but will reserve the resources in the acknowledgement phase, which avoids the excessive resource reservation issue.

2.7 Scout Routing

The *Scout routing* (SR) algorithm [3] is yet another distributed QoS-based routing approach. It requires every node in the network to constantly flood a small and fixed-sized scout message to its neighbors, thus flooding the network. The scout messages aggregate the link-state information of the nodes they have traveled through, thereby developing multi-path QoS "state" information from the sending node to the receiving nodes. Upon receiving a QoS request message, the source nodes use the multi-path QoS "state" information table and decide which path(s) the request message should take by evaluating the QoS metrics requested against the aggregated QoS state carried by the scout messages earlier. Even though this approach can achieve a higher acceptance ratio, it also puts a heavy burden on the network with its scout messages, which are crucial steps to collect accurate aggregated link-state information. By aggregating the QoS metrics, the degree of network state imprecision again increases. The proposed algorithms use the existing link-state information which is also maintained by OSPF to generate a multi-paths routing table — the link states from other nodes will not be aggregated.

2.8 Bounded Flooding

The *Bounded Flooding* algorithm (BF) [13] is a distributed flooding algorithm, where probes are allowed to traverse in a pre-specified search-scope. The search-scope is the hop count of the nth minimum-hop route for a set of alternative routes from the source to the destination. Nodes are required to maintain the almost static global network-topology information and a distance table with minimum hops counts to every other node. Every probe as well as the visiting node maintains the ID of intermediate node traversed. The node's resource used in the connection admission test is calculated as the node's resources subtract both the resources promised to probes and resources reserved. This is similar to PP algorithms, which reserve the QoS metrics requested at visit. By limiting the hop-counts of probes, BF dramatically reduces message overhead and operational cost. Unfortunately, its performance or acceptance ratio is highly topologically based. In networks with less connectivity, its acceptance ratio is close to the flooding algorithm. On the other hand, the algorithm will have a noticeable difference in blocking probability compared to flooding algorithm in well-connected networks. The proposed algorithms do not reserve resources at visit. Since network topology and accurate link states information are used during the probing phase, no limitation on probe hop-counts is enforced. Probes are rejected and discarded only if the QoS metrics requested cannot be fulfilled.

2.9 Selective-Function-based Delay-Constrained Least-Cost

Selective-Function-based Delay-Constrained Least-Cost (SF-DCLC) [14] is a distributed hop-by-hop routing algorithm. SF-DCLC requires that each node maintains two metric vectors — the least-delay vector and the least-cost vector — to any other destination nodes in the network. SF-DCLC also assumes that the two vectors are up-to-date and the contents of the two vectors do not change during the route setup period. Given a request delay bound, D, to a destination node, t, source node, s, will first check if the least-delay vector to t can satisfy D. If D can be satisfied, then s check is the delay for the least-cost vector to t to satisfy D. If both can satisfy D, then a PATH_CONTRUCTION message is send to the next hop, v, found in the least-cost vector to t. If the least-cost vector cannot satisfy D then s invokes the weight() and extract() function to evaluate and choose the neighbor node, w, which satisfies the delay bound, D, and send the PATH_CONTRUCTION message to w. The intermediate nodes that receive the PATH_CONTRUCTION message also perform the same checks and reconstruct the PATH_CONTRUCTION message by adding the delay accumulated and appending it to the partial connection path so far to the message. Intermediate nodes must take the delay accumulated so far when checking and evaluating D. The PATH_CONTRUCTION message may reach the destination in a finite time or the PATH_CONTRUCTION message is discarded due to insufficient resources and the connection is rejected.

The simulation is also performed to measure the *Cost Inefficiency* (CI), *Optimality Miss Ratio* (OMR) and *Average Number of Messages* (ANM) of SF-DCLC against Constrained Bellman-Ford (CBF) [19], least-delay path (LDP), and DCR [18] algorithms in two different network topologies. The results show that SF-DCLC has lower CI, lower OMR, and similar ANM to LDP and DCR. SF-DCLC is insensitive to network sizes and delay levels. SF-DCLC assumes that the network states are stable and performs "source" routing in each intermediate node. SF-DCLC may not perform well in a practical network since network states are highly dynamic and the distance vectors may not reflect accurate information for nodes far away from the source node. SF-DCLC sends the PATH_CONTRUCTION message to one path (optimum path) that is very likely to fail in highly dynamic network. The proposed algorithm utilized a multi-path routing table. Probes are forwarded to more than one path toward the destination nodes if the topology and network state allow. The feasible path discovered by probes may not necessarily be the shortest path from the source node to the destination node.

2.10 Off-Piste QoS-aware Routing Protocol

Off-Piste QoS-aware Routing Protocol (OPsAR) [1] uses a caching method to implement the multi-path routing from a transaction-source, s, to a transaction-destination, t. The total number of paths that can be branched in a transaction is controlled by the branch degree and by the ticket base [5] — both parameters pre-specified in s. The length of each path is also governed by an off-piste counter, which limits the distance from the shortest-path. OPsAR requires every node to process any messages traversing the node to construct or update its *knowledge-state*. The *knowledge-state* of a node is a bounded list of pairs ⟨target nodes, outgoing-link⟩, and each list has the resource availability toward the target node (s and t of a transaction) and from the target node with respect to that outgoing-link (path to the target node).

OPsAR also requires every node to maintain its local state to the neighboring nodes. OPsAR implements three phase protocols: Try_phase, Scan_phase, and Try_2_phase. Node s initiates the Try_phase by composing the TRY-MESSAGE, which contains the transaction-destination, QoS resources requested, and list of traversed nodes. Node s or the intermediate node that received the TRY-MESSAGE first evaluates the shortest path unicast route against the requested QoS requested by referencing to the *knowledge-state*. If the shortest path can satisfy the QoS requested, then the TRY-MESSAGE is sent to the next hop in the shortest path at the same time reserve the QoS requested. TRY-MESSAGE messages(s) are also sent to the deviation(s) from the shortest path, which also satisfies the QoS requested. The length of the off-piste path(s) (the deviation(s)) is governed by the off-piste counter. If any of the intermediate nodes in the shortest path are unable to satisfy the QoS requested, then a backward-path NACK-MESSAGE is sent to release the resources reserved so far. The TRY_MESSAGE is marked as failed and sent to the transaction-destination to initiate the Scan_phase.

The Scan_phase is similar to the Try_phase, however it is initiated from transaction-destination to transaction-source, it does not reserve resources, and it makes use of each node's *knowledge-state* to route the SCAN-MESSAGE to the transaction-source. The transaction-source chooses the "best" route discovered by the SCAN-MESSAGE and initiates a Try_2_phase to reserve the QoS requested along the "best" path. It also tries out the off-piste path offered by the node *knowledge-state* if the "best" path failed. A NACK-MESSAGE is sent back to transaction-source if Try_2_phase failed. The connection is successful if the transaction-destination received the TRY-MESSAGE and responds with an ACK-MESSAGE. Extensive simulation in terms of message overhead, amount of concurrent transactions, number of edge nodes, and number of destination nodes was run against RSVP, S-QMRP*, S-QMRP*D, and OPT algorithms [1].

The simulation results show that the OPsAR protocol was closer to the results of the OPT than the other protocols. The OPsAR required many parameters to be set and carelessness in assigning any of the parameters greatly impacts the performance of OPsAR. The three phase protocol may try to reserve the same resources from the same nodes or links more than once, and then release them when the connection fails. This may contribute to the network state instability and is unfair to other traffic which also is trying to reserve the same resources. The node's *knowledge-state* must be updated regularly to cope with the dynamic network states. An inaccurate *knowledge-state* may mislead the TRY-MESSAGE and SCAN MESSAGE messages and result in increased message overhead and connection failures. The proposed algorithms maintain only the records regarding the QoS metrics requested by and reserved for the probes and acknowledge message respectively. The proposed algorithms do not reserve the resources at visit. They use the link-state information broadcasted by the nearby nodes within the H hop counts from the source or intermediate nodes only.

2.11 Source Routing Destination Expansion

Source Routing Destination Expansion (SRDE) [7] is a multi-constrained-path QoS-based source routing algorithm using a constraint-aggregation heuristic approach. SRDE converts a single-source, single-destination QoS routing problem to a single-source multiple-destination QoS routing problem. Constraint-aggregation heuristics combine constraints into one cost function. The conversion increases the probability of finding a feasible path by keeping more accurate link-state information for those links located h hop(s) from the destination node.

The complexity of both the conversion process and the Extended Bellman-Ford algorithm [21] for finding the feasible path in single-source, multiple-destinations can be less than or equal to the standard Bellman-Ford algorithm by properly selecting the value of h. For a given source node, s, and destination node, t, SRDE first propagates the constraints requested from t to its neighboring nodes by subtracting the weight (QoS resources) from the constraints and so obtains a new and different constraint on each of the neighboring nodes. The neighboring nodes are now referred to as new destination nodes with the new constraints requested. Node t and the links traversed by the original constraints are removed. The expansion process mentioned above continues for h number of times (h hops from t) where the neighboring nodes are now the new t. The Extended Bellman-Ford algorithm is then used to search for a feasible path from s to the new destination nodes (nodes that located h hops away from t).

The Extended Bellman-Ford algorithm can also be replaced by any existing shortest path first routing algorithm. Simulation results showed that SRDE achieved a much better acceptance ratio than the standard Bellman-Ford. Acceptance ratio was the only performance measurement used in this paper. The simulation also showed that the acceptance ratio in a smaller network was better than a larger network. This implies that SRDE may not scale well in large network. The simulation was run on static topology and network states, which do not reflect a real and practical network. The proposed algorithms also used the similar idea of h count from a node. However, they consider the h counts from the source node (not from the destination node) and only the topology (connectivity) information are propagated throughout the network, not links weighs (link states). The proposed algorithms are based on distributed routing approaches. Again, the feasible path(s) discovered by probes may not be necessarily the shortest path from the source node to the destination node.

