SIMULATION AND EVALUATION OF PAP: A NEW ROUTING ARCHITECTURE FOR DIFFERENTIATED SERVICES DOMAINS HANDLING ”EXPEDITE FORWARDING” TRAFFIC CLASS

By

HRISHIKESH NULKAR

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2004

ACKNOWLEDGMENTS

I would like to acknowledge the supervision of Dr. Shigang Chen and his extremely helpful guidance throughout this work.I would also like to thank Dr. Richard Newman and Dr. Ye Xia for serving on my supervisory committee and reviewing my work.

I would like to thank all my friends who have helped me directly or indirectly in this work.

Finally, I am grateful and indebted to my parents who have helped me become what I am today, at this stage of my life.

TABLE OF CONTENTS

LIST OF TABLES

Table page

LIST OF FIGURES

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

SIMULATION AND EVALUATION OF PAP: A NEW ROUTING ARCHITECTURE FOR DIFFERENTIATED SERVICES DOMAINS HANDLING "EXPEDITE FORWARDING" TRAFFIC CLASS

By

Hrishikesh Nulkar

May 2004

Chair: Shigang Chen
Major Department: Computer and Information Science and Engineering

The research paper by Dr. Shigang Chen *et al.* proposes a new routing architecture (PAP) for differentiated services domain that integrates admission control signaling and the QoS routing. It differs from traditional routing in its ability to route most expedite forwarding (EF) traffic along the shortest possible paths while making use of alternative paths to absorb transient overload. Once an EF data flow is admitted, its performance is assured. The overhead for storing alternative path information is minimal since only one routing entry at a branching point is needed for each alternative path.

This thesis focuses on a simulation implemented to demonstrate the above proposed routing architecture and compare it with the traditional routing protocols. A GUI has been designed and implemented to show the simulation results graphically on the generated topologies. It displays the paths generated by the new routing protocol and by the shortest path routing protocol for different bandwidth requirements, different sources and destinations and compares them on the basis of specific parameters.

CHAPTER 1
INTRODUCTION

As the effort of converging voice and data into a single network accelerates, a wide range of multimedia applications emerges. The requirement for timely delivery of digitized audio-visual information raises new challenges for broadband networks. It is a challenging problem to commit network resources in a scalable way so that the delay or throughput sensitive traffic is appropriately treated as they are routed through the network. Different system functions like Admission control, QoS routing should co-operate with each other to achieve this. Admission control ensures that the total traffic in the network does not overwhelm the available resources. Traditional routing protocols make sure that the packets get to their destinations, while QoS routing protocols make sure that the QoS traffic is well spread on different paths to increase the utilization of the network's capacity. Packet scheduling and resource management at the routers allow differentiated treatment for packet streams with varied service requirements

1.1 QoS Technologies

QoS support roughly falls in two broad categories: Integrated Services (IntServ) and Differentiated Services (DiffServ). IntServ supports protocols like RSVP where the required resources are reserved at every router along the path of traffic stream to guarantee the performance of a traffic stream. But this has a drawback of storing per stream information at every core router and for even millions of such stream going through and hence is not so scalable.

DiffServ solves this problem by pushing the complexity to the edge of the network, where the data traffic is classified into different service classes by setting a code point in the IP header of every packet. Inside the DiffServ domain, the packets are treated according to the service classes they belong to. In this way, the per-stream information is eliminated inside the domain. The resource management is conducted at a course level based on service classes. In spite of this, routing support for DiffServ is still an open problem. Although the design of DiffServ is independent from routing, the routing function has a significant impact on the effectiveness of some classes like the expedite forwarding (EF). The routing function directly affects the admission control, which determines the traffic volume that a DiffServ network can accommodate for each service class.

1.2 The New Routing Architecture (PAP)

In the research paper by Dr. Chen *et al.*[2], a new routing architecture for DiffServ domains has been compared to traditional routing protocols like the shortest path routing. Shortest path routing is good for best effort traffic but too restrictive for QoS traffic. Before a traffic stream is delivered, the sender or the receiver activates the QoS routing protocol to establish a routing path that has the required resources. A routing entry is inserted at each router on the path thus depositing per stream information at each router. Another option is to carry the routing path in the source-routing IP option which causes excessive overhead on the routers. The core idea of the paper focuses on the fact that there might be several other paths other than the shortest path that might support the required QoS when the shortest path cannot.

1.2.1 The Protocol

The paper by Dr. Chen *et al.*[2], thus proposes a new architecture for routing of particularly the Expedite Forwarding traffic class for DiffServ domains. It relies mainly on the destination-indexed shortest path called as the primary path. But when the primary path is saturated and cannot support more traffic without QoS degradation, temporary alternate paths are established on demand. The traffic on these alternate paths will switch back to the primary path whenever it is again able to support the traffic. The major advantage of this architecture is that no per stream routing information is needed in normal conditions and in overloaded conditions, alternate paths are used to absorb the overloaded traffic and only one extra entry is needed at the branching point to store an alternate path. The admission control signaling is an integrated component in the new architecture and the existing QoS routing protocols find their place in the DiffServ world.

