Feedback Mechanism Validation and Path Query Messages in Label Distribution Protocol

Ahmed Gario

A Thesis

in

The Department

of

Computer Science

Presented in partial Fulfillment of the Requirements

For the Degree of Master of Computer Science at

Concordia University

Montreal, Quebec, Canada

April 2003

ABSTRACT

Feedback Mechanism Validation and Path Query Messages in Label Distribution Protocol

Ahmed Gario

In constraint based routing a topology database is maintained on all participating nodes to be used in calculating a path through the network. This database contains a list of the links in the network and the set of constraints the links can meet. Since these constraints change rapidly, the topology database will not be consistent with respect to the real network. A feedback mechanism was proposed by Ashwood-Smith, et al, to help correct the errors in the database. It behaves like a depth first search, and is meant to be useable only when the database sees the availability of resource more than there really are. In this mechanism, the source node can learn from the successes or failures of its path selections by receiving feedback from the path it is attempting. The received information is used in the subsequent path calculations.

We validated the feedback algorithm to see how it behaves in all database situations, and found out that the feedback algorithm was helpful in all cases not only when it was optimistic. We also propose adding query messages to make the feedback algorithm behave more like breadth first search. The path query messages algorithm reduces the retry attempts in setting up a path, and also utilizes the network by gathering much more information about the resources.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor Dr. J.W. Atwood for his unlimited precious guidance, motivation and direction throughout my research work and in the preparation of this thesis.

I would also like to thank my friends and colleagues who helped me in many ways.

Finally, words can't express my deepest appreciation to family for their unlimited help, support and continuous encouragement.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Glossary

CR-LDP	Constraint Routing Label Distribution Protocol
CR-LSP	Constraint Routing Label Switching Path
DSCP	DivServ Code Point
ER	Explicit Routing
FEC	Forwarding Equivalent Class
GT-ITM	Georgia Tech Internetwork Topology Models
IETF	Internet Engineering Task Force
IS-IS	Intermediate System to Intermediate System
LDP	Label Distribution Protocol
LSP	Label Switching Path
LSR	Label Switching Router
MPLS	Mulitprotocol Label Switching
OSPF	Open Shortest Path First
QoS	Quality of Services
RSVP	Resource ReSerVation protocol
SLA	Service Level Agreement
TE	Traffic Engineering
TLV	Type Length Value
TOS	Type Of Services

CHAPTER 1

Introduction

Multiprotocol Label switching (MPLS) network is meant to provide traffic engineering and better Quality of Services (QoS) due to its property that the source node may know the complete path the flow will traverse. To allocate the appropriate path that satisfies some constraints in an MPLS network, the ingress node should not only know the possible paths to reach any destination, but also know concrete information about the availability of the resources along these paths. Because the network is a distributed and dynamic system, the availability of those resources is subject to rapid changes. So it is necessary to have a mechanism to advertise information about links to all nodes in the network [1], [2]. The flooding mechanism proposed in [1] is one of these advertising mechanisms. In this mechanism the information is exchanged among the nodes in the network in a time interval. Every node periodically sends its state to the other nodes in the network. The nodes in the network will have the same view of the network at the flooding time. Within that interval every node in the network, regarding the actual resources in the network, is often either:

- Optimistic, when it sees more available resources than there really are. It becomes optimistic when the amount of resources in the network the

arriving requests reserve is much more than the amount of resources the departing ones release, or

- Pessimistic, when it sees less available resources than there really are. It becomes pessimistic when the amount of resources in the network the arriving requests reserve is less than the amount of resources the departing ones release.

Dealing with changing constraints is a major problem for a constraint based routing system. In a constraint based routing algorithm, however, a path from one node to another is calculated based on one or more of these constraints. Since the availability of the resources is changing, and the knowledge about these constraints often out-of-date regarding the reality, there must be a better way than the flooding to inform a node as soon as possible. Obviously, bandwidth is one of the essential constraints, and it is rapidly changing, but the flooding algorithm is not an efficient resource advertising algorithm that keeps the node up-to-date. That is due to the low flooding frequency compared to the resource change frequency. In smaller networks, one can resort to higher frequency flooding, but in a large network this obviously is not scalable [3].

The IETF draft "Improving Topology Data Base Accuracy with Label Switched Path Feedback in Constraint Based Label Distribution Protocol" [3] proposes adding to the node a topology database to build a map of the network, and

algorithms to deal with optimistic databases without resorting to shorter flooding intervals. The draft says its algorithm deals only with optimistic databases. This is theoretically correct and logical. It also describes the setting up of a path algorithm behavior as a depth-first search algorithm.

We are going to see how the label distribution protocol (LDP) with feedback algorithm deals with pessimistic, optimistic and up-to-date databases. We are also going to examine the effect on the path availability, and what the time taken to set a path up is if we make the LDP with feedback algorithm behave very much like breadth-first search instead of depth-first search. By breadth-first search we mean that query messages traverse through the next shortest paths the node would take in case of failure, if any, at the same time as the label request message. These query messages are to return feedback from the paths they traverse. This feedback is useful in case of either success or failure of the label request message. In addition to finding the path earlier, it helps to improve the topology database.

CHAPTER 2

MultiProtocol Label Switching (MPLS)

MPLS stands for "Multiprotocol" Label Switching. It was invented in 1997 by the Internet Engineering Task Force (IETF), named Multiprotocol because it is capable of working with any network layer protocol. MPLS is also known as a 2.5 layer protocol because it has the capabilities of both layer two and layer three. It has the flexibility of IP, layer three, routing; and the efficiency of link level, layer two, switching. MPLS networks carry traffic on virtual connections called "label switched paths". A label switched path is the path which packets having the same label follow. The label is a short length number that does not include any network layer address. It carries information that uniquely identifies the Forwarding Equivalent Class (FEC) encapsulated within the MPLS packet. A label is inserted in the packet's header by the ingress label switching router (LSR) as it arrives at the MPLS network region, prior to forwarding it through the network, and it is removed by the egress label switching router (LSR) when the packet departs from the MPLS domain [4]. Two label switching routers (LSR) may have more than one label between them. In other words, labels are associated with the flows rather than a router itself.

Every router in a connectionless network layer protocol analyzes every traveling packet's header and makes an independent forwarding decision for that packet

to the next router based on the analysis of the packet header and the information in the node's forwarding table. Choosing a packet's next hop is a composition of two functions. The first function is grouping the set of possible packets to a set of "Forwarding Equivalence Classes (FECs)". The second maps each FEC to a next hop. Packet headers contain much more information than needed to map them to the appropriate FECs and then to send them to the next hop, consequently, taking much more time than needed.

Unlike that in conventional IP, in an MPLS network, the assignment of a particular packet to a particular FEC is done just once, as the packet enters the network at the ingress router. The FEC to which the packet is assigned is encoded in the label. At succeeding hops, there is no further analysis of the packet's network layer header. Rather, the label is used as an index into a forwarding table, which specifies the next hop, and a new label. The old label is replaced with the new label, and the packet is forwarded to its next hop. This forwarding mechanism has a number of advantages over conventional network layer forwarding, which are the following: -

- MPLS forwarding can be done by switches which can look up and replace a label, but are not capable of analyzing the network layer header, or can not analyze it at a high rate of speed.

- The ingress router may use any information, including but not limited to the network layer header, to determine the assignment of the packet to an FEC when it enters the network. For example, packets arriving on different ports may be assigned to different FECs. Conventional forwarding, on the other hand, considers only the information in the packet header.

- Since a packet that enters the network at a particular router can be labeled differently from an identical packet entering the network at a different router, the ingress node, as a matter of policy or to support a Service Level Agreement (SLA) between adjacent networks in the Internet, can easily be considered when making the forwarding decision. This cannot be done with conventional forwarding, since the only considered forwarding information is gleaned from the packet network header.

- It is sometimes desirable to determine and explicitly choose the route the packet should follow, rather than being chosen by the normal dynamic routing algorithm, hop by hop forwarding, as the packet traverses the network. In an MPLS network, a label can be used to determine the path; no extra overhead information is needed to determine the explicit path. In conventional forwarding, "source routing" requires the packet to carry extra overhead information as an encoding of its route along with it.

Other than best effort forwarding, the packets that traverse the network may have to be treated differently. Doing so, some routers analyze a packet's network layer header to determine the packet's precedence and class of service, which are used to apply a different discard threshold or scheduling discipline to different packets. In an MLPS network, a label can also be used to determine the treatment the packet should have, since the label can represent the combination of an FEC and a precedence or class of services. An MPLS network allows (but does not require) the precedence or class of service to be fully or partially inferred from the label.

2.1 Label Distribution Protocol (LDP)

There must be a mechanism to make LSRs in an MPLS network distribute and agree on the meaning of the labels. The MPLS architecture document [4] defines this mechanism as a set of procedures and calls it the Label Distribution Protocol (LDP). The MPLS architecture allows label distribution protocols to also perform any other negotiation between two label distribution peers that is meaningful for that label binding. Two LSRs are called "label distribution peers" with respect to the binding information they exchange [5]. The negotiations between two label distribution peers concern the situations and capabilities of the LSRs that have an effect on the label binding and forwarding decision. They are performed in terms of messages such as, discovery messages, session control messages, advertisement messages, and label binding messages.

MPLS architecture [4] does not consider only one label distribution protocol; however, a number of different label distribution protocols such as label distribution protocol (LDP), constraint-based routing label distribution protocol (CR-LDP), and resource reservation protocol (RSVP), are being standardized, and each has its own properties. Other than distributing a label, every protocol has its own behavior and other concerns that differentiate it from other distribution protocols. For example, the way RSVP behaves is different from what LDP does, and the functionalities LDP performs and the considerations it takes are different from those CR-LDP considers.