2.12 QoS Provisioning in Large Scale Ad Hoc Networks

This paper proposes a scalable QoS architecture that adaptively manages available bandwidth and provisioning QoS in large-scale mobile ad hoc networks [20]. The proposed architecture extends the LANDMARK routing algorithm [11] to support QoS routing and it is also capable of incorporating Mobile Backbone Network (MBN) to further improve scalability in large area networks. The call admission control implemented in the source node accepts or rejects a QoS request based on the estimated minimum and maximum available bandwidth to the destination node. If the QoS request is accepted, the data traffic is marked as a real-time flow. The landmark (the representative node in a landmark group) is responsible for estimating the minimum and maximum available bandwidth to any other nodes in the group by using local QoS routing algorithm. Then the estimated minimum and maximum available bandwidth is broadcasted throughout the network proactively by the extended LANDMARK routing algorithm.

To prevent performance degradation due to network connectivity changes and mobility-triggered congestion, the proposed architecture implements congestion detection and congestion control in intermediate nodes. The network is congested if the node's measured channel utilization ratio is above a certain threshold value. The intermediate nodes try to reduce the best-effort traffic rates to relieve the congestion and suspend some real-time flows if the network is under heavy congestion.

The simulation results showed that the proposed QoS architecture with the extended LANDMARK routing algorithm achieves lower average end-to-end packet delay and higher packet delivery fraction compared to applying the LANDMARK routing algorithm alone. By incorporating both MBN and extended LANDMARK routing algorithms, the QoS architecture can achieve greater simulation results in average end-to-end packet delay and packet delivery fraction, and much lower average call admission delay even under mobility.

Chapter 3

QoS-based Routing Algorithms Discussion

3.1 Overview

QoS-based routing consists of two main ingredients: the routing protocol and the routing algorithm. The routing protocol is responsible for accurately collecting network states, and updating and maintaining the routing table at each network node. The network node then refers to the collected network states and the routing table generated to route "best effort" messages as well as QoS metrics request messages (probes) to the destination node efficiently.

Network sizes, capacities, functions, and technologies have been evolving to support more demanding real-time applications from online gaming to point-to-point secure VPN connections. The routing table and the network state collected serve an important role in helping network nodes in performing the admission test of such real-time applications. In reality, network states are highly dynamic. The link-states information from the node received from other nodes that are a few hops away may have been already out of date. This may be because the available resource advertised in the link-state messages may already have been reserved by another application. Therefore, the routing algorithm is introduced to compensate for the network states' imprecision by increasing the admission ratio with the cost of increasing overall message overhead per application admitted.

3.2 Routing Protocol

The routing algorithms discussed so far assume the network under consideration has some kind of routing protocol in place and is collecting accurate network states. The two routing protocols widely used are link-state algorithm and distance-vector protocol. Of the two routing protocols, link-state algorithm is more popular due to its nature of scaling well to larger networks. As the network continues to grow larger, link-state aggregation was proposed to reduce the message overhead introduced by link-state update messages from each network node. The drawback of link-state aggregation is the increasing degree of network states imprecision.

This thesis proposes multi-path routing tables that use the network topology information extracted from link-state update messages to generate the alternative paths. The multi-path routing tables provide at least one path through neighbor node(s) to any destination nodes in the network. The advantage of a multi-path routing table is to reduce the message overhead introduced by the brute force flooding routing algorithm where probes are routed only to the nodes that will lead to the destination node.

This thesis also proposes that every node in the network maintain a small neighborhood of link states and multi-path routing table to any nodes within the small neighborhood. The neighborhood is small so the link states and topology information collected are always up-to-date. Based on the accurate information collected inside the small neighborhood, source routing can be deployed to further reduce the message overhead while not affecting the acceptance ratio. The same multi-path routing table can also be used by "best effort" routing. In addition, the node has an option to choose the least-loaded path to the same destination through the other neighbor nodes as allowed by the underlying network topology.

3.3 Routing Algorithm

The purpose of the QoS-based routing algorithm is to find a path from the source node (service provider) to a destination node (client) that also satisfies the QoS metric required to support the applications or services rendered. There are three major types of QoS-based routing algorithms: source routing, distributed routing, and hierarchy routing.

Source routing algorithms use the collected network topology and link-states information to determine the feasible path from the source node to the destination node. Despite having lower message overhead per connection, it is known to have a lower acceptance ratio. Since network states are highly dynamic, the source node may have used inaccurate link-states information in its routing algorithm for finding the feasible path.

Distributed routing algorithms require source nodes to send probes to the network for discovering the path to the destination node. Probes are admitted or discarded at the receiving node with the accurate link-state information. Distributed routing algorithms are known to have a high acceptance ratio and message overhead per connection. Hierarchical routing algorithms require the underlying network to be partitioned to multiple clusters by aggregating the link-states information of all nodes inside the cluster. Hierarchical routing algorithms scale very well in large networks but also share the advantages and disadvantages of source routing. Aggregating the link-states information increases network states imprecision.

This thesis proposes several distributed QoS-based routing algorithms to use the multi-path routing table discussed in section 3.1. The proposes routing algorithms are designed based on the idea of forwarding information to the neighbors nearby, who can hear you and also know the designee to whom you wish the information forwarded. The advantages of the proposed routing algorithms include forwarding probes to paths that can lead to the destination and also satisfy the QoS metrics requested. This will greatly reduce the overall message overhead per connection and maintain having a higher acceptance ratio in distributed routing algorithms. The proposed algorithms are also designed to be used in highly dynamic networks as well as stable networks.

Chapter 4

QoS Routing Model and Algorithms

4.1 Overview

There are typically two main streams of connection establishment phases in QoS-based routing protocols: (i) The probing phase followed by the acknowledgement phase, and (ii) the reservation phase followed by the relaxation phase. The latter is used in PP algorithms, which start by reserving QoS metric(s) required, then either relaxing or releasing the QoS metrics reserved previously in the relaxation phase. This method is unfair to other traffic in the network and is directly responsible for driving down the overall connection acceptance rate. Since the method starts by reserving node(s) or link(s) resources during the probing phase, the node(s) or links(s) may not have enough resources to meet other requests and will thus reject the connection. The resources reserved in the same node(s) or link(s) may later be released in the relaxation phase due to excessive reservation. Furthermore, the performance of this method becomes unacceptable when the network is already heavily loaded with other traffic.

4.2 Data Structures

A probe, ACK, or failure message consists of the source node's ID, s; destination node's ID, t; probe-unique ID for a given session, *cid*; intermediate node's ID, k; message type, *type*; QoS information including type of QoS metrics requested, *qos.type*; probe current delay, *qos.curd*; QoS metrics constraints requested, *qos.con*; and lastly, hop counts traversed by the probe, *hop*. The message would be represented as message $[s, t, cid, k, type, qos, hop]$. The message types are: probes, $type = 1$; acknowledge, $type = 2$; failure, $type = 3$.

Every node is required to maintain several probe tables to record and identify each probe that arrived at this node. The probe table consists of probe-unique identification, *pbCid*; QoS metrics type (bandwidth, delay, etc), *type*; amount of reserved or requested resources, *rsvQoS*; Current probe delay, *curDelay*; current probe hop counts, *hop*; the node's ID which send this probe, *revFromRtr*; and the flag to indicate whether the resources specified are reserved, *acked*. If a resource was successfully reserved, *acked* is set to 1 and 0 otherwise.

4.3 QoS Metrics Requirement and Satisfaction

Generally, there are two types for QoS metrics: subjective and connective. Examples of subjective QoS metrics are bandwidth and buffer size, whereas delay, delay jitter, and cost are connective QoS metrics. A connection request with M, the QoS metrics requirement, requires QoS-based routing algorithms to find a path P from the source nodes s, to a destination node (client), t such that:

$$P = nodes(s, i, j, \ell, t) \tag{4.1}$$

where s is the source, t is the destination node, and i, j, and ℓ are the intermediate nodes. The order of the nodes in path P is such that: node s is adjacent to node i, node i is adjacent to node j, and finally node ℓ is adjacent to node t to complete the path.

We let $m(P)$ be the QoS metrics reserved from all the nodes located along path P, i.e.,

$$m(P) = m(s, i, j, \ell, t) \tag{4.2}$$

This equation can also be further expressed as:

$$m(P) = m(s, i)\ E\ m(i, j)\ E\ m(j, \ell)\ E\ m(\ell, t) \tag{4.3}$$

where E may be expressed as the $\wedge$ or $+$ operator depending on the type of the QoS requested. In addition to finding path P, the QoS metrics reserved in path P, $m(P)$ must also satisfy the requested QoS metrics, M:

$$m(P)\ C\ M \tag{4.4}$$

where C can be any comparator such as $<$, $\leq$, $\geq$, $>$, $=$, or $\cong$.

If M is a subjective QoS metric, then Eq. 4.3 should be represented as:

$$m(P) = m(s, i) \wedge m(i, j) \wedge m(j, \ell) \wedge m(\ell, t) \tag{4.5}$$

Eq. 4.4 can now be represented as:

$$(m(s,i)\ C\ M) \wedge (m(i,j)\ C\ M) \wedge (m(j,\ell)\ C\ M) \wedge (m(\ell,t)\ C\ M) \tag{4.6}$$

This equation shows that in order to establish the connection with subjective QoS metrics, M, QoS metrics that are reserved from each pair of connected nodes in path P must also satisfy M.