1.2.2 The Simulation

A simulation based on a simple DiffServ network model can demonstrate the advantages described above. This thesis focuses on developing a GUI based on the simulation to model the topology graphically, select the required source and destination nodes in the network, select and compare the traditional shortest path routing and the new architecture (PAP) based on certain bandwidth requirements for thousands of requests generated. A simulation has been implemented in C++ to demonstrate the new routing architecture, creating several topologies by the Waxman model, generating thousands of requests for each by selecting a random source and destination node, and then comparing the new architecture with the traditional routing protocols. The GUI depicts this comparison graphically displaying paths for a single Waxman topology.

1.3 Organization of Thesis

The thesis is organized as follows: Chapter 2 provides an overview of differentiated services and QoS routing. Chapter 3 introduces the proposed routing architecture (PAP). It also describes admission control and QoS routing in the new architecture. Chapter 4 describes the implementation of this architecture in the form of a simulation (in C++) and a GUI developed for this simulation (Using MFC[14, 10]). It also describes the comparison of the new architecture and the shortest path routing based on the simulation results. Chapter 5 gives conclusions and discusses scope for further research in this area.

CHAPTER 2
QOS ROUTING AND DIFFERENTIATED SERVICES NETWORKS

2.1 QoS Routing Overview

Today, network traffic is highly diverse and each type of traffic has unique requirements in terms of bandwidth, delay, loss, and availability. The IP protocol was originally designed to reliably get a packet to its destination with less consideration to the amount of time it gets there. Many of the applications supported over IP require low latency and the end-user quality may be significantly affected or in some cases, the application simply does not function at all. $VoiceOverIP$ could be a good example of an application which requires this type of behavior (low and fixed amount of delay with minimum loss), to function properly. The best-effort IP network introduces a variable and unpredictable amount of delay to the voice packets and also drops voice packets when the network is congested. QoS techniques could be applied to this best effort IP network to make it capable of supporting VoIP with acceptable, consistent, and predictable voice quality[9].

QoS based routing has been regarded as an essential mechanism for providing QoS guaranteed services in the Internet[15]. Its primary aim is to manage limited resources of IP networks efficiently. It aims to achieve two key goals:

- Improving network resource utilization,
- Achieving user's QoS satisfaction.

Networks are built by concatenating network devices such as switches and routers. They forward traffic among themselves using interfaces. If the rate at which traffic arrives at an interface exceeds the rate at which that interface can forward traffic to the next device, then congestion occurs[8]. Thus, the capacity of an interface to forward traffic is a fundamental network resource. QoS mechanisms work by allotting this resource preferentially to certain traffic over other traffic.

In order to do so, it is first necessary to identify different traffic. Traffic arriving at network devices is separated into distinct flows via the process of packet classification. Traffic from each flow is then directed to a corresponding queue on the forwarding interface. Queues on each interface are serviced according to some algorithm. The queue-servicing algorithm determines the rate at which traffic from each queue is forwarded, thereby determining the resources that are allotted to each queue and to the corresponding flows. Thus, in order to provide network QoS, it is necessary to provision or configure the following in network devices:

1. Classification information by which devices separate traffic into flows,
2. Queues and queue servicing algorithms that handle traffic from the separate flows.

This thesis focuses on one such traffic handling mechanism called Differentiated Services.

2.2 Overview of Differentiated Services

The growth of business requirements and new applications has set a clear need for simple methods of providing differentiated classes of service for Internet traffic. Differentiated Services is a modular, high performance, incrementally deployable and scalable approach on the way of developing end-to-end QoS of Internet[5]. It is a set of technologies that allow the network service providers to offer different kinds of services to different customers and their traffic streams.

DiffServ follows the philosophy of mapping multiple flows into a few service levels, sometimes referred to as Class of Service (CoS). DiffServ does not define signaling mechanisms; instead, packets are marked with a DiffServ Code Point (DSCP), which provides information about the QoS requested for a packet[6]. The code point enables network routers to handle IP packets differently depending on

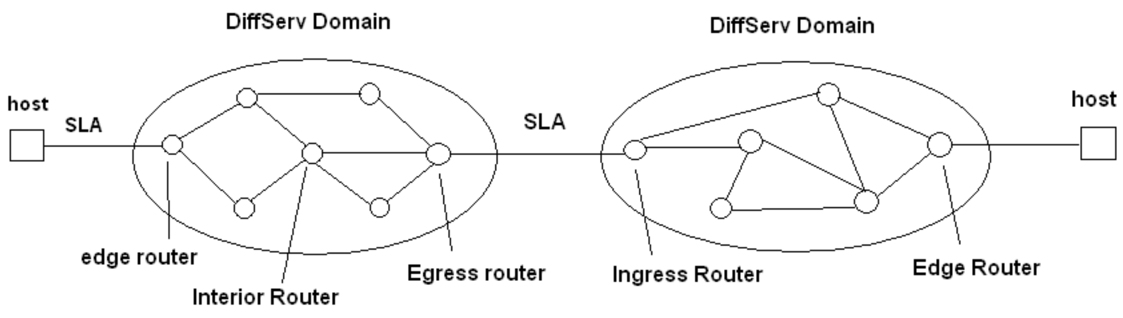


Figure 2–1: The differentiated services domain architecture showing the types of DiffServ routers

the code point and hence the relative priority. DiffServ code points are located in the 8-bit Type of Service (TOS) field in the IP header in the lower order 6 bits.