CR-LDP is the protocol that is concerned with constraint-based routing. It is used to set up a constraint-based path. Constraint routing label switching path CR-LSP is a path through an MPLS network that satisfies some constraints to support the Traffic Engineering (TE) requirements, and Quality of Services (QoS) in an MPLS network. The difference between LSP and CR-LSP is that while other paths are set up only based on information in routing tables or from a management system, the constraint-based route is calculated at the edge of the network (ingress node) based on criteria, including but not limited to routing table information. The purpose of this functionality is to give the LSP desired characteristics in order to better support the traffic sent over it. The reason for setting up CR-LSPs might be the need of assigning certain bandwidth or other service class characteristics to the LSP, or to ensure the separation of alternative paths through the network [6].

Explicit Routing is a subset of the more general constraint-based routing where the constraint is the explicit route. Other constraints are defined to provide a network operator with control over the path taken by an LSP. An explicit route is a list of nodes or groups of nodes represented in and followed by the label request message. When establishing the CR-LSP, the label request message may traverse all or a subset of the nodes in a group. Certain operations to be performed along the path can also be encoded in the constraint-based route. CR-LDP allows for explicit routes, using both strict and loose hops, providing maximum flexibility in building a specific path through a network. It also has the ability to allocate bandwidth based on an LSP's priority. Capabilities of CR-LDP include the following:

- CR-LDP is an extension on an already existing LDP protocol.

- It remains in a hard state.

- It has both explicit setup and explicit teardown.

- It needs no refreshing; once established, it stays up until torn down.

2.1.1 Label assignment

Label assignment occurs based on a common grouping or forwarding equivalence class (FEC); packets are classified together based on common attributes, such as source addresses (policy based routing), source and destination address pairs, destination address, and even Type of Service (ToS) or Differentiated Service Code Point (DSCP) bits. All packets grouped into the same FEC receive similar treatment along the LSP.

The decision to bind a particular label to forwarded equivalent classes (FEC) is made by Label distribution router (LSR), which is downstream with respect to that binding. The LSR up stream sends a label request message to its downstream LSR peer asking for a label for that FEC. The downstream LSR informs the upstream LSR of that binding. Some FECs correspond to address prefixes, which are distributed via a dynamic routing algorithm. The setup of the LSPs for these FECs can be done in one of two ways: Independent LSP Control or Ordered LSP Control. In an independent LSP control, an LSR distributes the binding to its LSR peers if it recognizes that FEC. However, in Ordered LSP Control, an LSR only binds a label to a particular FEC if it is the egress LSR for that FEC, or if it has already received a label binding for that FEC from its next hop for that FEC [4].

The ordered LSP control technique is useful in constraint based routing, where it should be known that the whole path has sufficient resources prior to starting to

forward data packets. On the other hand, in the independent LSP control, the ingress router LSR starts sending data packets as soon as it receives the binding from the nearest LSR peer. This is to avoid wasting time waiting for the ordered binding from the egress node.

2.2 Path calculation

Due to the variations in the requested destination, changes in the available resources, and the type of desired services, there are many available paths to reach each node in the network, which can not be manually configured and stored in each node, but have to be calculated based on what resources the network has and what attributes they should satisfy. Therefore, a node calculates a path to a desired destination on some metrics to satisfy some constraints. The constraints are those, which appear in the Type of Services (TOS) field in the IP header, or those classes of services in a DiffServ code point. TOS is a part of the IP header that tries to provide prioritization; it interprets assigned services to the packet such as low latency, high throughput, high reliability, and low cost.

Calculating a normal path, an IP best effort path, is slightly different from calculating a path to satisfy some constraints. In calculating a path all the working links and nodes participate in the path calculation regardless of their metrics. On the other hand, in constraint-based routing, a node excludes the links and nodes

that do not meet the constraints from the path calculation. For example, if a node requests a certain amount of bandwidth for this flow, it will exclude the links that do not have that amount. So, the node either finds a path that satisfies these constraints or finds no path at all.

The existing path calculation algorithms such as Open Shortest Path First Protocol (OSPF) can also calculate a separate set of routes for each IP Type of Service (TOS) [7]. This means for any destination there can be more than one entry in the routing table, one for each IP TOS. The OSPF protocol maintains multiple equal-cost routes to all destinations. Each route has its separate next hop and advertising router. It is not required that a router running an OSPF keep track of all possible equal-cost routes to a destination. The number of kept routes is an implementation choice and does not affect any of the algorithms presented in the OSPF specification [1].

It is almost impossible to find multi equal-cost paths if the path to the destination is to be calculated upon constraints other than hop count, so that there will be better available paths to a specific destination. That means the node could have multi "almost equal" paths or let us name it "multi best paths" to the destination for every IP type of service (TOS). These multiple-equal paths are to be used for a load balance by distributing the load over them.

Distributing a micro flow even over different exactly equal cost and loaded paths does not work. This is due to the fact that every path may have changing circumstances. These circumstances may result in longer queuing, consequently reordering the flow by the destination node. Equal-cost multiple paths are very useful when they are used to distribute the traffic over the network. A node can calculate multiple paths to determine which one is good for which flow. There are always trade-offs: a node may choose to use a higher delay path to ensure a bandwidth or reliability, or it could make a decision to equalize the load over the network if the flow desired services are not tightly constrained.

2.3 Query Messages

A label distribution protocol LDPs has the ability to inquire about the already established LSPs by sending query messages through them [8]. This message is sent from the source node to gather information needed by the inquiring node about LSPs. The query message can be used for LDP LSPs as well as for Constraint-Based Label Switched Paths (CR-LSPs). It can be used to gather information about:

- LSRs, which form the LSP.

- Labels along the LSP.

- Information on which LSRs are merging points along the path.

- Unused bandwidth (as described in "Improving Topology Data Base Accuracy with Label Switched Path Feedback in Constraint Based Label Distribution Protocol "[3]).

- Anything that is needed in the future and can be computed and encoded in a TLV [8].

A Query-Reply message carries the queried information that is generated by the egress LSR of that LSP, or an intermediate LSR in case of partial reply, and sent back upstream as a response of the query message. Every intermediate node that receives the reply attaches the queried information to the Query-Reply message and sends it upstream. Eventually, this information arrives and is used by the ingress node.

CHAPTER 3

Label Distribution Protocol (LDP) with feedback

The IETF draft "Improving Topology Data Base Accuracy with Label Switched Path Feedback in Constraint Based Label Distribution Protocol" [3] proposes that a node can build a topology map of the network from the advertised information about the links, which is mentioned in [1] and [2]. Information about links that may be useful for reasons of quality of service (QoS) includes parameters such as available bandwidth and delay. The information in this topology database is often out-of-date with respect to the real network. Available bandwidth is the most significant of these attributes and it can float considerably with respect to reality, due to the low frequency of link state updates that can be sustained in a very large topology.

Because this information is required to be as up-to-date as possible for accurate traffic engineered paths, the IETF draft [3] also proposes adding to the signaling protocol the ability to attach actual link bandwidth availability information at every link that the signaling message traverses. This means that every time a feedback message flows backwards toward the source to tell it of the success, failure, or termination of a request, this message contains detailed information about the availability of the bandwidth for the path that the message has followed. This information, which is very up-to-date, is received by the source node, attached to the source node's topology database, and will be considered on further source

route computations. The result is that the source node's topology database will keep up-to-date regarding that part of the network through which it is establishing paths. Also, every node along the path copies the information from the feedback message to its database. It will be up-to-date about the down stream slice of that path. This makes the intermediate node benefit from other nodes' feedback messages.

This mechanism is nothing more than that the source node receives feedback every time it attempts to establish, or release, or withdraw a path. It represents an alternative way to either waiting for floods or introducing guessing into the path calculation algorithm. These fed-back data that the node has learned should not be re-flooded to the other nodes in the network; the data override flooded information to be used by the node for its own route calculation until a superseding flood or new feedback value arrives [3].

If the topology database is optimistic, the first selected path will likely contain links that do not in reality have sufficient unreserved bandwidth [3]. Therefore, the path is only established up to the link that does not have sufficient bandwidth. The signaling message will be blocked in that link and a feedback message containing the actual bandwidth is sent back toward the source node collapsing the partially created path. The source computation path will be calculated again. This procedure will continue until the destination is reached or no path is available. Each time, only one path is computed and used to send messages.

The source node will receive feedback from those links that the signaling message has just traversed. The same procedure may be repeated as many times as is necessary. Each time the node learns from its mistakes, until a path to the destination that satisfies the request is found, or the node knows that no paths remain in its topology database to the destination. It actually behaves a lot like a depth-first search. This property is not present with flooding mechanisms alone since the source node must randomly guess, or continually make the same mistakes, or abort until the next flood arrives [3].

If the topology database is pessimistic, the IETF draft [3] proposes using other algorithms to bring the topology database back to the optimistic state, so the feedback algorithm can operate. A selective forgetting algorithm, for example, is one of these proposed algorithms. It requires no more than changing the value of reserved bandwidth in the node's topology database to zero over a short time interval, so the node will be optimistic, but not up-to-date regarding the real network. Although such algorithm enables the feedback algorithm to find a path, this path might not have sufficient resources. This is due the fact that the path is calculated upon an optimistic database but not actual data.

CHAPTER 4

LDP with Feedback Algorithm validation and path query messages

4.1 LDP with Feedback Algorithm validation

The IETF draft "Improving Topology Data Base Accuracy with Label Switched Path Feedback in Constraint Based Label Distribution Protocol" [3] proposes that its algorithm deals only with the topology database when it is optimistic. This is theoretically correct because the node excludes from the path calculation the links that do not meet the required attributes. In this case, a node will find no path that satisfies the request. However, the network is a dynamic system, dynamic as a whole, and a node is participating in other nodes' requests, which makes it well informed from the feedback that the node receives from other established or released LDPs that pass by it.