If m is a connective QoS metric, then Eq. 4.3 should be represented as:

$$m(P) = m(s,i) + m(i,j) + m(j,\ell) + m(\ell,t) \tag{4.7}$$

Eq. 4.4 should now be represented as:

$$m(s,i) + m(i,j) + m(j,\ell) + m(\ell,t)\ C\ M \tag{4.8}$$

which shows that in order to establish the connection with connective QoS metrics, M, the total amount of QoS metrics that are reserved from each pair of connected nodes in path P must also satisfy M.

4.4 Multi-path Routing Table (MRT)

In this section, an overview is presented for the new multi-path routing table, MRT, that is generated based on the information carried by the traditional link-state algorithm. To support the OSPF and even MOSPF (Multicast Open Shortest Path First) routing protocol, link-state update messages must include the local state of the broadcasting node such as distance, bandwidth, delay, and so on to its neighbors.

From Node	Neighbors	QoS	Sequence Number
A	B,C,D	Q_{AB}, Q_{AC}, Q_{AD}	S_A
B	A,E	Q_{AB}, Q_{BE}	S_B
C	A,E	Q_{AC}, Q_{CE}	S_C
D	A,F	Q_{AD}, Q_{DF}	S_D
E	B,C,G	Q_{BE}, Q_{CE}, Q_{EG}	S_E
F	D	Q_{FD}	S_F
G	E	Q_{EG}	S_G

Table 4.1: Link-state table constructed from link-state-update messages.

Table 4.1 shows all the link-state update messages collected in a given network. The link-state table is maintained by comparing the corresponding sequence number stored in the link-state table with the subsequent update messages. If the sequence number in the update message received is newer, then the corresponding entries in the link-state table are updated. Otherwise, the link-state update message is discarded.

Based on the link-state table, we were able to extract the interconnectivity information among the nodes in the network. **Figure 4.1** shows the topology constructed based on **Table 4.1**. From the extracted topology information, we were able to generate the MRT routing tables. There are two types of MRT routing table — MRT Type 1 and MRT Type 2. These two types of MRT tables are discussed in more detail in the next sections.

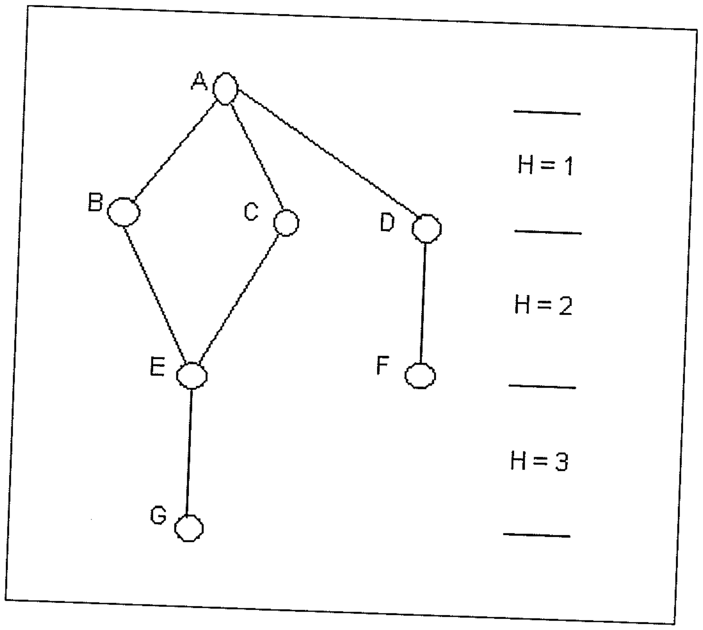

Figure 4.1: Topology information extracted from link-state table.

Destination Node	Next Hop(s)	Shortest Distance
B	B, C	1
C	B, C	1
D	D	1
E	B, C	2
G	B, C	3
F	D	2

Table 4.2: MRT-1 routing table of node A constructed from **Figure 4.1**.

4.4.1 MRT Type 1 (MRT-1)

Assuming unit distance between each node in the connectivity information shown in **Figure 4.1**, the MRT-1 routing table can be constructed for any node in the network. **Table 4.2** shows the MRT-1 routing table constructed by node A. Using the node A MRT-1 routing table, if node A needs to send a probe to node G, node A must send the probe to node B and send a similar probe to node C as next hops. The shortest distance from node A to node G is 3.

The MRT-1 routing table enables a node to send probes to the destination in parallel and avoid sending probe to neighbor nodes that are not located in a feasible path. In combining the MRT-1 routing table and the link-state table, the QoS-based routing algorithms proposed reduce the average number of probes per connection when compared to brute-force flooding QoS-based routing algorithms. For example, node A knows from **Table 4.2** that the next hops to destination node G are node B and node C. Referring to the link-state table, node A can now evaluate the QoS metrics (local state) to node B and node C from Q_{AB} and Q_{AC} with the requested QoS metric respectively, from which decisions are made to send probes to the nodes which satisfy the QoS metrics requirement.

4.4.2 MRT Type 2 (MRT-2)

The MRT-1 routing table enables QoS routing algorithms to send and forward probes to the nodes in potential feasible paths only. To further reduce the average number of probes sent per connection, another type of MRT routing table is introduced — MRT Type 2 (MRT-2). The MRT-2 routing table is based on the idea of maintaining a small vicinity measured by hop count, HOPARAM (H), with precise link-state information collected from link-state update messages broadcasted

Destination Node	Next hop(s)	Shortest distance
B	B	1
C	C	1
D	D	1
E	B;E , C;E	2
G	E	3
F	D;F	2

Table 4.3: MRT-2 routing table for node A.

from other nodes in the small vicinity. Using the connectivity information collected shown in **Figure 4.1** and setting the HOPARAM to 2, resulted in the MRT-2 routing table for node A.

Table 4.3 shows the MRT-2 routing table constructed for node A, in which a semicolon means "next node in the path". By assigning a value to H, a root node — in this case node A — is maintaining the precise link-state information to a number of nodes in the small vicinity, where those nodes inside the small vicinity are at most H hop count away from node A. Value H can be adjusted to control the level of confidence in the degree of link-state information precision maintained. Setting $H = 1$ gives the highest degree of confidence in link-state information accuracy, where it will be similar to node A's MRT-1 routing table.

For example, in this section, we set $H = 2$ and assumed that the rate of link-state information update is greater than the rate of change of the link-state in the network. Link-state update messages for the MRT-2 routing table are not required to be propagated and processed by nodes that are located H hop count away from the broadcasting node. This is because the link-state information carried is not required for generating the MRT-2 routing table. Therefore, maintaining the MRT-2 routing table requires very low or even eliminates ($H = 1$) the overhead introduced by link-state update messages. However, update messages related to topology changes such as link down or recovery are still required to be propagated to all the nodes in the network for regenerating the MRT-2 table with the correct network topology information.

To route a probe from node A to node G, node A knows from **Table 4.3** that the intermediate destination is node E. Furthermore, by looking up the node A MRT-2 routing table, node A knows that the two potential feasible paths are F1: node B then node E and F2: node C then node E. From the link-state table shown in **Table 4.1**, node A can evaluate the QoS of the two potential feasible paths by evaluating Q_{AB} and Q_{BE} for F1 and Q_{AC} and Q_{CE} for F2 against the requested QoS metrics. In combining the MRT-2 and link-state tables, node A sends the probe through the feasible path(s) and avoids sending probes to other potential feasible paths that do not satisfy the QoS metrics requested, according to the information found in link-state table. Therefore, this method further reduces the average number of messages overhead per connection by sending probes to the potential feasible paths that also satisfy the QoS metrics requested by looking ahead for the QoS metric of the nodes and links located H hop count away from the root node.

The MRT-1 and MRT-2 routing tables both use the update messages generated by the traditional link-state algorithm to generate the hybrid routing tables. The network vicinity formed by the nodes and links within the HOPARAM of the node is referred to as the *limited scope domain* (LSD). The generated routing table will have multiple paths to a destination if the topology allows. Each alternate path contains a list of intermediate nodes up to HOPARAM−1 to the edge nodes if the destination is out of the LSD, or to the destination nodes if it is in the LSD.

The hybrid routing tables proposed not only serve as traditional topology-based routing tables but also contain at least one path that leads to the destination node or to the intermediate nodes if the destination node is outside of the LSD. Therefore, the probes will "flow" from the source to a given destination outside the LSD in the network. The routing is considered brute force even with the topology and local link-states information taken into consideration when a probe is forwarded to a node within the LSD. A scalable QoS-enabled network can be created by just having all the nodes in a network not forwarding the link-state update messages for a longer time window or hop count greater than H unless the update messages contain information about topology changes.

4.5 Proposed QoS-based Routing Phases and Algorithms

Proposed QoS-based routing protocols have three phases: (a) probing the network for finding feasible path(s) from the source node to destination node, which satisfies the QoS metrics requested. (b) Acknowledgement or reservation of the QoS metrics negotiated with the nodes and links during the probing phase, and finally (c) connection failure handling when the acknowledgement phase fails.

The probing phase begins when the source node, s, receives a connection request from a client at the destination node, t, with a given QoS metrics requirement (e.g., with a bandwidth, delay, or both specified). The actual events that occur in the probing phase differ based on the strategies used in implementing the probes forwarding policies and routing table. For example, in the selective probing technique, the source node, s, sends the probe to its neighbors that do not violate the QoS requirements. The neighboring nodes propagate probes further while ensuring the probes do not go in loops.