Differentiated Services is a combination of

1. Marking packets with a DSCP at boundary nodes;
2. Using the DSCP to determine how packets are forwarded by interior nodes;
3. Conditioning packets at boundary nodes.

2.2.1 Types of DiffServ Routers

There are three types of routers in a DiffServ domain[2]:

1. Edge Routers,
2. Interior Routers,
3. Ingress and Egress Routers.

Figure 2–1 shows the three types of routers in the Differentiated Services domain architecture.

Figure 2–2 shows the functions of the the different types of routers in the Differentiated Services domain.

Edge routers. As shown in Figure 2–2, an edge router is at the boundary of a DiffServ domain. It negotiates and enforces a Service Level Agreement (SLA) with a customer. SLA may consist of parameters like maximum packet loss,

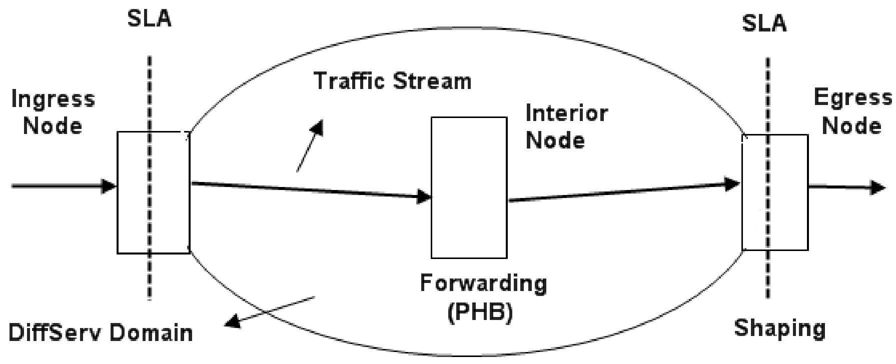


Figure 2–2: The differentiated services domain architecture showing the types of DiffServ routers and their functions in the domain.

maximum packet delay. The SLA specifies the terms and conditions of the service being offered. The edge router implements shaping, routing, policing, packet classification, marking, monitoring and other functions.

Ingress and egress routers. Between two domains, the router from which the traffic leaves a domain is called an egress router, and the router from which traffic enters a domain is called an ingress router. These are responsible for ensuring that incoming traffic conforms to the SLA between the two domains. Thus most of the workload is shifted on to the edge routers and the interior core routers remain simple.

Interior routers. The interior core router only needs to know how to handle a few traffic classes rather than keeping knowledge about thousands of individual traffic streams. In this way, per-stream information is eliminated inside the domain. They do not implement traffic conditioning. Resource management is conducted at a course level based on service classes. Differentiated services can thus alleviate core network and boundary router bottlenecks by better managing high priority traffic.

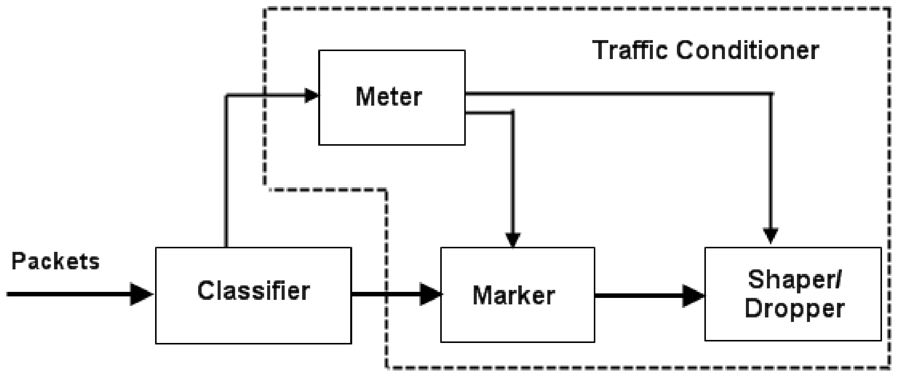


Figure 2–3: The architecture of a DiffServ router with its internal components

2.2.2 DiffServ Router Architecture

A DiffServ router consists of five components as shown in Figure 2–3, a packet arrives at the classifier and will be classified according to the bilateral service level agreement[6]. The classifier forwards the packet to the traffic conditioner. The traffic conditioner may include a meter, a marker, a shaper and a dropper.