Although we cannot take an individual node to study a network, we assume the situations where the node might be pessimistic:

- The node is out-of-date because not too many LDPs have been established after the network has been saturated, or, more precisely, some links are saturated. In this case, either the network or the links are

still saturated, and the node will find no path even if it is up-to-date, or the node will receive feedback in release of LDPs, which is proposed in [3].

- The node is out-of-date because not too many LDPs have been established and the network is not loaded; the node is still optimistic even though it is out-of-date, and it will find a path if there is one.

- The node was down; it will receive flooding about the real status of the network as soon as it comes up again [1], [2].

Based on these viewpoints we strongly believe that a node will never be pessimistic regarding the whole network, or we can say that a node can be pessimistic regarding a very few number of links. We can confidently say that the feedback algorithm proposed in [3] deals with all topology database states, optimistic and pessimistic, and this is what are we going to see in the simulation chapters.

4.2 Path Query Messages

The IETF draft [3] describes its algorithm as behaving very much like depth-first search, discovering only one path each time. Since the source node repeats the procedure of path calculation, and label request message more than once in order to make the label request message reach the destination, it is better to

have that procedure synchronized. We propose modifying that algorithm to behave like breadth-first search instead of depth-first search, where the source node explores more than one path at the same time. This will reduce the time taken to set up a path and correct the node's topology database by receiving much more feedback. This proposal is nothing more than that the source node sends query messages through the best paths, which the source node would discover in the next path calculations in case this label request message failed to reach the destination, and receives feedback.

These path query messages could be sent to inquire about any other attributes. (Refer to Appendix A for more information). However, bandwidth is the attribute we are interested in here, due to its importance, and its rapid change. The ingress node sends the signaling message through one path and sends query messages through the other paths. It will receive query reply messages that carry the actual values of the unreserved bandwidth in each link that it has traversed. If the signaling was blocked at any point, the source node would have a better chance to find a path that has sufficient bandwidth the next time it computes the path. This kind of feedback will:

1. Reduce the number of path computation iterations, reduce the overhead of path computation especially if the path computation is done by what is called a server node, where one node calculates the paths for all the nodes in the network.

2. Give the node the ability to find the proper path earlier, due to the synchronization between the label request message and query messages. The ingress node does more than one step at the same time. It does not receive feedback from only one path, but it receives feedback from different paths at once.

3. Make the computation of the path more efficient especially if the paths are independent. By efficient we mean that it gathers more feedback than that if the paths share some links. The ingress will have feedback about more links that will be included or excluded from the next path computation.

4. Reduce the chances that the node will be pessimistic regarding some links, and the node will have a better chance to find a cheaper path if there are any.

5. Make the source node have a bigger and more up-to-date vision of the state of the network than that of feedback from one path at a time. It would be very useful for a load balance as well. A load balance here is not meant to be distributing the same flow over different paths, due to the fact that it is almost impossible to find equal-cost paths unless the cost was meant to be the hop count. It is the distribution of different

flows upon different paths. The source node would be able to consider the load balance when choosing a path through which to send a signaling message.

The number of query messages is an implementation matter. The query and query reply message formats and procedures are described in appendix A.

CHAPTER 5

Simulation

The aim of this simulation is to study the effect of different LDP algorithms on allocating a path through the network: how they affect the blocking probability, and how long it takes to set up a path in each one. The first LDP algorithm is the original LDP algorithm, where, in case of failure, the node does not retry sending the message again, but rather waits for flooding. The second one is LDP with feedback algorithm where the node sequentially tries sending other requests until it reaches the destination or knows there is no path to take. However, the third is LDP with query messages algorithm, where the node tries to discover more than one path at the same time.

5.1 Generating Networks

Georgia Tech Internetwork Topology Models (GT-ITM) [9] is used to generate the networks. Two types of networks have been generated: flat random networks and 2-level hierarchical networks. From both types, a large number of networks have been generated by changing the probability of the connectivity, changing the number of nodes in each network, and changing the number of nodes in the core for the hierarchical type. See Table 5.1.1 and Table 5.1.2 for more details.

For each topology, between 5 and 10 random seeds were used, each random seed contains number of requests that are more than the capacity of the network.

Network Type	Number of nodes	Nodes in the core network
Hierarchical	248	8
Hierarchical	96	8
Hierarchical	96	6
Hierarchical	91	8
Hierarchical	91	6
Hierarchical	90	8
Hierarchical	90	6
Hierarchical	72	8
Hierarchical	72	6

Table 5.1.1 Hierarchical networks used in this simulation

Network Type	Number of nodes	Connectivity probability
Random	248	0.033%
Random	96	0.033%
Random	96	0.035%
Random	90	0.033%
Random	80	0.043%
Random	72	0.045%
Random	72	0.033%
Random	50	0.043%

Table 5.1.2 Random networks used in this simulation

5.2 Simulator structure

The simulator is built using Visual C++ language. A network node in the simulator is represented as a class. Each node is an independent object and has its own independent properties such as node ID, node routing table, to which node the node is directly connected, and the properties of the links (cost, delay, maximum bandwidth, the available bandwidth). After the node calculates the path, the sequence of nodes that represent the path is inserted in the label request message, and message is sent to the next node in the path. It actually behaves the same as if it were a real network.

5.3 Simulation flow

The simulator tends to demonstrate the three phases any network goes through, where the topology databases in the nodes are pessimistic, optimistic, and up-to-date. After building up the network, which was created by the network generator GT-ITM, the simulator starts the first phase, which is feeding the network with random requests. A request is a tuple of three items: source node ID, destination node ID and the amount of desired bandwidth. Although the seeds are meant to be random, we created the source and destination nodes in a way that makes them cross the core network of the hierarchicy. This phase continues till one of two conditions holds:

- Either the time reaches the maximum point, which can be changed as needed, or
- The blocking probability reaches or exceeds 95%. The blocking probability is the ratio between the failed requests and the total number of requests from the last time the blocking probability was calculated.

In the simulation, the second condition is always the trigger.

Then the next phase, where the number of arriving and departing requests is almost the same, and which tends to illustrate up-to-date topology database, takes place. This phase lasts one third of the time of the first phase. Then the final phase, where the number of departing requests is more than the number of arriving requests starts. This is used to demonstrate a pessimistic topology database.

Before we go to the next section to analyse the graphs, we should point out the following:

- The points that represent the output graphs are taken every 100 time units.
- Not all the blockings are due to a lack of the topology database knowledge. This can be seen in very low connectivity networks where

one link might make a difference in the label request message reaching a destination, but this has nothing to do with the node being up-to-date or not.

5.4 Simulation results

Two measurements are used in this simulation to differentiate the usefulness among the three algorithms. The first one is the blocking probability. It is the ratio of failed requests to the total number of requests in every 100 time units. It is independently calculated every time interval. The second measurement is the time taken to set up a path. It is the time the ingress has to wait until it receives a mapping message or knows that there is no path to use.

5.4.1 Simulation graphs representing the behaviour of the original LDP algorithm, LDP with feedback algorithm, and LDP with query messages

Despite the variety of the networks used in this simulation and the large number of different randomly generated traffic used for each network, the output graphs look very much the same. The time and the load may differ, but the shapes of the graphs are identical. The following graphs are taken from both hierarchical and random 248 node networks.

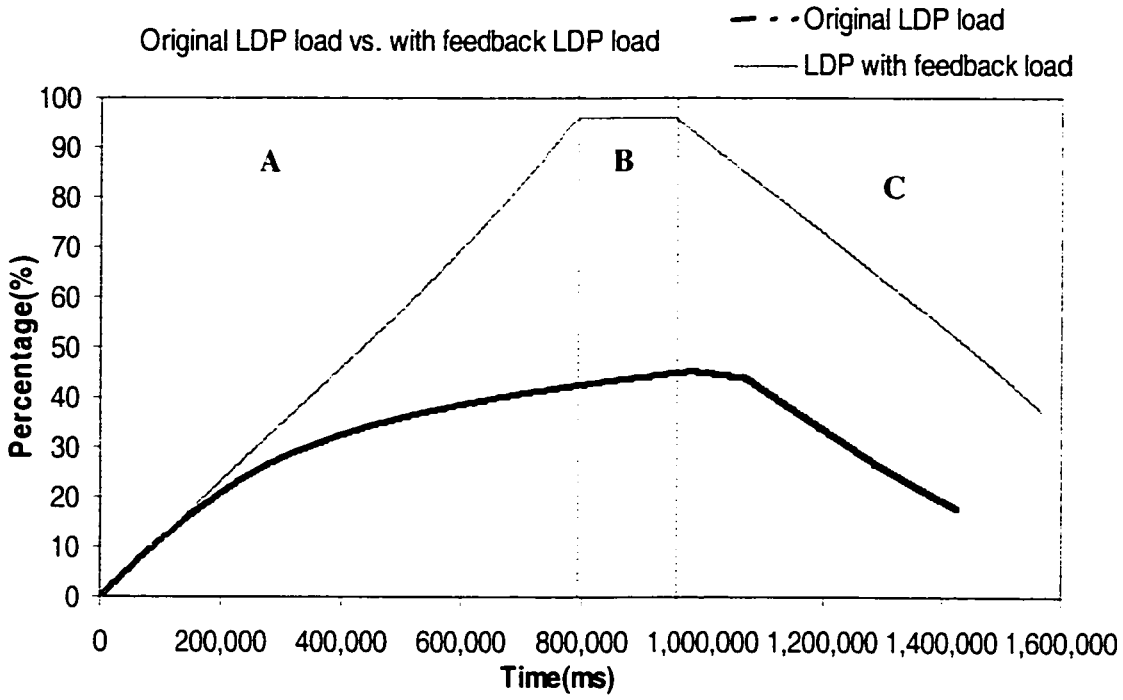

Figure 5.4.1.1 original LDP and LDP with feedback load comparison in a random network