For a network with sufficient resources, one or more probes will reach the destination node, t. The paths traversed by the probes are candidate feasible paths. The destination node, t, will select one of these paths and send an *acknowledgement message* (ACK) toward s node. The ACK message attempts to reserve the required resources along the path, which has been discovered during the probing phase. Although the feasible paths have enough resources during the probing phase, they may not have sufficient resources at acknowledgement phase due to network state variations. The receipt of an ACK message by the source node, s, indicates that a path with the given QoS metrics constraints has been successful between the source and the destination node. Otherwise, a failure message is used to release any resources reserved in this session.

The last phase of the connection establishment is the connection failure phase. It is implemented as a timeout, i.e., the resources reserved to support the desired QoS level are freed if the reserved resources have not been used within a specified time window. Timeout also erases the record of the QoS metrics negotiated by the probe if it is not confirmed in a specified time window. In an ideal situation, an explicit failure message is sent to release the resources reserved along the path towards the destination node, t.

4.5.1 Directional Flooding (DF)

The directional flooding routing algorithm uses routing table MRT-1, which provides the next hop(s) to all available feasible paths leading to the destination node in the network. DF also uses the link-state table to accurately store and reference its local state information.

Upon receiving a request message, DF knows the next hop(s) that leads to the destination node (the client) from the MRT-1 routing table by matching the requested QoS metrics with the local state information of each next hop from the link-state table. DF then forwards the probe to the next hop that has sufficient resources to satisfy the requested QoS metrics. Eventually, at least one probe will reach the destination node. The destination node initiates the acknowledgement phase upon receiving the probes. Therefore, the probe turns around and becomes an ACK message to reserve the QoS metric previously negotiated or requested by the probe from each node through which the probe has traveled. A connection is established when the source node receives the ACK message from the destination node. Timeout and failure messages are implemented to release the resources reserved if the acknowledgement phase fails.

The primary advantage of the DF algorithm over the traditional flooding algorithms is the reduction in overall message overhead per connection by preventing nodes from forwarding probes to the next hop(s) that may not lead to the destination node. DF is a simple and practical distributed QoS-based routing algorithm, which is based on the local state information and topology information extracted from link-state update messages.

The generic algorithm of DF for a connection request with destination node t, requested QoS metrics qos, and root node or intermediate node i, is presented below:

```
DFfwd($t, qos$) at node $i$

1. for each node $\ell$ connected to $t$
```

```
2.   if never sent probe to $\ell$ before then
3.     if $m(i,\ell)\ C\ qos$ then
4.       send probe$[s, t, cid, i, 1, qos, hop + 1]$ to $\ell$
5.     end if
6.   end if
7. end for
```

DF refers to the MRT-1 routing table for a list of neighbor node(s) (next hop(s)), ℓ, which lead to t. DF only forwards a probe to the neighbor nodes if the expression $m(i,\ell)\ C\ qos$ is evaluated and returned true. The evaluation is based on the link-state information collected and stored in the link-state table.

4.5.2 Limited Scope Probing (LSP)

As the name implies, LSP further restricts the extent of flooding to further reduce the overall message overhead per connection. In LSP, an OSPF-like (link state) algorithm is assumed to be the routing protocol. LSP uses the MRT-2 routing table and the link-state table to forward a probe to its neighbors, which satisfy the QoS metrics requested and locate a feasible path to the destination node. Each node is required to collect the QoS information for links that lie within a user-defined radius given by *HOPARAM (hop parameter)*, H for generating and updating its MRT-2 routing table. Such a node is also referred to as the root of the LSD, root node. The root node collects the state information about its associated LSD periodically, i.e., the root node maintains a database of the link status for its LSD. The state information can be assumed accurate if the update frequency is higher than the rate of change.

In LSP, the probe-forwarding process is handled based on the distance between the root node and the destination node, i.e., whether the destination node can or cannot be found within the LSD. If the destination node is within the LSD, then the probe is routed to the destination node. If the destination node is located outside the LSD, then the probe is routed to the "edge" of the LSD. The root node uses the MRT-2 routing table to discover all alternative paths that lead to the "edge" node. From the link-state table, it can learn about the links within the LSD. Multiple copies of the probe are routed towards those edge nodes where at least one path exists from the root node to the "edge" nodes that also satisfy the QoS metrics requirement.

In LSP, flooding does not occur if at least one path extends towards the edge nodes with the given QoS requirement. If no path is found from the LSD state database, the root or intermediate node floods its neighbors with probes, regardless of whether the neighbor is in the routing entry for the destination. In addition, a probe will only be forwarded if the QoS-constraint is fulfilled and the next hop is the destination or intermediate destination node. The connection is established when the source node receives the ACK message. The acknowledgement and connection failure-handling phase is also implemented in the same way as in the DF algorithm.

The generic algorithm of LSP for a connection request with destination node t, and requested QoS metrics qos and a root node or intermediate node i is shown below:

```
LSPfwd($t, qos$) at node $i$:

1. if $t$'s distance $\leq$ HOPARAM then
2.     for each list of nodes, $\ell$ to $t$
3.         if never sent probe to $\ell_1$ before then
```

```
4.          if $m(i, \ell_1, \ell_2, \ldots, \ell_h)$ $C$ $qos$ then
5.            send probe$[s, t, cid, i, 1, qos, hop + 1]$ to $\ell_1$
6.          end if
7.        end if
8.      end for
9.  else
10.     for every edge node, $e$ to $t$
11.       for each list of nodes, $\ell$ to $e$
12.         if never sent probe to $\ell_{e1}$ before then
13.           if $m(i, \ell_{e1}, \ell_{e2}, \ldots, \ell_{eh})$ $C$ $qos)$ then
14.             send probe$[s, t, cid, i, 1, qos, hop + 1]$ to $\ell_{e1}$
15.           end if
16.         end if
17.       end for
18.     end for
19. end if
```

The variables ℓ and ℓ_e are alternative paths and h is their length. If t is located HOPARAM hop count away from i, LSP refers to the MRT-2 routing table for the LSD edge nodes, e which leads to t. Again, LSP looks up the MRT-2 routing for a list of nodes, ℓ_e, which lead to each edge node, e. LSP only forwards probes to the neighboring nodes, ℓ_{e1} if the expression $m(i, \ell_{e1}, \ell_{e2}, \ldots, \ell_{eh})$ C qos is evaluated and returned true.

On the other hand, if t is located within the LSD, LSP looks up the MRT-2 routing for a list of nodes, ℓ, which lead to t. LSP only forwards probes to the neighboring nodes, ℓ_1 if the expression $m(i, \ell_1, \ell_2, \ldots, \ell_h)$ C qos is evaluated and returned true.

4.5.3 Limited Scope Hop-by-Hop Probing (LSHP)

The LSHP is very similar to LSP. The LSP performs source routing within the LSD, i.e., the root determines the path for the probe to travel towards an edge or destination node by evaluating the link-state information. The alternative paths and their corresponding link-state information are made available by the MRT-2 routing table and link-state table. In LSHP, each intermediate node is required to check whether the next node in the alternative path(s) has sufficient resources to admit the required QoS.

For a given destination node, each alternative path chosen is re-validated by each intermediate node that performs independent routing. Unlike LSP, LSHP will not look ahead for the link-state information of all the nodes in each alternate path, but will examine only the next hop in each path. If the root or an intermediate node cannot find any feasible paths that satisfy the given QoS requirement, it then attempts to send the probe to all its neighbors that satisfy the QoS requirement. The ACK phase and connection failure handling used are similar to DF.

The generic algorithm of LSHP for a connection request with destination node t, and requested QoS metrics qos, and a root node or intermediate node i is shown below:

```
LSHPfwd($t, qos$) at node $i$:

1. if $t$'s distance $\leq$ HOPARAM then
2.    for each list of nodes, $\ell$ to $t$
3.       if never sent probe to $\ell_1$ before then
```

```
4.          if $m(i, \ell_1)$ $C$ $qos$ then
5.            send probe$[s, t, cid, i, 1, qos, hop + 1]$ to $\ell_1$
6.          end if
7.        end if
8.    end for
9. else
10.   for every edge node, $e$ to $t$
11.     for each list of node, $\ell$ to $e$
12.       if never sent probe to $\ell_{e1}$ before then
13.         if $m(i, \ell_{e1})$ $C$ $qos$ then
14.           send probe$[s, t, cid, i, 1, qos, hop + 1]$ to $\ell_{e1}$
15.         end if
16.       end if
17.     end for
18.   end for
19. end if
```

The variables ℓ or ℓ_e are alternate paths.

If t is located HOPARAM hop count away from i, LSHP refers to the MRT-2 routing table for the LSD edge nodes, e, which lead to t. Again LSHP looks up the MRT-2 routing for a list of nodes, ℓ_e, which lead to each edge node, e. LSHP only forwards probes to the neighboring node, ℓ_{e1} if the expression $m(i, \ell_{e1})$ C qos is evaluated and returned true.

On the other hand, if t is located within the LSD. LSHP looks up the MRT-2 routing for a list of nodes, ℓ, which lead to t. LSHP only forwards probes to the neighboring node, ℓ_1 if the expression $m(i, \ell_1)$ C qos is evaluated and returned true.