- *Meters* measure the temporal properties of the stream of packets selected by a classifier against a traffic profile specified in a Traffic conditioning Agreement (TCA.
- *Shapers* delay some or all of the packets in a traffic stream in order to shape the stream into compliance with a traffic profile.
- *Droppers* discard some or all of the packets in a traffic stream in order to bring the stream into compliance with the traffic profile. This process is known as policing the stream.

2.2.3 Diffserv Per-Hop Behaviors

The service of DiffServ is realized by mapping the DSCP contained in the IP packet header to a per-hop Behavior (PHB) at each network node along the path of the packet. There are two primary PHBs defined for DiffServ networks[6]:

1. Expedite Forwarding
2. Assured Forwarding

Expedite forwarding. EF is a service with low loss, low latency, low jitter and guaranteed bandwidth[1]. It has the following properties: peak bit rate on flows or on aggregated flows, no bursts, only within the peak bit rate, and low queuing delay. The idea is to always keep the total EF traffic passing through any link in the network under a limit, which is set to be smaller than the link bandwidth. Traffic that exceeds this limit must be discarded. A simple priority queue is then used to schedule EF packets before packets from other service classes. If EF PHB is implemented by a mechanism that allows unlimited preemption of other traffic, the implementation must include some means of to limit the damage EF traffic could do to the other traffic. Since the receiving rate of EF traffic is always smaller than the sending rate at every router, EF traffic is guaranteed for minimized delay and assured bandwidth. In order to provide this high service level in practice, the amount of traffic injected into the EF class needs to be carefully policed. EF PHB is in charge for guaranteing a minimum departure rate for some traffic and this rate is independent of the rest of the traffic to be forwarded by the same router at the same time. Admission control and routing are essential to make sure that EF traffic never exceeds the link bandwidth. It is most idly suited for VoIP.

Several types of queue scheduling mechanisms may be employed to deliver the forwarding behavior described above and thus implement the EF PHB. A simple priority queue will give the appropriate behavior as long as there is no higher priority queue that could preempt the EF for more than a packet time at the configured rate.[1]

[1] This could be accomplished by having a rate policer such as a token bucket associated with each priority queue to bound how much the queue can starve other traffic

The host that sends out an EF stream is called the source host. The edge router that the source host connects to is called the source router. The host that receives the EF traffic stream is called the destination host. The edge router that the destination host connects to is called the destination router. An EF traffic stream is a one-way traffic stream. Two EF streams in opposite directions model two-way traffic. Two-way traffic is admitted into the network only after its two EF streams are both accepted by the admission control[2]. In the paper and this implementation, only a single EF traffic stream has been considered.

Assured forwarding. AF does not provide bandwidth guarantee but packets are given a higher priority. It is mainly used for delay insensitive traffic. The AF PHB group defines the dropping precedence among different classes of AS traffic when network congestion occurs[6]. Queue management is more essential to the implementation of AF service.

2.3 Summary

The chapter describes the need for QoS routing to route delay sensitive traffic through the network with guaranteed bandwidth and minimum delay. Differentiated Services is one such traffic handling mechanism that does provide the classification of network traffic and provides QoS guarantee for each class according to its requirements, by differentiating packets on the basis of Code Points at the edge routers. It scores over IntServ by eliminating per stream information to be deposited inside the domain and thus making interior routers simple.

Expedite Forwarding service is the focus of this thesis since it provides guaranteed QoS if admission control and routing are properly done. Clearly EF traffic has distinctive characterization from AF traffic. Hence the new routing architecture described in the paper by Dr. Chen *et al.*[2] and its implementation in this thesis is concerned mainly for EF traffic because of the important role that routing and admission control plays in efficient working of EF PHB. The next chapter will discuss the new routing architecture for DiffServ domains.

CHAPTER 3
PAP: A ROUTING PROTOCOL FOR DIFFSERV DOMAINS

This chapter discusses a new routing protocol that can be used with the proposed routing architecture for Differentiated Services domains[2].

3.1 Overview

Expedite forwarding traffic can be routed by the traditional routing table (TRT) as is done in the case best-effort traffic. The traditional routing tables provide a single path between each pair of nodes. If the path is overloaded by the EF traffic, the TRT approach lacks the flexibility of using alternate paths.

As proposed in the research paper, the new routing architecture primarily uses TRT to route the EF traffic but relies on alternate paths to handle the overload condition. The alternate paths are stored in the QoS routing tables (QRT) which are constructed on demand by the QoS routing protocols. The QoS routing protocols are invoked only when the EF traffic overloads the primary routing path. Under normal conditions when the network is not overloaded, the system will not see the existence of the QoS routing protocols.

When an EF stream arrives, the source host issues a request to the source router. The source router initiates the admission control signaling between the source router and the destination router. A REQUEST message is sent towards the destination router to check the bandwidth availability along the path. In the proposed routing architecture, all control messages and non-EF data packets are routed along the primary paths by TRT in the same way the current IP networks route packets.