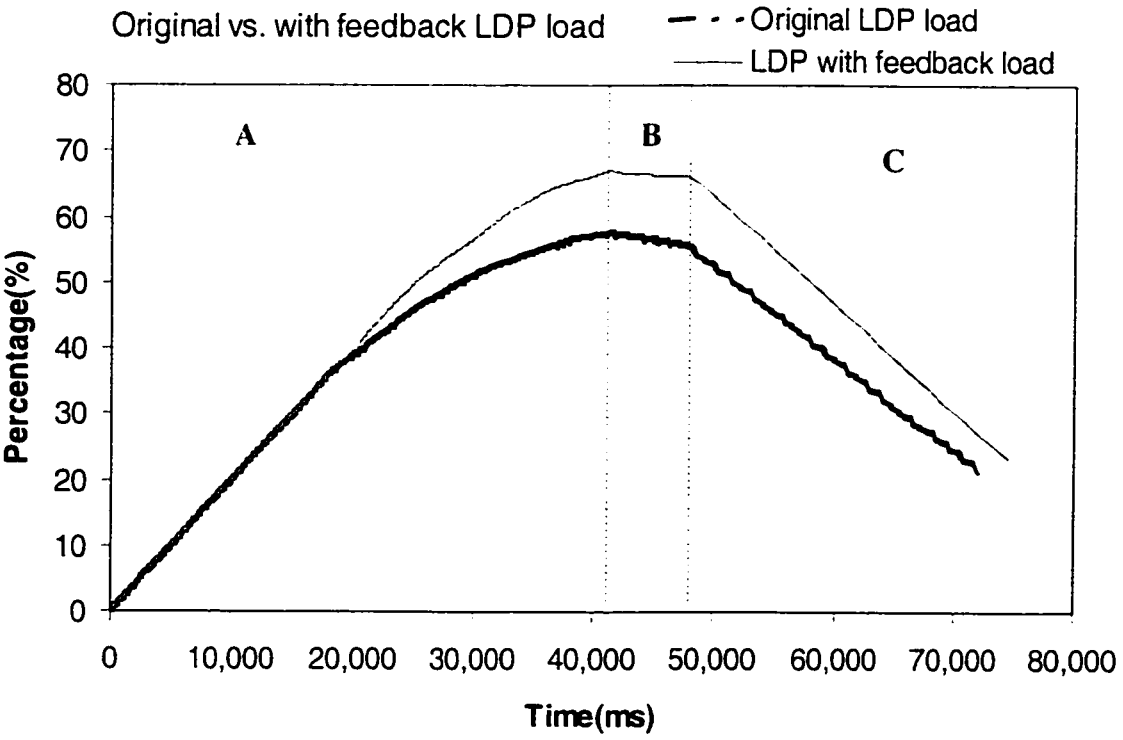

Figure 5.4.1.2 original LDP and LDP with feedback load comparison in a hierarchical network

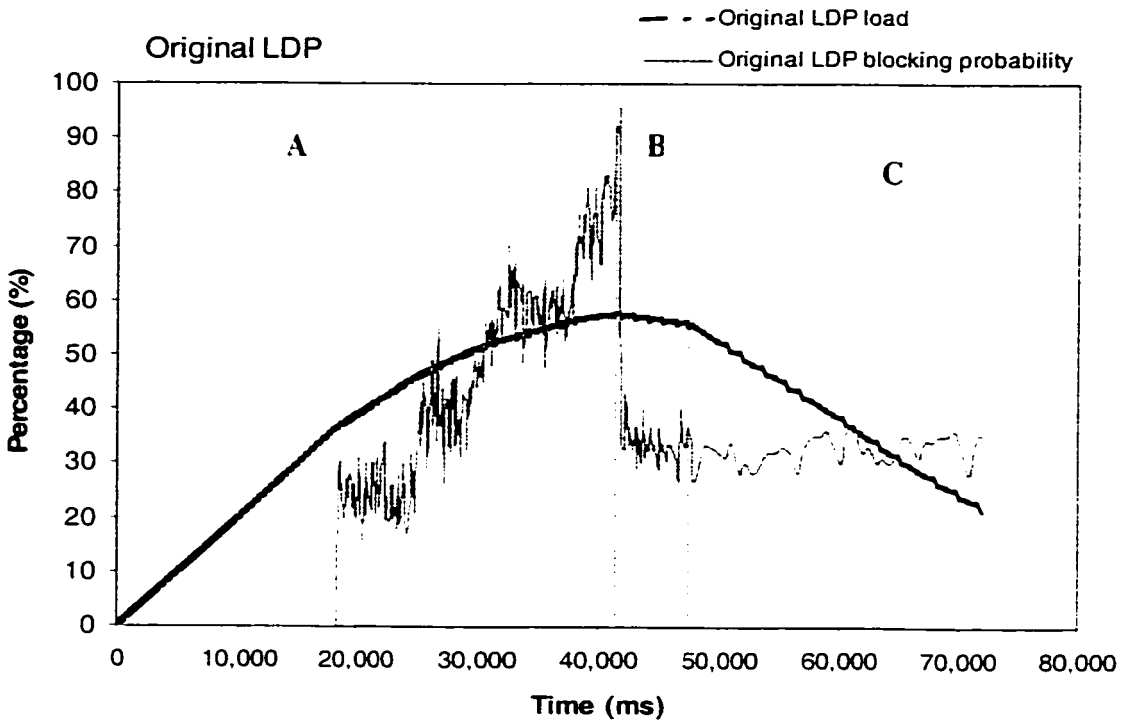

Figure 5.4.1.3 original label distribution protocol in a hierarchical network.

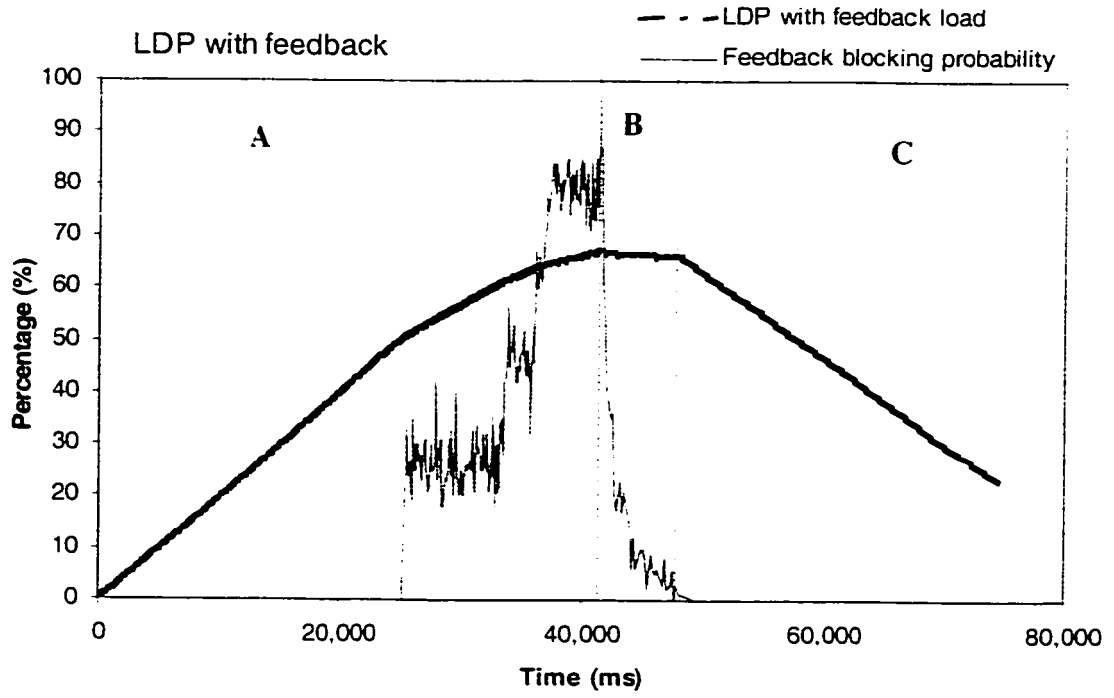

Figure 5.4.1.4 label distribution protocol with feedback in a hierarchical network

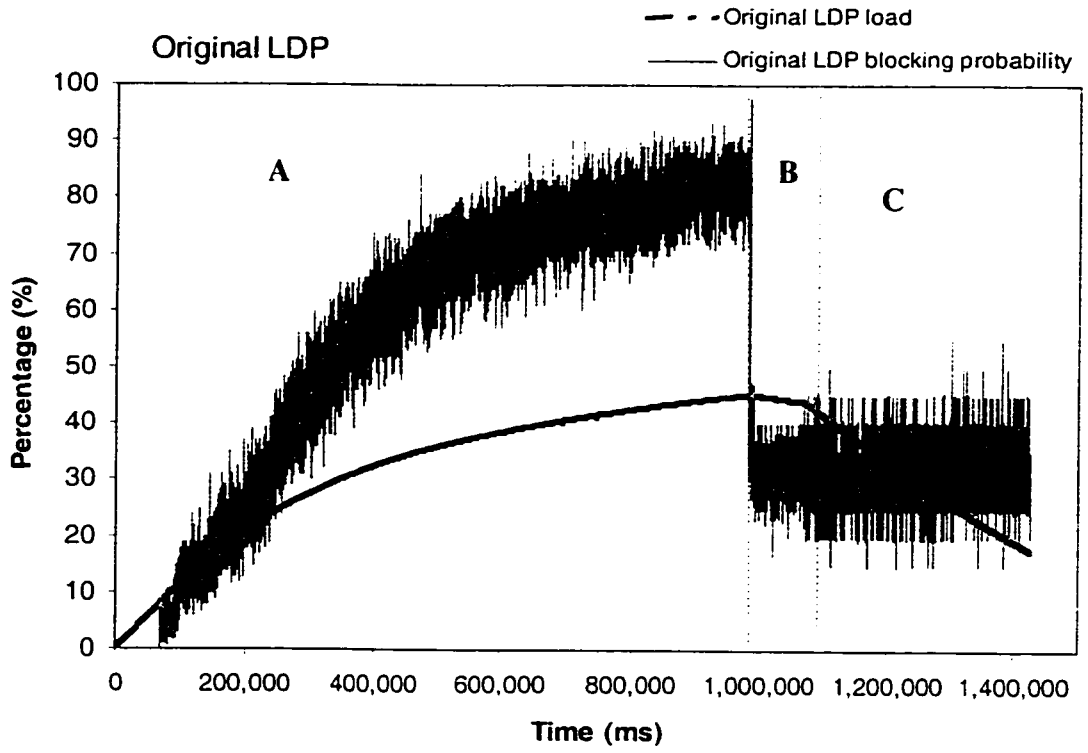

Figure 5.4.1.5 original label distribution in a random network.