4.6 Distributed QoS-based Routing Framework

A distributed generic QoS-based routing framework for DF, LSP, and LSHP at node i is presented below. The framework was first introduced in [4] and serves as a reference for implementing the simulation program used in this thesis.

```
1.  for (;;)
2.    wait for message $[s, t, cid, k, type, qos, hop]$
3.    switch (message.$type$)
4.    case 1: /*probe*/
5.      if $i == t$ then
6.        send ack$[t, s, cid, i, 2, qos, 0]$ to $k$
7.      else
8.        if never forwarded probe to $t$ before then
9.          DFfwd$(t, qos)$ or LSHPfwd$(t, qos)$ or LSPfwd$(t, qos)$
10.       else
11.         record probe if probe $hop \leq$ the first probe
12.       end if
13.     end if
14.     break;
15.   case 2: /* acknowledgement message */
16.     if node $i$ has the required resources then
17.       reserve the $qos$ for $k$
18.       if $i == t$ then
19.         connection established
20.       else
```

```
21.           search probe table for the node $h$ that forwarded probe $cid$
22.           send ack$[s, t, cid, i, 2, qos, hop+1]$ to $h$
23.         end if
24.       else
25.         send fail$[t, s, cid, i, 3, qos, hop-1]$ to $k$
26.       end if
27.       break;
28.   case 3: /* fail message */
29.       search probe table for the node $h$ that forwarded probe $cid$
30.       if $(h \neq k) \wedge (h$ satisfies the request constraint$)$ then
31.         send ack$[s, t, cid, i, 2, qos, hop+1]$ to $h$
32.       else
33.         release the $qos$ reserved for $x$
34.         send fail$[t, s, cid, i, 3, qos, hop-1]$ to $x$
35.       end if
36.       break;
37. end for
```

According to the framework, the node i waits for a message to arrive. For a received message $[s, t, cid, k, type, qos, hop]$ being a probe, if the probe is designated to node i, where i matched t, node i immediately sends an ACK message to node k, which is the sender of this probe and reserves the qos QoS metrics requested. If the probe is not designated to node i and i has never forwarded the probe before, a DF, LSHP, or LSP QoS-based routing algorithm is used to forward the probe. If node i has forwarded the probe with cid session identification before, the information carried by the probe is recorded if and only if the probe hop count is equal to or less than the first probe.

When an ACK message $[s, t, cid, k, type, qos, hop]$ is received at node i, the connection is successfully established. If node i is not the designated node, try every probe *cid* table entry for a node h, where the *qos* QoS metrics requested can be reserved. Node h is also one of the senders of probe *cid*. If there is no such node h that can satisfy the QoS requested then a fail message is sent back to the sender of this ACK, k. If such a node h exists, then the *qos* QoS metrics requested are reserved and an ACK message is sent to node h.

When a fail message $[s, t, cid, k, type, qos, hop]$ is received at node i, the node i tries every probe *cid* table entry for a node h, where the *qos* QoS metrics requested can be reserved. Node h is a sender of probe *cid* and must not be node k, which sends an i fail message. An ACK message is sent to node h if such a node exists. If such a node h does not exist then i sends a fail message back to node x, where node x is the node which sends ACK to node i. Meanwhile, the resources reserved to node k are released. If t from the fail message matches node i, the connection fails and the connection request is rejected.

4.7 Routing Table Lookup Complexity Analysis

The complexity of the route look up of both LSP and LSHP was also studied. The level of complexity also serves as a measure of operational cost of a given node.

4.7.1 Route lookup complexity of LSHP

Let n be the number of entries of the routing table. Each entry has at most b number of alternative paths or at most k number of edge nodes. The corresponding link-states table will also have n number of entries. For a hop count H of HOPARAM setting, if the destination node is within the LSD, then the route lookup complexity is $O(n(b+1))$. If the destination node is located outside of the LSD, then the lookup complexity is $O(n(kb+1))$. The route lookup complexity can be reduced by implementing the following optimization techniques:

- Implement edge nodes as pointers pointing to its routing entry respectively to eliminate the k term in the route lookup complexity.
- Implement pointers to the appropriate link-state table entries in each entry of the routing table to eliminate the b term in the route complexity.

4.7.2 Route lookup complexity of LSP

Let n be again the number of entries of the routing table. Each entry has at most b number of alternative paths or at most k number of edge nodes. Each alternative path consists of at most H number of nodes. The corresponding link-states table also has n number of entries. For a H hop count of HOPARAM setting, if the destination node is within the LSD, then the route lookup complexity is $O(n(Hb+1))$. If the destination node is located outside of the LSD, then the lookup complexity is $O(n(kHb+1))$. The optimization techniques used in LSHP can also be used in LSP to eliminate term k and H by implementing pointers to the appropriate link state table entries for each alternative path.

The route look up complexity presented could be greatly reduced if the optimization method was deployed. Notice that the routing table lookup complexity of DF algorithm is not discussed, because the DF algorithm lookup complexity is equivalent to the LSHP lookup order when the distance t is less than or equal to HOPARAM, which is $O(n(b+1))$.

4.8 Optimization

Each node in the network is allowed to forward at most one probe to each of its neighbors. Such policy prevents the probe from traveling in a loop formed when nodes forward the same probe at least once to the same next hop. We decided to record the information carried by the probe, such as hop count, QoS requested, QoS metric collected so far, the sender of the probe and the QoS metric negotiated into the node. This ensures that the probes forwarded are smaller and have a fixed size. Probes with smaller and fixed size have a better chance to fit into a queue and thereby lower the transmission time from node to node. The timeout method is again used in governing the information recorded by the probe. If the node does not receive an ACK message *cid* matching the probe *cid* in a period of time set to timeout, the node erases the record.

To increase the chance of establishing the connection during the acknowledgement phase, every node in the network is required to record the information carried by probes that arrived later before discarding it. This way, whenever a node is unable to reserve the QoS metrics negotiated by the first probe, the node can now send the ACK message to the next sender by looking up the probe table. This also enables the ACK message to be backtracked to the node, which sends this ACK and directs the node to try another path if available. In the worst case, the ACK message may be traced back to the destination node and the connection is rejected.

A probe of connection request with subjective QoS metrics, which arrives later, is required to be recorded in the node only if the probe hop count is less than or equal to the first probe arrived. On the other hand, a probe of connection request with connective QoS metrics is required to be recorded only if the total collected QoS metrics is less than or equal to total QoS metrics collected by the first probe. Notice that the hop count parameter is not important for the connective QoS metrics, because a probe with less hop count may not have the superior total collected connective types of QoS metrics when compared to the first probe. In most of the simulation, probes that arrive later often have smaller hop counts but with inferior connective QoS metrics collected. This may cause the connection to fail in the acknowledgement phase if the information carried from the later probe is used.

The MRT-2 routing table does not consider a "stretching in" path as an alternative path. An example of a "stretching in" path is a path that leads to a node z with K hop from a root node, but node z is also located R hop count away from the same root node through another path. It is given that K is greater than R and greater than HOPARAM in terms of hop count and R is less than or equal to HOPARAM. This is to prevent a loop formation in the alternative path should the length of the alternative path be greater than HOPARAM -1. This is also why we allowed the LSP and LSHP algorithms to send probes to the neighbors that satisfy the QoS metric(s) requested, when there is no feasible path found in the MRT-2 routing table and link-state table. Flooding the probe to neighbor nodes increases its chance of discovering the "stretching in" path with hope that the resources that were unavailable earlier are available now.

Chapter 5

Simulation

5.1 Overview

Simulation studies compared the performance of the proposed algorithms with selective probing [4]. The simulator implemented in this thesis was written using the PARSEC simulation language [2]. The simulations were performed using two topologies provided in [4] and four topologies that were generated using the Tiers Internet topology generator [9]. Each network link is full duplex with a bandwidth capacity of 155Mbps and link delay of a random value from 0.0 to 125.0ms. The QoS metric considered in this simulation is bandwidth only. The random bandwidth requests generated were uniformly distributed between 64Kbps to 1.5Mbps.

5.2 Performance Measurement Model

The algorithms were evaluated using two parameters: *acceptance ratio*

$$AR = \frac{\sum \text{Accepted connection}}{\sum \text{Number of simulation}} \tag{5.1}$$

and *average number of probes*

$$ANP = \frac{\sum \text{Number of probes sent}}{\sum \text{Number of simulation}}. \tag{5.2}$$

The acceptance ratio is computed by dividing the total number of successful connections made by the algorithm by the total number of connections requested. If the acceptance ratio is 1.0, then all requests for connections are successfully made. The results reported were obtained by taking the average of repeated simulation runs of 1000 times. The ANP denotes the average number of probes across the different runs of the simulations. The simulation studies observed the above two parameters while varying the following parameters: *network size*, *network state variation* (NSV), and *average bandwidth load* (ABL). The NSV gives the percentage by which the network state (i.e., the different link states) can vary from the probing to the acknowledgement phases. This parameter captures the dynamics in the network due to background traffic.

5.3 Results and Discussion

Six different network sizes were used in this simulation study. The first two, as shown in **Figure 5.1**, are similar to the two topologies NET-1 and NET-2 in [4]. The remaining four graphs were generated by the Tiers topology generator with the number of nodes equal to 20, 50, 151, and 205, and they are shown in **Figure 5.2**,

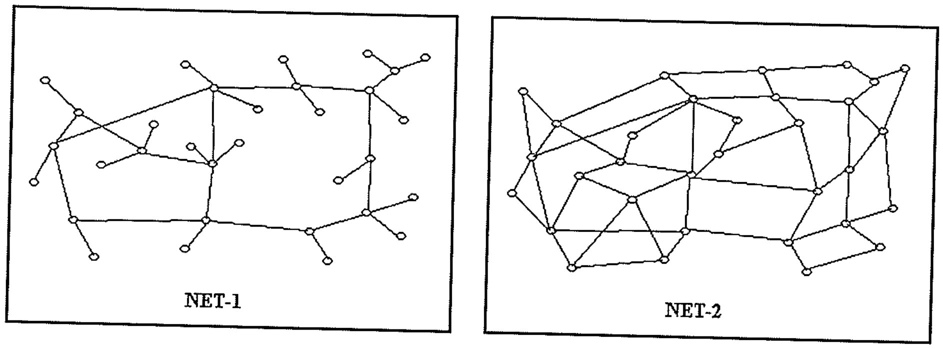

Figure 5.1: Two Internet test topologies.