As a router on the primary path receives the REQUEST, it performs a simple acceptance test to check if it has enough bandwidth for the EF traffic.

If it does, the REQUEST is forwarded to the next hop on the primary path. If the acceptance tests of all intermediate routers are passed and the REQUEST successfully arrives at the destination router, it means that the primary path can support the new EF traffic. The destination router sends an ACCEPT message to the source router, which in turn notifies the source host to start sending data packets. The data packets will be routed by TRTs and follow the primary path to the destination host.

On the other hand, if the acceptance test fails at an intermediate router, which means the primary path can not support the traffic, then a QoS routing protocol is triggered to find an alternative path. If an alternative path that supports the traffic is found, an ACCEPT message is sent to the source router and the data packets of the EF stream will follow the alternative path. If an alternative path cannot be found, a REJECT message is sent to the source router, which will either reject the EF traffic or retry the admission control at a later time.

The alternative paths are stored in QRTs. In order to keep the size of the QRTs small, traffic using an alternative path merges back to the primary path when there is sufficient bandwidth freed up on the primary path. The difference between the Traditional routing table and QoS routing table is discussed, noting how data packets are forwarded by these routing tables.

3.2 TRT and QRT

Each router has one TRT maintained by the traditional routing protocols such as RIP, OSPF, IGRP, and/or BGP. In addition, it has a QRT for each network interface. The QRTs are maintained by the QoS routing protocols. The reason to use multiple QRTs instead of one for the entire router is to reduce the size of each QRT. It is clear that each incoming packet will be matched against one QRT, smaller table size results in faster processing.

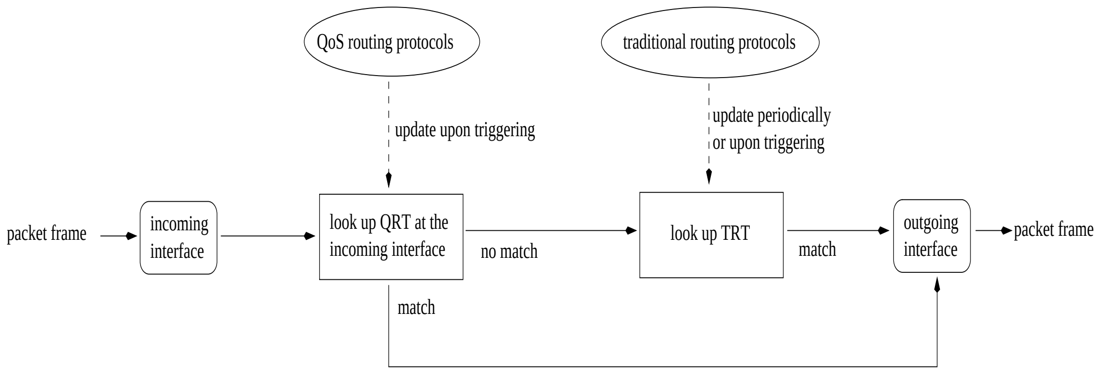


Figure 3–1: TRT and QRT: When an EF packet arrives, the QRT at the incoming interface is first looked up. If there is a match, the packet is forwarded to the next hop and TRT is not checked; otherwise the TRT is looked up

TRT is indexed by destination IP addresses. Each TRT table entry consists of a destination IP address, a next-hop IP address, and other information. The outgoing interface to which a packet is forwarded can be determined from the next-hop IP address. QRT is indexed by EF traffic identifiers. An EF traffic identifier is composed of a source IP address, a destination IP address, a protocol identifier, a source port number, and a destination port number. Each QRT table entry consists of an EF traffic identifier, a next-hop IP address, and other information. In this architecture the route map needs to be dynamically updated by QoS routing protocols in order to support EF.

For all non-EF packets, only TRT is looked up to find the next hop, which is exactly what the current IP networks do. Figure 3–1 illustrates how an EF packet is forwarded at a router. After the packet arrives at the incoming interface, the QRT at that interface is looked up. If there is a matched table entry, the packet bypasses TRT and proceeds directly to the outgoing interface which sends the packet to the next hop. If there is not a match in the QRT, the TRT is looked up and the default shortest-path is used.

The main objective is to minimize the size of the QRT. Most of the EF traffic is kept on the primary path. If the network is in normal conditions without any congestion, all EF traffic will travel along the primary path and the size of QRT will be zero. In this case, only TRT is looked up for all packets. On the other hand, when the aggregated EF traffic on a primary path reaches the maximum allowed quota for EF traffic, a QoS routing protocol will be triggered to find alternative paths for new EF streams. New table entries are inserted into QRT for the duration of the traffic streams. It should be stressed that only the overload portion of the EF traffic is routed via QRT along the alternative paths, and this portion of the traffic will switch back to the primary path whenever possible.