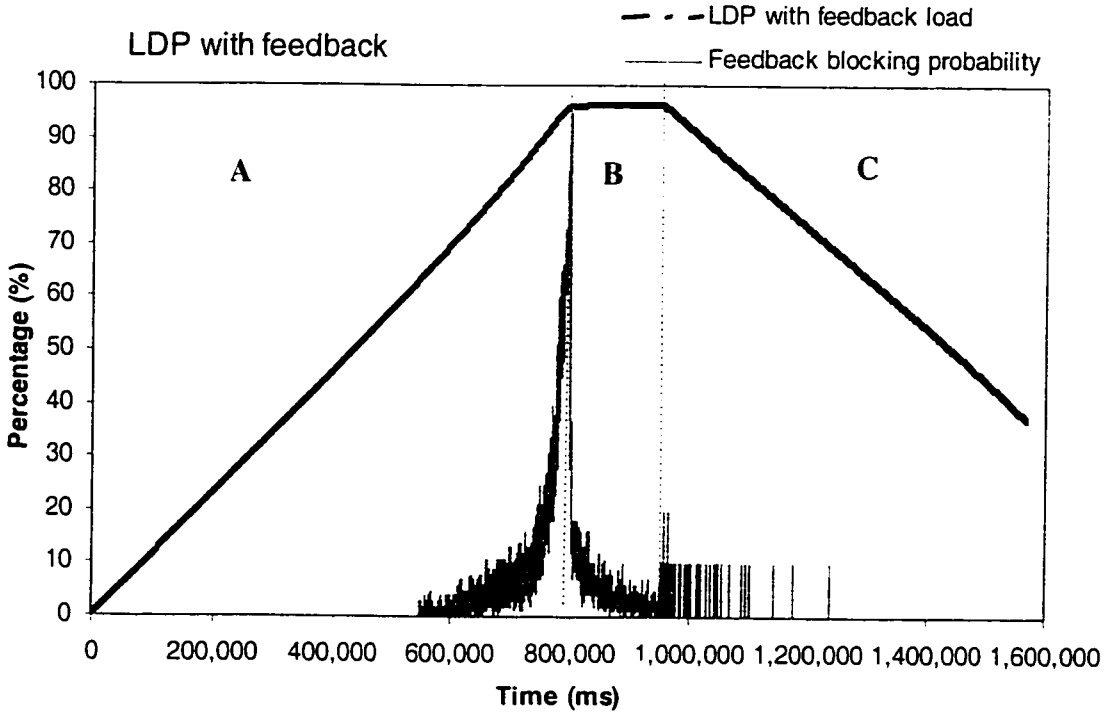

Figure 5.4.1.6 label distribution protocol with feedback in a random network.

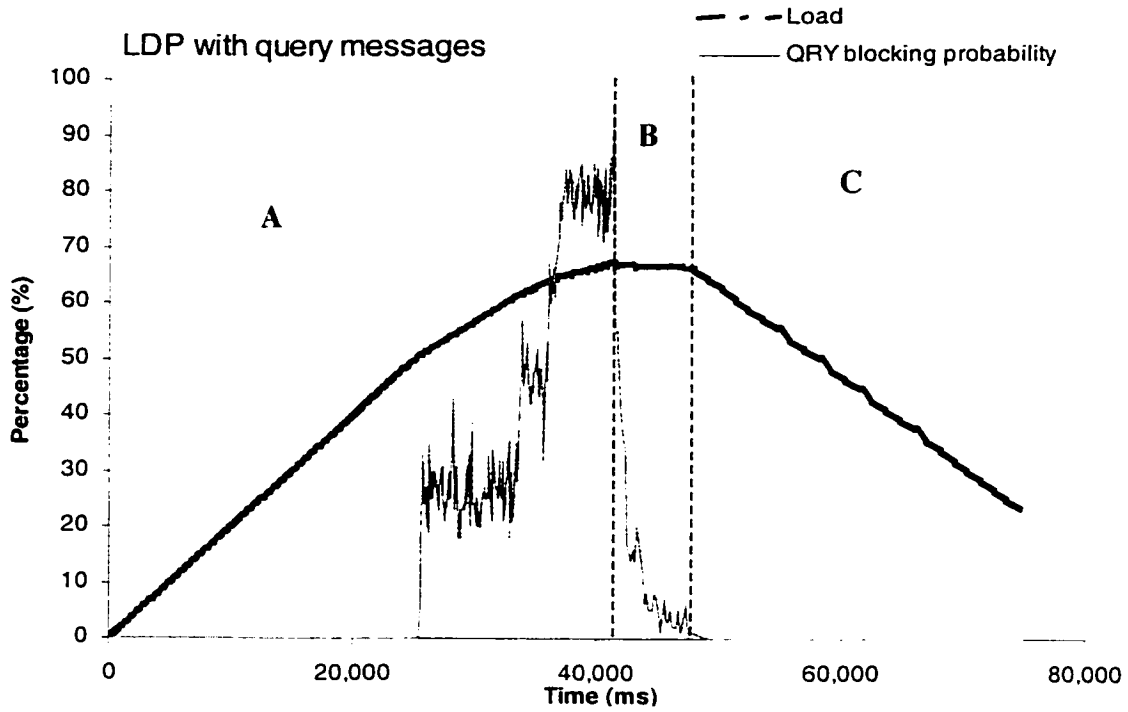

Figure 5.4.1.7 LDP with feedback and query messages in a hierarchical network.

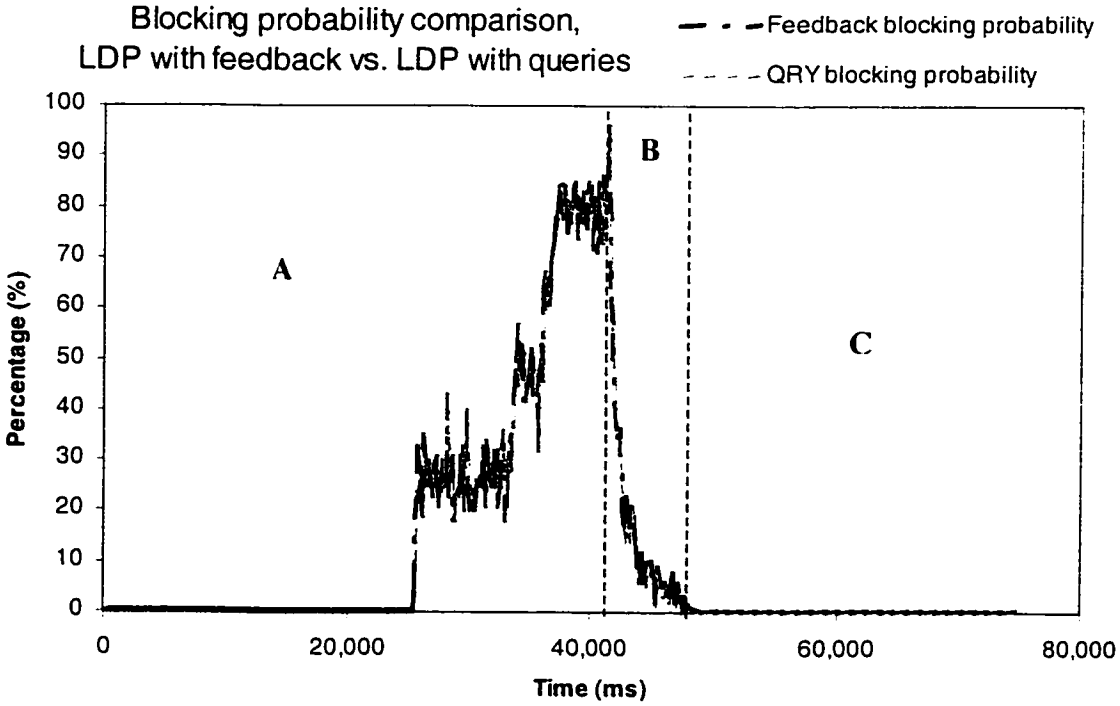

Figure 5.4.1.8 LDP with feedback and LDP with query messages blocking probability comparison in a hierarchical network