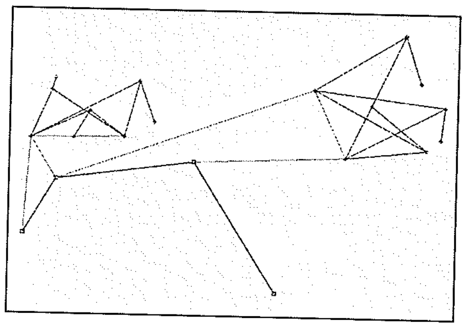

Figure 5.2: Internet topology with 20 nodes generated from Tiers program.

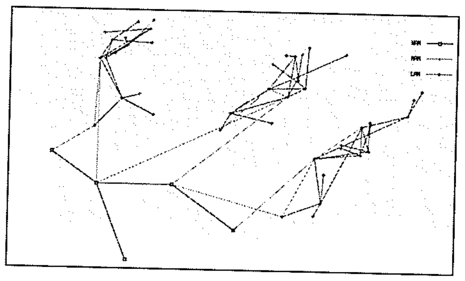

Figure 5.3: Internet topology with 50 nodes generated from Tiers program.

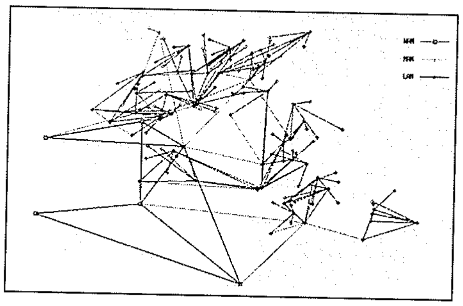

Figure 5.4: Internet topology with 151 nodes generated from Tiers program.

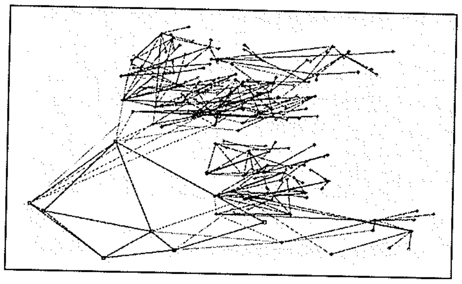

Figure 5.5: Internet topology with 205 nodes generated from Tiers program.

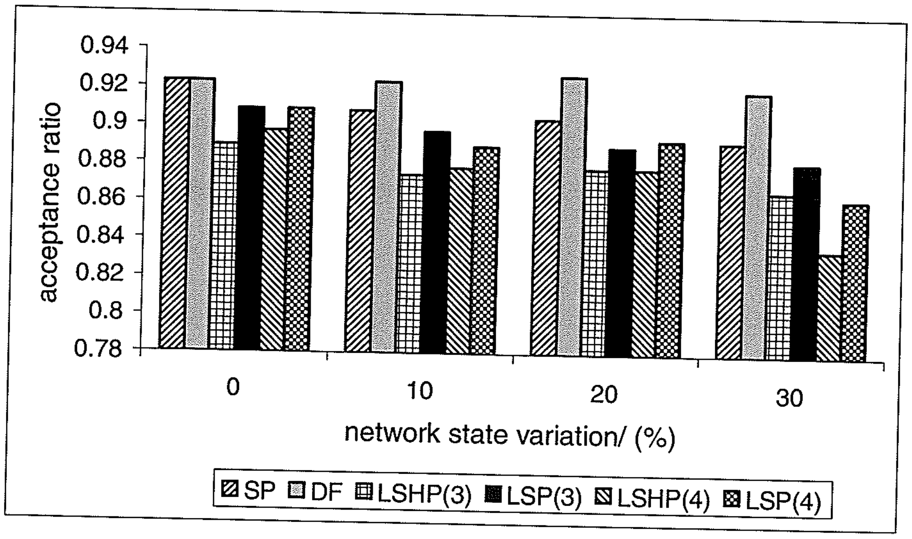

Figure 5.6: Acceptance ratio versus network state variation for a 151-nodes graph with 143 Mb/s average bandwidth load.

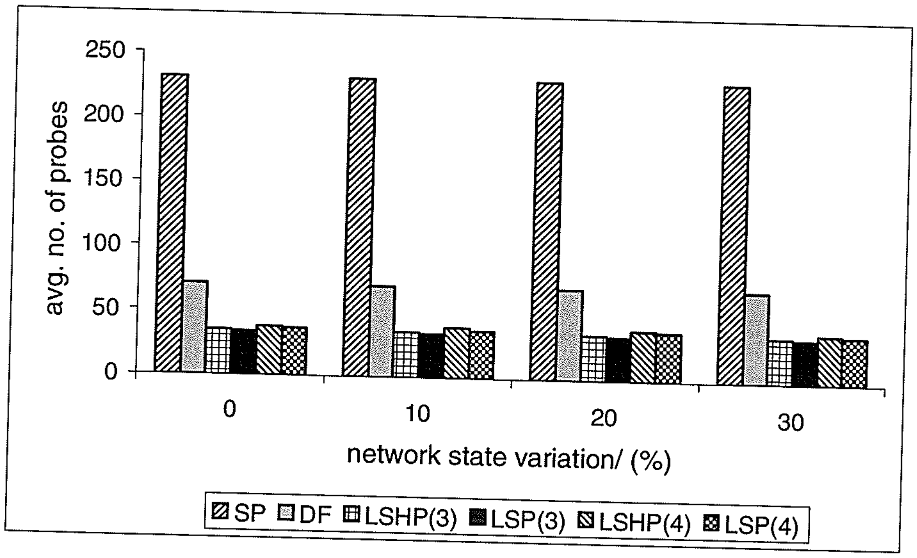

Figure 5.7: Average number of probes versus network state variation for 151-nodes graph with a 143 Mb/s bandwidth load.

Figure 5.3, **Figure 5.4**, and **Figure 5.5** respectively. **Figure 5.6** shows the variation of the AR with NSV for a 151-node graph with ABL set at 143 Mb/s. We assumed that the links have 155 Mb/s capacities. **Figure 5.7** shows the variation of the ANP with the NSV for the same 151-node graph with ABL set at 143 Mb/s. As expected, the ANP does not change with the NSV and the AR decreases with increasing NSV. Interestingly, the DF algorithm is least affected by the changes in NSV.

When a connection request is sent from a destination towards the source to initiate the routing process, the destination starts a timer. The timer expires after a predefined interval. The connection request fails if the destination does not receive a positive confirmation before the timer expires. Because the DF algorithm tries only the feasible paths, if multiples paths exist between the source and destination the DF may deliver the probes to the destination faster than the other

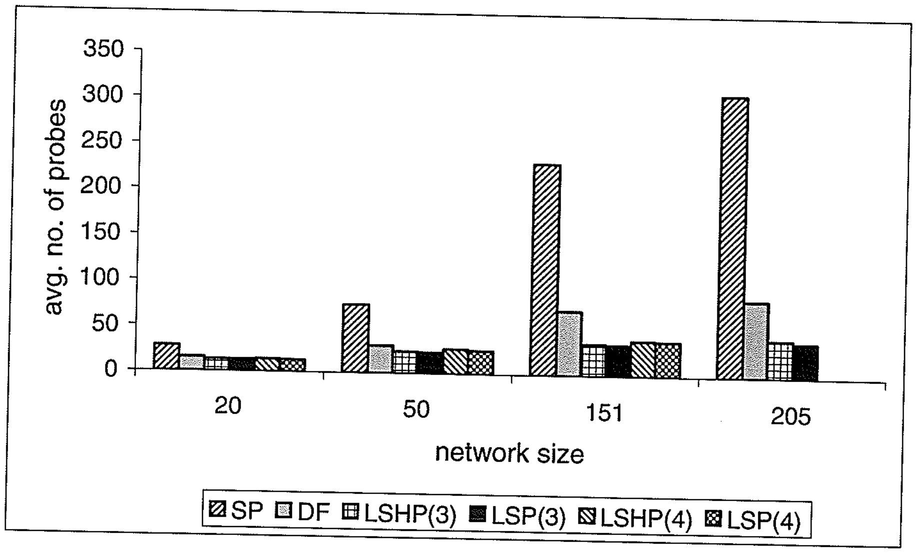

Figure 5.8: Average number of probes versus network size with 143 Mb/s bandwidth load and network state variation of 20%.

algorithms. Therefore, the DF has more opportunities to try all possible paths that exist between the source and destination before the timer expires. This results the in DF having lower degradation in the overall acceptance ratio as the NSV increases compared to the other algorithms.

Figure 5.8 shows the variation of the ANP with the network size with a 143 Mb/s ABL and NSV of 20%. As the network size increases, the ANP increases for all the algorithms. However, the rate of increase with the network size is highest for SP and lowest for LSP with HOPARAM= 3. Because SP uses only the QoS requirement as the selection criteria for the next hop, it will also send probes along infeasible directions, thereby causing a significant increase in ANP for larger networks. The LSP has a "HOPARAM-wide look-ahead" for the QoS requirements; therefore, it is very cautious in sending the probes. The resulting LSP has the least ANP.