3.3 Admission Control

The task of admission control is to determine whether a new EF traffic should be accepted into the network or rejected. In our routing architecture, the admission control and the QoS routing is closely related. In fact, if the admission control function finds out that the primary path does not support the new traffic, it will trigger the QoS routing function to find an alternative path. If the QoS routing function finds such a path, the admission control accepts the traffic, otherwise, it rejects the traffic.

Admission control is done only for the EF traffic, which receives assured bandwidth and fast forwarding under our routing architecture. The signaling process starts from the source router and follows the primary path towards the destination router. The signaling message, REQUEST, carries:

- The traffic identifier (source IP address, destination IP address, protocol identifier, source port, and destination port),
- The service class identifier (EF),
- The bandwidth requirement B.

In order to assist the admission control, each router keeps track of its resource availability. Two variables are maintained for each outgoing interface:

- EF_{max}: EF_{max} is the maximum sustainable bandwidth allowed to be used for sending EF traffic from this interface

- EF_{agg}: EF_{agg} is the aggregated bandwidth currently used by all EF traffic on this interface.

EF_{max} is set at the configuration time, while EF_{agg} is measured at run time. When a router receives a REQUEST, it uses TRT to find the outgoing interface to the next hop on the primary path. A simple acceptance test is performed at the outgoing interface to see if there is enough bandwidth for the new traffic.

$$\text{If } B \leq EF_{max} - B_{agg},$$

the REQUEST is sent to the next hop.

$$\text{If } B > EF_{max} - B_{agg},$$

the traffic can not be admitted by using this outgoing interface. A QoS routing protocol is triggered to find an alternative routing path. If an alternative path is not found, a REJECT message is sent to the source router that rejects the traffic or retries the admission control after certain delay. On the other hand, if the REQUEST successfully arrives at the destination router or the QoS routing protocol finds an alternative path, an ACCEPT message will be sent to the source router. The source router will notify the source host to send data traffic.

3.4 QoS Routing

If the primary path has the bandwidth to support the new EF traffic stream, the QoS routing protocol is not triggered by the admission control. All data packets will follow the primary paths directed by TRT because there is no matched table entry in QRT. Under congestion conditions, however, an intermediate router may not have the required bandwidth. As shown in Figure 3–2 (b), suppose i fails the acceptance test at the outgoing interface connecting to j. The corresponding link (i, j) is called an infeasible link, which is represented by a dotted line. A link that passes the acceptance test is called a feasible link. The admission control signaling can not proceed along link (i, j), and a QoS routing protocol is activated.

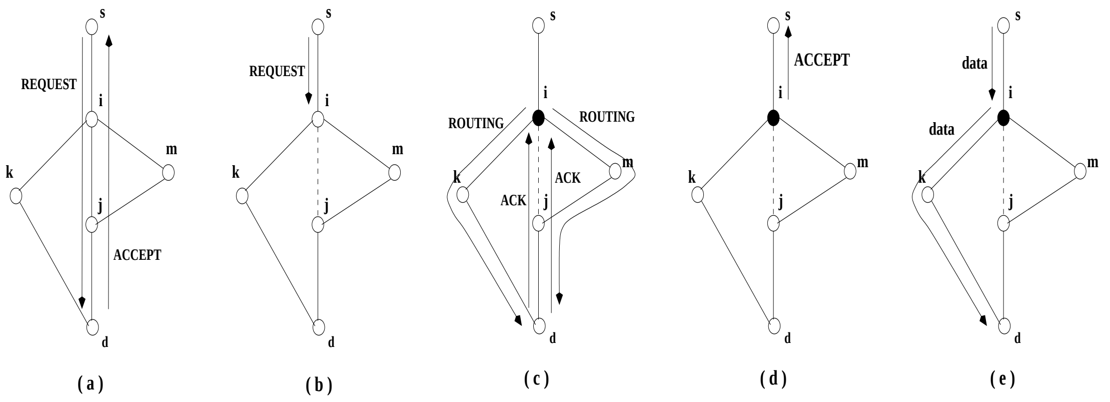


Figure 3–2: QoS routing

The proposed routing architecture is independent of any particular QoS routing protocol. For the purpose of completeness, only one QoS routing protocol has been implemented in this thesis to show how it fits in the architecture. One big advantage of the protocol is that it relies only on the local state stored at each router, which makes it scalable and easy to implement. The basic idea is as follows: When an infeasible link is encountered, it is considered as an indication of local congestion. The QoS routing protocol tries to find an alternative path by detouring around the infeasible link. Without the knowledge about the extent of the congestion, the protocol branches out towards multiple directions and searches multiple paths for one that can support the new EF traffic.

In Figure 3–2 (c), i sends out ROUTING messages along all adjacent feasible links. i is called the branching point. Apparently, a ROUTING message should not be sent to the link from which the REQUEST was received, and it will not be sent to (i, j), which is an infeasible link. A ROUTING message carries two IP addresses: bpAddr, which is the address of the branching point, and nbAddr, which is the address of the neighbor that receives the message. It also accumulates the delay of the path it traverses.