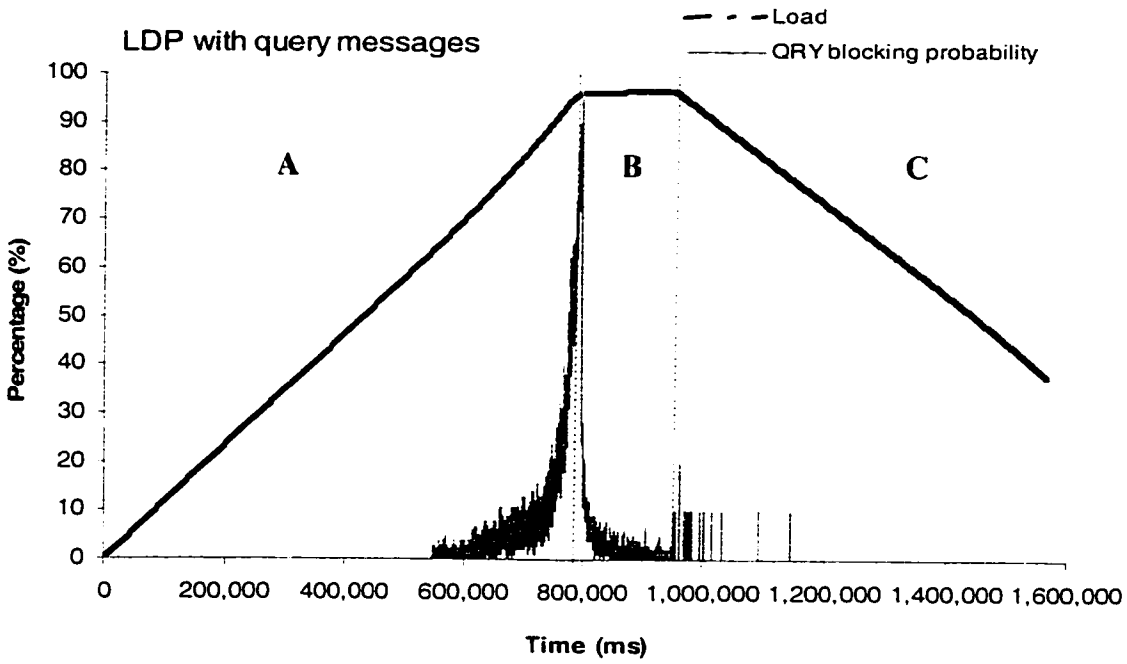

Figure 5.4.1.9 LDP with query messages blocking probability in a random network

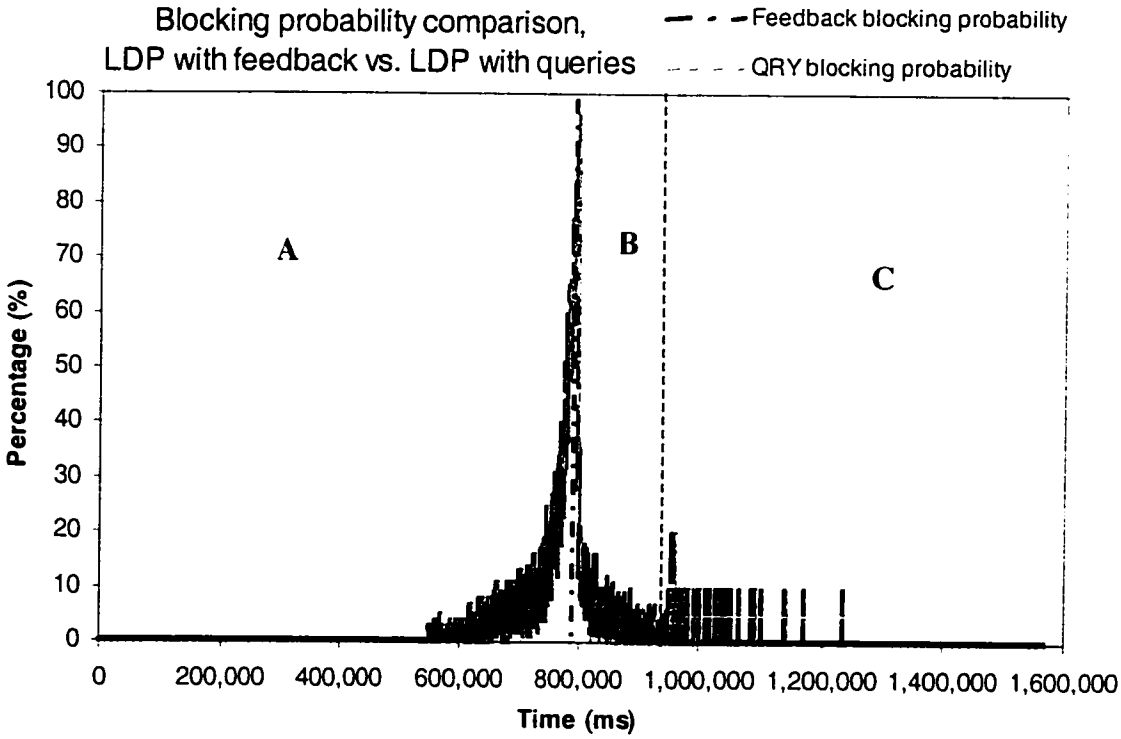

Figure 5.4.1.10 LDP with feedback and LDP with query messages blocking probability comparison in a random network

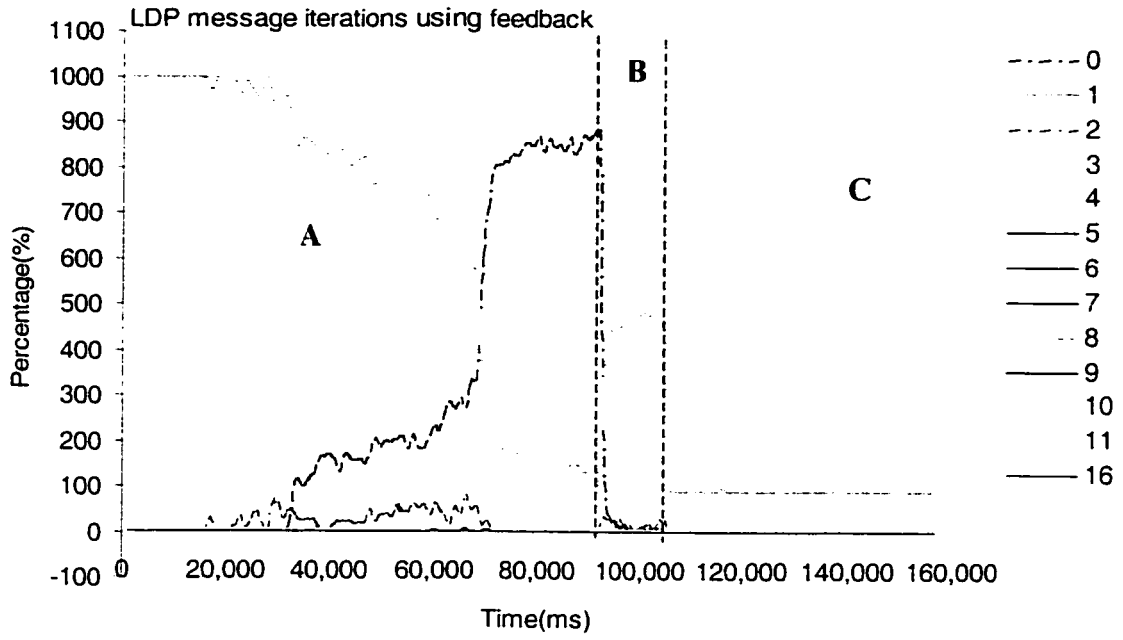

Figure 5.4.1.11. LDP messages iterations using LDP with feedback

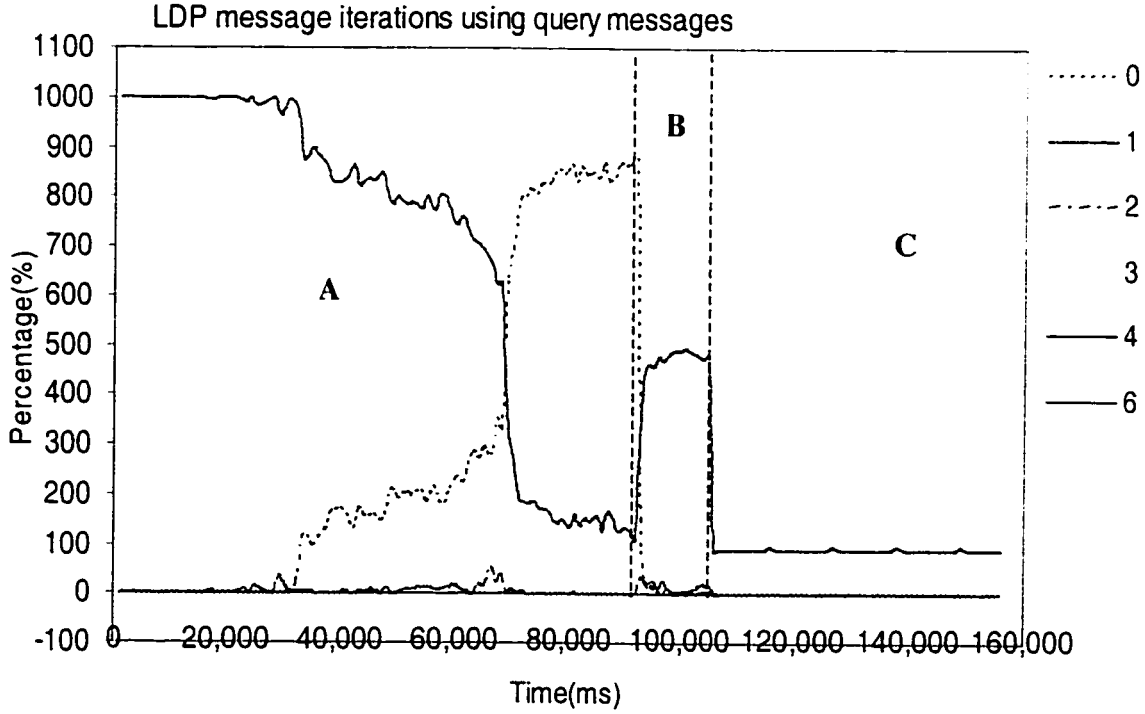

Figure 5.4.1.12. LDP messages iterations using LDP with query messages.