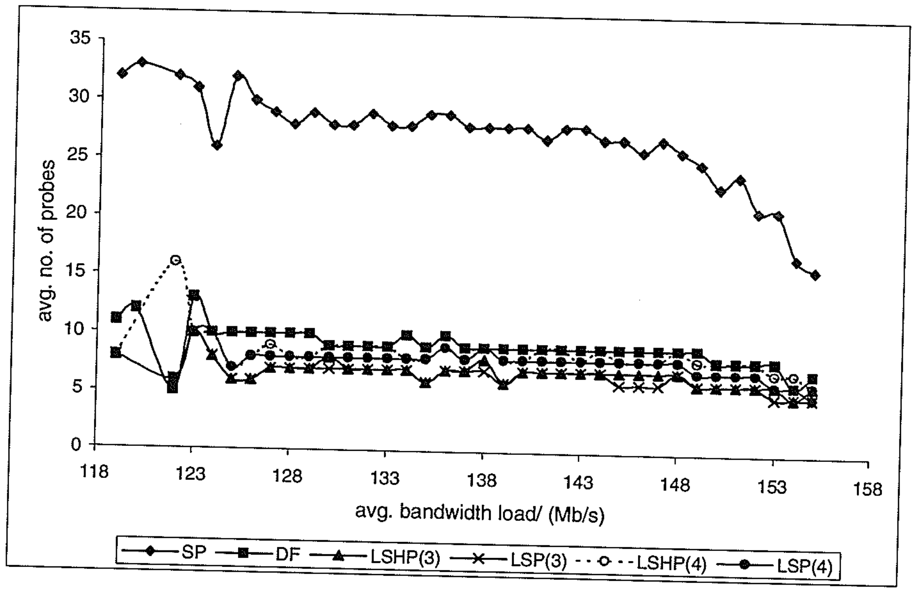

Figure 5.9: Average number of probes versus average bandwidth load for Net-1 with 20% network state variation.

Figure 5.9 shows the variation of the ANP with the ABL for Net-1 (shown in **Figure 5.1**) with 20% NSV. The variation of AR with ABL for the same network and NSV is shown in **Figure 5.10**. As mentioned earlier, the links in the network have a maximum capacity of 155 Mb/s. As it can be observed in **Figure 5.9**, the ANP decreases with increasing ABL. This is because all the algorithms first check the QoS requirement and as ABL increases, the links would not have enough capacity to admit the probes. The LSP with HOPARAM= 3 out-performs all the other algorithms in terms of ANP and performs similarly to the others in terms of AR. It is worth noting that LSP with HOPARAM= 3 has less message overhead than the LSP with HOPARAM= 4. This may be because with a larger look-ahead the LSP may more often decide that there is no single feasible path. This triggers the algorithm to flood and causes an increase in message overhead. The same effect can be observed in **Figure 5.11** and **Figure 5.12**. The **Figure 5.11** shows the

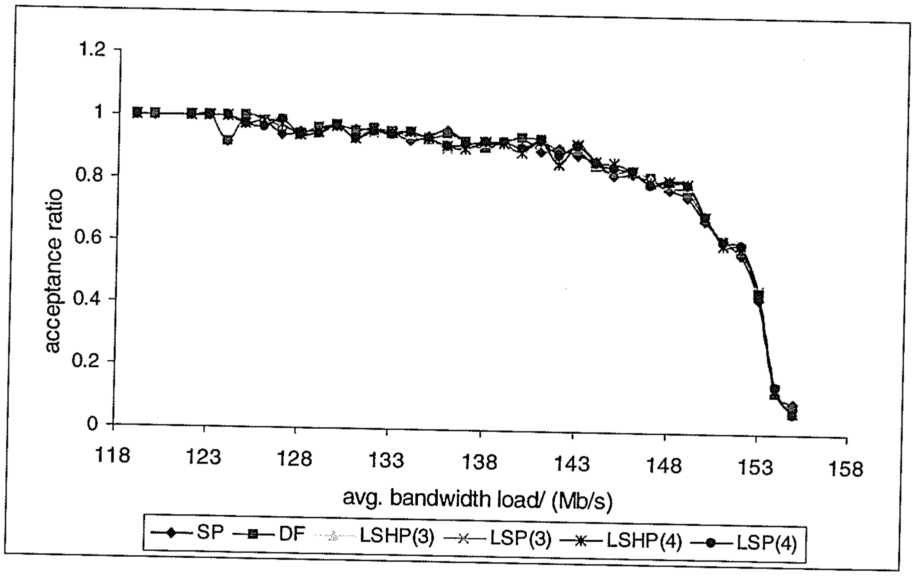

Figure 5.10: Acceptance ratio versus average bandwidth load for Net-1 with 20% network state variation.

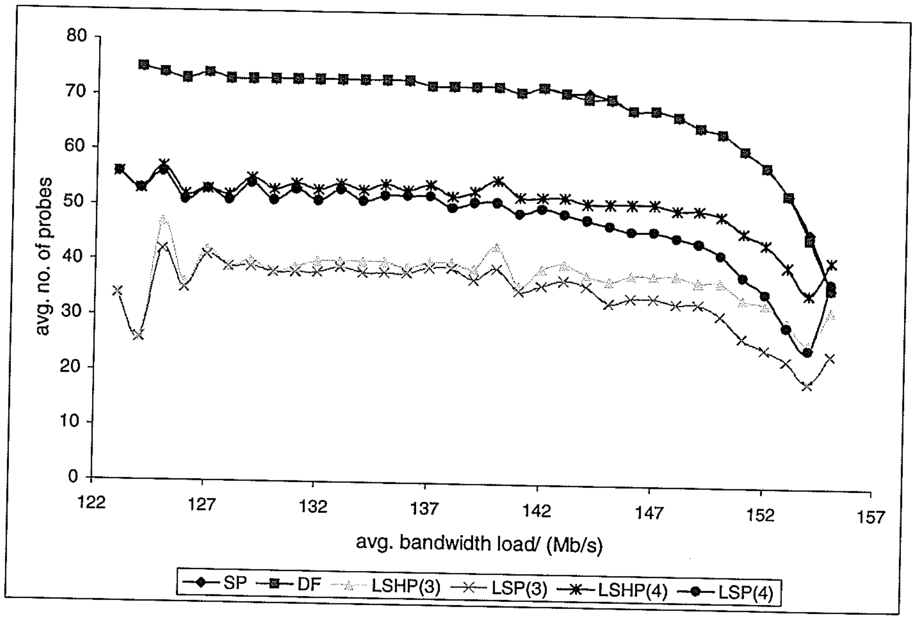

Figure 5.11: Average number of probes versus average bandwidth load for Net-2 with network state variation of 20%.

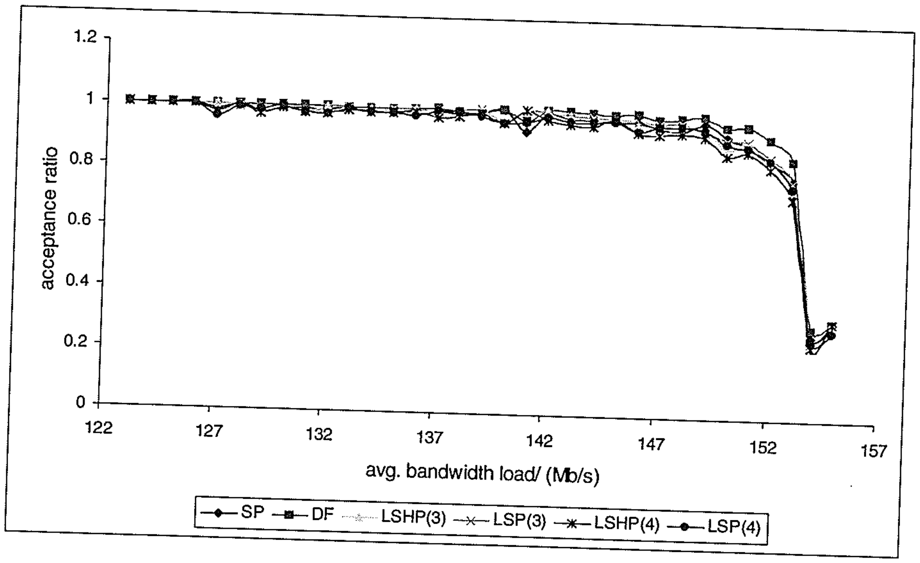

Figure 5.12: Acceptance ratio versus average bandwidth load for Net-2 with network status variation of 20%.

variation of ANP with ABL and **Figure 5.12** shows the variation of AR with ABL for Net-2 with NSV at 20%.

Figure 5.13 shows the variation of ANP with ABL for a 151-node graph with NSV of 20% and **Figure 5.14** shows the corresponding AR versus ABL variation. The graph is generated using the Tiers program. The graphs generated by the Tiers do not have multiple paths to the extent seen in Net-2 in **Figure 5.1**. For this reason, the different variations of the limited scope algorithm do not perform differently with respect to ANP.