After a ROUTING messages arrive at the neighbor node k (or m), it is routed by TRTs from there on. Hence, the ROUTING message follows the primary paths from k (or m) to the destination router. This has a very important implication: in order to store an alternative path, it is sufficient to add a single table entry in the QRT at i to direct traffic to k (or m). From k (or m) on, TRTs will be used. Similar to REQUEST, a ROUTING message causes an acceptance test to be performed at every router it traverses.

The ROUTING message is sent to the next hop only if the acceptance test is passed. Therefore, if the ROUTING message successfully reaches the destination router, an alternative path for the new EF traffic is found, the path that the message has just traversed is. In this case, an ACK is sent back to the branching point i, whose IP address is *bpAddr* that was carried in the ROUTING message. The ACK copies *nbAddr* and the accumulated path delay from the ROUTING message. Upon receipt of the ACK, i inserts a table entry in the QRT at the incoming interface from which the REQUEST was received. The next hop in the entry is set to be *nbAddr*. Also stored in the entry is the delay carried back by ACK, which is the total expected delay for EF traffic on the alternative path. In addition, i sends an ACCEPT message to the source router to admit the traffic.

As Figure 3–2 (d) shows, the source router will notify the source host. As shown in Figure 3–2 (e), the data packets follow the primary path until it reaches the branching point i, where the QRT at the incoming interface directs the packets away from the primary path. Once the packets reach the neighbor router k, they are again routed by TRTs. If multiple ROUTING messages arrive at the destination router, then multiple alternative paths are found and multiple ACKs are sent back to the branching point. When the branching point receives an ACK and finds that there is already a table entry in the QRT for the EF traffic stream, it checks if the delay in the ACK is smaller than the delay in the entry. If it is smaller, the next hop in the entry is replaced by the *nbAddr* value in the ACK. Therefore, the best found alternative path will be used. [1]

When a router receives a ROUTING message, if the acceptance test fails, it sends a NACK message back to the branching point. If the branching point receives a NACK from every ACK sent out, it concludes that the QoS routing protocol fails in finding an alternative path. A REJECT message is sent back to the source router, indicating that the traffic can not be admitted at this moment.

The above routing protocol allows only one branching point to deviate from the primary path. More sophisticated design may allow multiple branching points. When a ROUTING message reaches a router that fails the acceptance test, the router can branch again and send ROUTING messages to the neighbors other than the one pointed by TRT. In such a design, every branching point needs a table entry in QRT in order to store the alternative path

In order to cancel the alternative path and switch back to the primary path whenever possible, the branching point periodically sends CHECK messages down the primary path to see if the primary path can now support the EF traffic stream. It is a mini-version of admission control. If the CHECK message passes the acceptance test at all intermediate routers and successfully reaches the destination router, the source router will be instructed to not send KEEPALIVE. The EF traffic will automatically follow the primary path when the alternative path times out.

[1] An ACK for an alternative path with smaller EF delay may arrive later because the control messages are not EF traffic and hence may experience a different delay

3.5 Summary

The chapter describes the new PAP routing architecture for DiffServ domains in detail. The PAP QoS protocol can be used to find alternate routing paths for Expedite Forwarding traffic streams when the primary routing protocol fails at a branching point. Support for admission control and QoS routing has also been integrated in the new architecture to route Expedite Forwarding traffic class with guaranteed bandwidth and minimum delay.

The next chapter describes a simulation that has been implemented to demonstrate the advantage of the above proposed routing protocol in use with the DiffServ domains. It compares the new protocol PAP with the traditional routing protocols like the shortest path routing with respect to the admission ratio (number of requests that are admitted) and also considers the performance affected due to the number of RT entries deposited per admitted flow on average. A GUI has also been described which will display graphically, the network topology, source and destination nodes selected and the paths as generated by the two compared protocols. It also shows the branching point in case when the acceptance test fails at a certain node along the shortest path and displays the QoS path (as generated by the new QoS protocol) from the branching point to the destination node.

CHAPTER 4
SIMULATION

This chapter describes the simulation and the GUI that has been implemented to demonstrate the advantage of the PAP QoS routing protocol in the new routing hierarchy for the differentiated services domains.

4.1 Overview

The simulation generates a number of network topologies based on the Power-Law model. Each link is assigned randomly a link success probability. It is the probability of the link that it will pass the acceptance test. A large link success probability corresponds to a light load condition thus increasing the probability that the particular link will be able to forward the traffic stream. A low link success probability on the other hand corresponds to a heavy load condition[2]. The network topology can be generated using either the $Inet$ or the $Waxman$ topology model, which are described later in this chapter. For each of the topologies generated, several thousand requests are generated which are routed from a specific source to destination. The source and destination are selected at random or can be input implicitly as a parameter. For this particular set of source and destination nodes, the new routing protocol PAP is compared with the shortest path routing protocol. The basis of comparison for the two protocols is:

- $Admissionratio$ (percentage of requests that are admitted)
- $AverageRTentries$ deposited per admitted flow(for PAP).