5.4.2. Interpretation of graphs

Figure 5.4.1.1 and Figure 5.4.1.2 show the load comparison between original LDP and LDP with feedback algorithms in hierarchical and random networks. It is obvious from these figures that the LDP with feedback algorithm utilises the network more effectively. By using the label request messages with feedback algorithm, the network could handle much more load than using those without feedback messages. Figure 5.4.1.3, Figure 5.4.1.4, Figure 5.4.1.5, and Figure 5.4.1.6 illustrate the relation between the load and the blocking probability over all the nodes in the network using original LDP and LDP with feedback.

Figure 5.4.1.7 and Figure 5.4.1.9 demonstrate the relation between the load and the blocking probability in hierarchical and random network using LDP with query messages. Figure 5.4.1.8 and Figure 5.4.1.10 show a blocking probability comparison between LDP with feedback and LDP with query message. It is clear from this comparison that there is some improvement when using the query message mechanism. Finally Figure 5.4.1.11 and Figure 5.4.1.12 show the number of label request message attempts when using LDP with feedback and when using LDP with query message. The next sections will describe the three database regions these graphs have shown.

5.4.2.1. Optimistic topology database

The first phase is where the topology database is optimistic (area A in the graphs) and the load curve starts from point 0 where the bandwidth in the network is totally free going up until it reaches the point where the blocking probability curve reaches 95%. From Figure 5.4.1.3, Figure 5.4.1.4, Figure 5.4.1.5, Figure 5.4.1.6, Figure 5.4.1.7 and Figure 5.4.1.9, we can see:

- The failure of finding a path through the network using the original LDP starts at earlier time than that in LDP with feedback.

- The number of blocked requests when using original LDP is more than that when using LDP with feedback even before the network is nearly saturated.

5.4.2.2. Up-to-date topology database

After the blocking probability reaches 95%, the simulator keeps feeding and releasing the same number of LDP, which illustrates the up-to-date topology database state (area B in the graphs). As Figure 5.4.1.3, Figure 5.4.1.4, Figure 5.4.1.5, Figure 5.4.1.6, Figure 5.4.1.7 and Figure 5.4.1.9 show, the blocking probability drops. The drop in the blocking probability implies that the blocked requests were due to the lack of paths, not due to the fact that the node could not find them.

5.4.2.3. Pessimistic topology database

After some time, the simulator starts releasing more LDPs than requesting them, which is the state when the nodes' topology database is supposed to be pessimistic (area C in the graphs). We can clearly see that the blocking probability drops to 0, which proves our beliefs about the pessimistic database. This, in turn, proves that the LDP with feedback algorithm works for all states of topology database. See Figure 5.4.1.3, Figure 5.4.1.4, Figure 5.4.1.5, Figure 5.4.1.6, Figure 5.4.1.7 and Figure 5.4.1.9

5.4.2.4. LDP with feedback and with query messages blocking probability comparison

It is obvious from certain figures that the LDP with feedback and query messages behaves almost the same as LDP with feedback, with some improvement in the blocking probability. See Figure 5.4.1.8, Figure 5.4.1.10, Table 5.4.2.1 and Table 5.4.2.2.

Algorithm	No. Of LDPs	No. Of blocked LDPs	Overall blocking probability
LDP with Feedback	930561	32577	3.500791%
LDP with Query messages	930561	30594	3.287694%
Improvement in blocking probability			0.213017%

Table 5.4.2.1 label request message with feedback and with query messages comparison in a random network.

Algorithm	No. Of LDPs	No. Of blocked LDPs	Overall blocking probability
LDP with Feedback	44542	7973	17.8999%
LDP with Query messages	44542	7910	17.7585%
Improvement in blocking probability			0.1414%

Table 5.4.2.2 label request message with feedback and with query messages comparison in a hierarchical network.

5.4.3 Simulation results for time taken to setup a path by using LDP with feedback vs. LDP with feedback and query messages

Assuming LDP with feedback and LDP with feedback and query messages find the same path, table 5.4.3.1 shows the total time taken to set up a path in both scenarios and the amount of improvement there is. The variation in improvement is due to the number of tries the algorithms makes to set up a path or to know that there is no path to take. Figure 5.4.1.11 and Figure 5.4.1.12 show the number of label request messages tries that LDP with feedback and LDP with

queries make in order to find the path or to know that there is no path that satisfies the request. Zero attempt means that the node has found no path in its topology database and a label request message has not been sent.

The number of iterations is taken every 1000 ticks. It is obvious from Figure 5.4.1.12, and Figure 5.4.1.12 that the number of label request message iterations in LDP with feedback is more than that in LDP with query messages.

1-The time taken to set up path with feedback	**2-** The time taken to set up path with feedback and query messages	Improvement in the path setting up time
3572	1649	53.8%
3176	1888	40.5%
2975	1628	45.2%
3359	2331	30.6%
1618	1287	20.4%
1511	697	53.8%
29913	15001	49.8%
672	600	10.7%

Table 5.4.3.1 label request message setting up time

5.4.3.1 Simulation interpretations and observations

From table 5.4.3.1 we can see that the simulation shows different improvements in time for label request message setup and we can observe the following:

- The variation in the improvement in time depends on the state of the network and where the blocking occurred: the further the blocking is from the ingress node, which should be the case most of the time, the better the improvement. It is expected that the node has less up-to-date information about the further nodes than the nearer ones.

- The outcome shortest path in the single path calculation equals the second shortest path in multi path calculation, unless the excluded link(s) up on the previous feedback make(s) the difference.

- Most of the time the multi paths happen to share some link(s), in other words, it is rare to have completely independent cheapest paths.

- The label request message with feedback and query messages show the improvement in both cases: setting up the path or knowing that there is no path to take.

- The models of the network have no effect on comparing the two scenarios for path allocation, because the ingress node deals with a sequence of nodes, which tend to handle the label request message.

- If the ingress node is able to make it to the egress the first time, label request messages take the same time for all the procedures.

- The label request message without feedback is excluded from the result in Table 5.4.3.1 because it is either it waits for the next flooding, or the time taken is the same as what other procedures take (in the case where no blocking occurs).

Chapter 6

Conclusion

Label distribution protocol with feedback messages helps to reduce the error in the topology database in the source node to 0 and to make the topology database as up-to-date as possible. Consequently, the source node will handle the upcoming requests more efficiently, and find paths for them if they exist. Label distribution protocol with feedback deals with any kind of deviation in the topology database, optimistic, up-to-date, and pessimistic. Adding label distribution protocol query messages improves the network's ability to handle some requests even more than LDP with feedback. It also decreases the number of retry attempts the node takes to find the proper path.

We have seen through the simulation how useful the feedback algorithm and the query messages are in utilizing the network, and serving the upcoming requests. We have also seen the improvement in setting up time and network utilization the query messages algorithm has over the feedback algorithm. Although feedback and query messages algorithms introduce an overhead in terms of extra messages traversing the network, using them is reasonable.

6.1 Future Work

Feeding back the ingress node with some other constraints, which affect the path computation, than the unreserved bandwidth may have interesting results. We also expect that feedback in both directions would better improve the topology database. Yet testing this algorithm on a real world network simulation with actual traffic, where messages between nodes might be synchronised or dropped, will introduce too many new issues.

References

[1] J. Moy,"OSPF Version 2", RFC 1583, March 1994.

[2] Li, T., Smit, H., "Extensions to IS-IS for traffic engineering", Internet Draft, draft-ietf-isis-traffic-04.txt, August 2001.

[3] Ashwood-Smith, P., Jamoussi, B., Fedyk, D., Skalecki, D., "Improving Topology Data Base Accuracy with Label Switched Path Feedback in Constraint Based Label Distribution Protocol", Internet Draft, draft-ietf-mpls-te-feed-05.txt, Nov 2002.

[4] Rosen, E., Viswanathan, A. and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031, January 2001.

[5] Andersson, L. et al., "LDP Specification", RFC 3036, January 2001.

[6] B. Jamoussi et al., "Constraint-Based LSP Setup using LDP", RFC3212, January 2002

[7] Guerin, R., Kamat, S., Orda, A., Przygienda, T. and D. Williams, "QoS Routing Mechanisms and OSPF Extensions", RFC 2676, August 1999.

[8] Ashwood-Smith, P., Paraschiv, A., "Multi Protocol Label Switching Label Distribution Protocol Query Message Description", Internet Draft, draft-anto-ldp-query-04.txt, May 2002.

[9] GT-ITM, "Georgia Tech Internetwork Topology Models", http://www.isi.edu/nsnam/ns/ns-topogen.html#gt-itm.

[10] Awduche, D., Malcolm, J., Agogbua, J., O'Dell, M. and J. McManus, "Requirements for Traffic Engineering Over MPLS", RFC 2702, September 1999.

Appendix A

A.1 Path Query message

The path query message is similar to that defined in [8] in the way it behaves, but it is different in its encoding:

- It is not used to enquire about an already established path, so that it can't include a query label TLV as a parameter.

- It doesn't contain FEC TLV and LSPID as optional parameters because it enquires about resources in the path regardless the kind of FEC that the path carries.