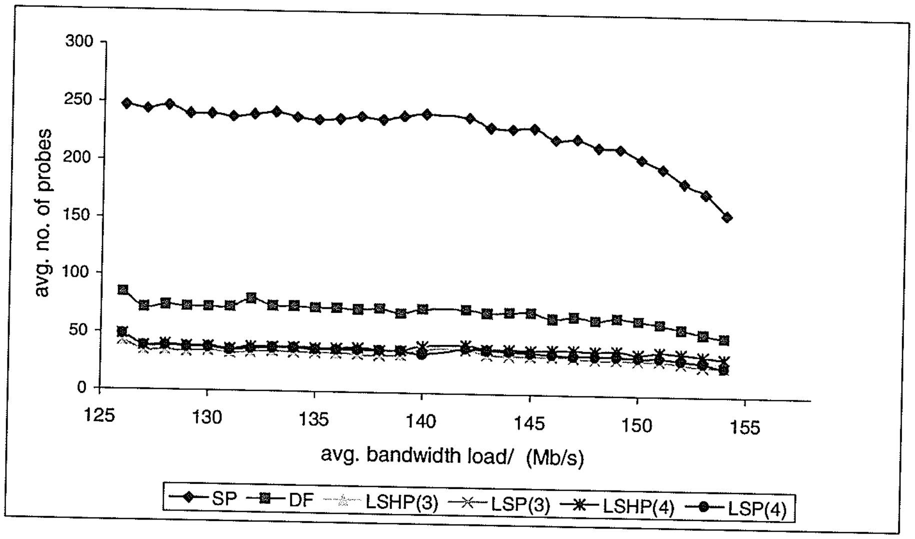

Figure 5.13: Average number of probes versus average bandwidth load for 151-nodes graph with network state variation of 20%.

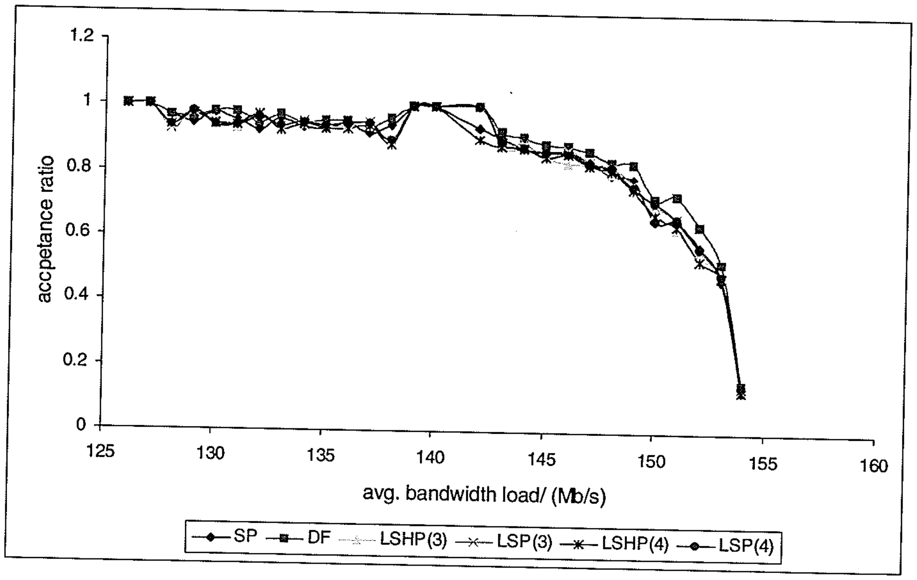

Figure 5.14: Acceptance ratio versus average bandwidth load for 151-nodes graph with network state variation of 20%.

5.4 Summary

QoS-based routing algorithms are refined with every new finding and innovative of routing methods. Considering all the available types of routing algorithms, distributed routing algorithms are more attractive than the rest due to their scalability, robustness, and flexibility in adapting to new network environment changes and new QoS metrics requirements. However, the common problem in distributed QoS-based routing is the high degree of overall network message overhead. Advantages of source routing (The ability to calculate and determine the paths a probe should travel) can be added to that of the distributed routing algorithm by just redefining the meaning of "local state". This thesis successfully redefines the "local states" as a combination but not aggregated network state of a group of nodes with a hop count parameter of H from a node under consideration. By adjusting the value of H, the degree network state precision can be increased. That is the main factor for LSP and LSHP out performing other distributed algorithms in terms of performance.

Chapter 6

Conclusion and Future Work

In any deployment of QoS-based routing schemes, we can expect these schemes to handle only a small percent of the overall traffic. The rest would be handled in a best-effort manner [17]. Therefore, it is very important for the QoS-based routing schemes to be very efficient in resource usage, i.e., the QoS-routing schemes should not impose a significant overhead for their infrequent usage. In this thesis, we proposed three QoS-based routing algorithms to achieve this objective. Extensive simulation studies were performed to evaluate the performance of the proposed algorithms with an existing QoS-based routing algorithm. Results of the simulations showed that our approaches cause significantly lower message overhead compared with SP while maintaining a similar acceptance ratio.

The proposed unicast QoS-based routing algorithms can be easily modified to support multicast QoS-based algorithms by substituting the current single destination node address to multiple or matching prefix destination node addresses. When a client needs to join the existing multicast tree, a request message with the QoS requirements is sent from the source node and the source node starts the probing phase again to find a path to join the new client to the existing tree. Another approach is to have a new client to join the existing tree where any nodes, m, in the existing multicast tree can act as source nodes. When node m sees a request message, it will not forward the request message to the actual source nodes if the request QoS service type and identification match one of its currently supporting services. Node m may act as the source node and start the probing phase to find a path to the new client that satisfies the QoS requirements.

The MRT-2 routing table can be improved to include a "stretching in" path limiting the length of the path to at most HOPARAM-1. The MRT-2 table with the "stretching in" path will have the knowledge of all feasible paths to any node with its LSD. With this knowledge there is no need to flood the neighboring nodes with probes when there is no feasible path found in the new MRT-2 routing table and link-state table. We believe this will further reduce the average number of probes sent per connection without negatively affecting with LSP and LSHP performance.

Bibliography

[1] Tal Anker, Danny Dolev, Yigal Eliaspur. "Off-Piste QoS-aware Routing Protocol," in Proceedings of 12th IEEE International Conference on Network Protocols, Computer Society, Berlin, Germany, Oct.2004, pp. 24-35

[2] R. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, B. Park, and H. Song, "PARSEC: A Parallel Simulation Environment for Complex Systems," IEEE Computer, vol. 31, no. 10, Oct. 1998, pp. 77-85.

[3] J. Chen, P. Druschel, and D. Subramanian, "A Simple, Practical Distributed Multi-Path Routing Algorithm," Technical Report TR98-320, Rice University, July 1998.

[4] S. Chen and K. Nahrstedt, "Distributed Quality-of-Service Routing in High Speed Networks Based on Selective Probing," Annual Conference on Local Area Networks (LCN '98), Oct. 1998, pp. 80-89.

[5] Shigang Chen and Klara Nahrstedt, "Distributed QoS Routing with Imprecise State Information," in Proceedings of 7th IEEE International Conference on Computer, Communications and Networks, Lafayette, LA, Oct. 1998, pp. 614–621.

[6] S. Chen and K. Nahrstedt, "An Overview of Quality-of-Service routing for the Next Generation High-Speed Networks: Problems and Solutions," IEEE Network Magazine; Special Issue on Transmission and Distribution of Digital Video, vol. 12, no. 6, Nov.-Dec. 1998, pp. 64-79.

[7] G. Cheng, Y. Tian and N. Ansari, "A New QoS Routing Framework for Solving MCP," Special Issue on Internet Technology, IEICE Trans. on Communications, vol. E86-B, no. 2, pp. 534-541, Feb. 2003.

[8] J. Crowcroft, M. Handley, and I. Wakeman, "Internetworking Multimedia," Morgan Kaufmann, San Francisco, CA, 1999.

[9] M. Doar, "A Better Model for Generating Test Networks," IEEE Global Telecommunications Conference (Globecom '96), Nov. 1996.

[10] D. Durham and R. Yavatkar, "Inside the Internet's Resource reSerVation Protocol: Foundations for Quality of Service," John Wiley, New York, NY, 1999.

[11] M. Gerla, X. Hong, and G. Pei, "Landmark Routing for Large Ad Hoc Wireless Network," In Proceeding of IEEE GLOBECOM 2000, Nov. 2000.

[12] K. Kilkki, "Differentiated Services for the Internet," Macmillan Technical Publishing, 1999.

[13] Seok-kyu Kweon and Kang G. Shin; "Distributed QoS Routing with Bounded Flooding for Real-Time Applications," In Proceedings of IEEE GLOBECOM, November 1997.

[14] W. Liu, W. Lou and Y. Fang, "An Efficient Quality of Service Routing Algorithm for Delay Sensitive Applications," *Computer Networks* , vol.47, pp. 87-104, 2004.

[15] G. Manimaran, H. S. Rahul, and C. S. R. Murthy, "A New Distributed Route Selection Approach for Channel Establishment in Real-Time Networks," IEEE/ACM Transactions on Networking, vol. 7, no. 5, 1999, pp. 698-709.

[16] J. T. Moy, "OSPF: Anatomy of an Internet Routing Protocol," Addison-Wesley, Reading, MA, 1998.

[17] W. Stallings, "High-Speed Networks: TCP/IP and ATM Design Principles," Prentice-Hall, New Jersey, 1998.

[18] Q. Sun, H. Langendorfer, "A New Distributed Routing Algorithm for Delay-Sensitive Application," Computer Communications 21 (6) (1998) 572578.

[19] R. Widyono, "The Design and Evaluation of Routing Algorithms for Real-Time Channels," Technical Report ICSI TR-94-024, International Computer Science Institute, UC, Berkeley, June 1994.

[20] Kaixin Xu, Ken Tang, Rajive Bagrodia, Mario Gerla, Michael Bereschinsky; "Adaptive Bandwidth Management and QoS Provisioning in Large Scale Ad Hoc Networks," IEEE MILCOM 03, Boston, Massachusetts, Oct. 2003.

[21] Xin Yuan, "Heuristic Algorithms for Multiconstrained Quality of Service Routing," IEEE/ACM Trans. on Networking, vol. 10, issue. 2, pp. 244–256, Apr. 2002.

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