A.1.1 Path Query Message encoding

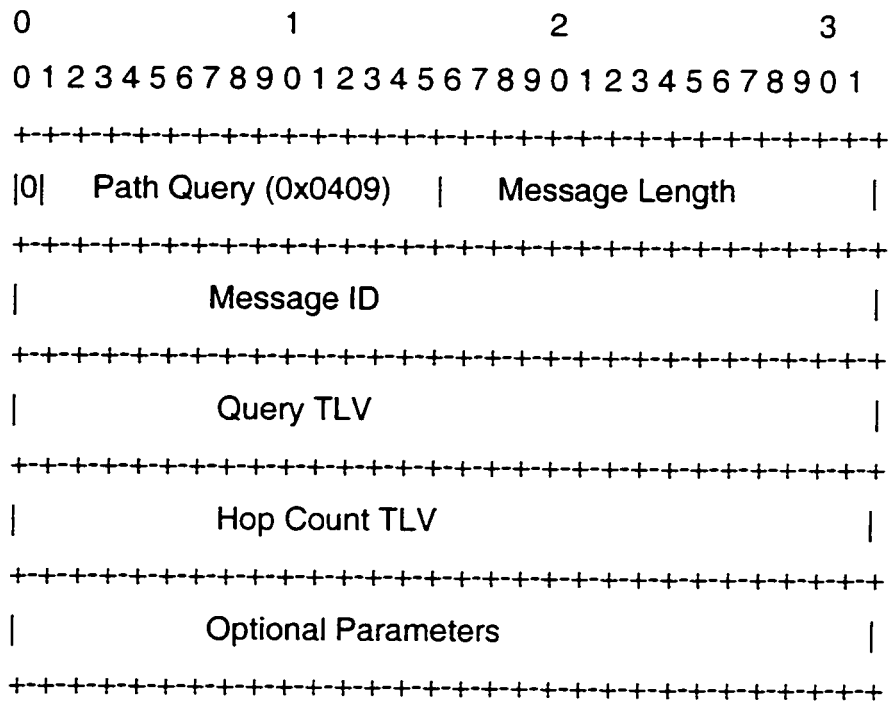

Figure A.1.1 path query message format

Message ID

32-bit value used to identify this message.

Query TLV.

What to query. Refer to [8] Section 7 Query TLV for encoding.

Hop Count TLV

specifies the number of hops that can still be traversed before the message is dropped. Its initial value is set to 255 (or the configured value, if any). Every LSR that receives the Path Query Message has to subtract

1 from the Hop Count value. The Path Query message must be dropped if the hop count value becomes zero. An LSR that drops it should send a Notification signaling Loop Detection in reply to the ingress of the message. See [5] for Hop Count TLV encoding.

Optional Parameters

This variable length field contains 0 or more parameters, each encoded as a TLV. See [8] for more details.

A.1.2 Path Query message Procedure

1. The source node sends a label request message through one path and sends x query messages through the other paths that seem to satisfy the constraints of the requesting flow. The ingress node initiates the Path Query messages. It populates the Query TLV Parameters according to what kind of information it wants to gather. The path query message could carry the list of hops from the ingress to the egress. This way, each node along the path can have a complete route from source to destination. This is useful for network management. If the Path Query message does not contain the ER TLV, it should be propagated by LSRs based on the knowledge that the LSRs have about the path.

2. Upon receiving a Path Query Message, LSR passes it to the downstream peer.

3. If the path query message arrives at LSR that has another LSP to the destination, this LSR initiates path partial query reply message containing its knowledge about the whole remaining partial path's inquired information and sends it back to the ingress node.

4. If for any reason the LSR couldn't forward the path query message to the down stream LSR, it sends partial path reply to the ingress node.

5. Upon receiving the Path Query Message, the egress node has to reply with a Path Query Reply Message. The Path Query Reply Message contains the Query TLV, which was received in the Path Query Message. The Query TLV tells the LSRs along the path which information is being queried and allows intermediate LSRs to attach their own queried information on the Path Query reply message.

6. When it receives a feedback in failure from the label request message, the ingress node times the path query reply messages out before taking the proper decision based on the feedback it already received.

A.2 Reply Messages

These messages are propagated upstream. There are two types of reply messages:

- Path Query-Reply message.

- Partial Path Query-Reply message.

The Reply messages carry the queried information upstream. A Path Query-Reply Message is sent in response to a Path Query Message. The ingress initiates the Path Query Message to gather the information from all the nodes along the queried paths. However, the egress node sometimes can't be reached. In these cases it would be better if the ingress LSR gathered information up to the point of failure. The Partial Query-Reply Message provides this mechanism. It is recommended to use the Partial Query-Reply Messages when a Query message fails.

Both reply messages are described in the following sections.

A.2.1 Path Query-Reply Message encoding

The egress LSR generates this message and propagates it upstream. Each intermediate LSR along the path propagates it upstream.

The encoding for the Path Query-Reply message is:

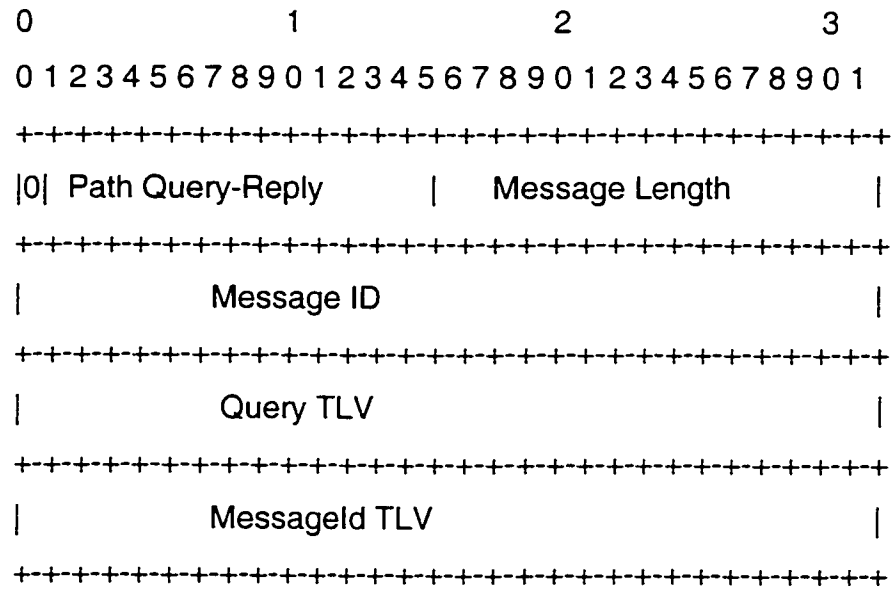

Figure A.2.1 path query reply message format

Message ID

32-bit value used to identify this message.

Query TLV

What is to be queried? See [8] Section 7 - Query TLV - for encoding.

MessageId TLV

The value of this parameter is the message id of the corresponding Path Query message.

A.2.2 Path Query-Reply Message Procedures

A Path Query-Reply message is initiated by an egress node, which receives a path Query message. Upon receiving the Path Query message, the egress node has to reply with a Path Query-Reply message. The egress node has to encode into the Path Query-Reply message a MessageId TLV. The mapping between a Path Query and a Path Query-Reply Message is done based on the message id. Besides the MessageId TLV, the egress has to encode the information that was queried (bandwidth, etc).

After the encoding is done, the Path Query-Reply message is sent back, on the reversed path, towards the ingress. Every LSR across the path has to encode its information according to what query flags are set.

A.2.3 Partial Path Query-Reply Message encoding

The Partial Path Query-Reply message is initiated by any LSR along the queried path. The message is generated only if the following rules apply:

- If the LSR couldn't forward the Path Query message to the down stream peer, or

- If the LSR has LSP that traverses the same queried path to the destination.

The encoding for the Partial Path Query-Reply message is identical to the Path Query-Reply, except the message type. Figure A.2.3 shows the message format.

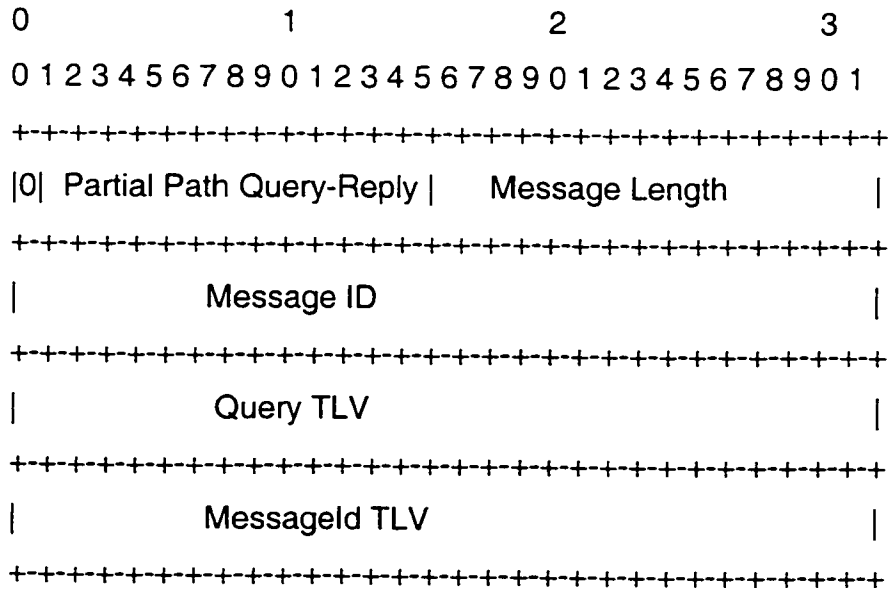

Figure A.2.3 Partial Path-Query Reply message format

A.2.4 Partial Path Query-Reply Message Procedure

The procedure is similar to a Path Query-Reply procedure. The only difference is that the node that initiates the reply is not the egress node. Upon receiving a Path Query Message, and not being able to forward it downstream or it is well informed about that path, and if the LSR supports partial replies, it has to create a Partial Path Query-Reply and encode the queried data and send it upstream like any Query-Reply messages, after it attaches its knowledge about the remaining path inquired information if it has LSP traversing the same path. The ingress receives the messages and attaches the information to its topology database to be considered in the future use.