The GUI that has been implemented uses the same simulation design, but offers several facilities with respect to generation of the topology by allowing configuration of its parameters like the number of nodes, type of topology generator, the out degree exponent value for Waxman model, minimum and maximum link delay, minimum and maximum link bandwidth. It generates a single request for a specific source and destination node that can be selected by the user. It then displays the path as generated by the shortest path routing protocol. If at certain point along the shortest path, the acceptance test fails, then the QoS routing protocol is triggered and the remaining QoS path from the branching point to the destination is also displayed. If the bandwidth requirements are such that the existing links are not able to support the traffic request, then the routing fails and the path only up to the branching point is shown. It also displays the comparison results of the two protocols.The simulation is thus modelled in two parts.

1. Generate a single network topology by using the Waxman topology generation method, specify the source and destination for which the path is to be found, specify a single request with the bandwidth requirement for the traffic and select the protocol to be used.
2. Generate several network topologies based either on the Inet model or the Waxman model. Specify the number of requests to be generated along with the min and max bandwidth and delay values. The source and destination are taken at random for each request.

The comparison results for the second simulation part are shown in terms of the factors stated above.

A brief overview of topology generators and the power-law model has been provided to get a brief knowledge of topology generation methods used in the implementation and how the link success probabilities are assigned to the nodes in the network.

4.2 Network Topology Generation Methods

The internet can be decomposed into connected sub networks that are under separate administrative authorities. These sub networks are called as domains or autonomous systems. This way, the topology of the Internet can be studied at two levels of granularities. At the router level, each router is represented by a node and each link between the routers by an edge. At the inter-domain level, a node represents each domain and each edge is an inter-domain connection. Protocols are developed for both inter-domain and intra-domain communication. Network protocols are designed to be independent of the underlying network topology. While topology should not have an effect on the correctness of the protocol, it does have a major impact on the performance of the network protocols[11]. Hence topology generators are used often to generate realistic topologies in simulations. These topology generators do not aspire to produce the exact replicas of the current $Internet$, but they merely attempt to create network topologies that embody the fundamental characteristics of real networks[12]. There are mainly three categories of topology generators[11]:

1. Random,
2. Degree based,
3. Structure (hierarchical) based.

4.2.1 Metrics

The metrics that have been used to describe graphs are mainly the $nodeout-degree$ and the distance between the nodes. Given a graph, the out-degree of a node is defined as the number of edges incident on that node. The distance between two nodes is the number of edges of the shortest path between the nodes.

4.2.2 Random Topology Generators

The process of generating random topologies can be generally stated as follows:

Given an input of N nodes and a 2-dimensional plane of size n by m, first the placement of the nodes is to be decided[3]. The nodes can be distributed uniformly across the plane, or clustered around some region. The out degree for each node is then decided, which is the number of nodes that it is connected to. After the nodes are positioned and their out degree has been decided upon, the links are to be identified to decide which nodes should be linked to which other nodes.

The probability of creating an edge (*link*) between two nodes can be uniformly distributed or weighted by the Euclidean distance between them. The edges are then added to the network until all the nodes have their out-degrees satisfied.
A minimum spanning tree can be built prior to the generation of other edges to ensure that the graph is a connected graph. If the graph is disconnected then extra edges can be added to nodes at random to ensure that the graph is connected.

Waxman model. Waxman introduced one of the most popular network models. Waxman topology generator is a random graph topology generator. The generator produces random graphs based on the Erdos-Renyi random graph model, but it includes network specific characteristics such as placing the nodes on a plane and using a probability function to interconnect two nodes, which is parameterized, by the distance that separates them in the plane. The Waxman topology generator uses the Euclidean distance between the nodes to govern their connectivity. It starts by placing the nodes in a nxn plane. Once the nodes have been placed, it calculates the probability of creating an edge between two nodes u and v using the following probability function:

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

where

$P(u,v)$ is the probability of linking nodes u and v,

$d(u,v)$ is the Euclidean distance between u and v

L is the Euclidean diameter of the network [1] ,

α is the average out degree

β determines the average edge length

[1] Euclidean diameter is the maximum Euclidean distance between any two nodes in the graph

Then a random number is generated between 0 and 1. An edge is created between nodes u and v only if the random number is smaller than $P(u,v)$ which is calculated as above. Finally, a spanning tree can be connected to ensure that the resultant graph is connected. The use of Euclidean distance now makes the geographic distribution of nodes a factor in the topology. Thus Waxman is concerned only with randomly generated networks. Waxman has been used in this simulation GUI to generate such random network topologies with several nodes to demonstrate the new routing architecture.

Inet model. The Inet model generates a topology by placing N nodes on an nxn plane[3]. Each node is assigned and out degree based on Power Law 2. Then a full mesh is used to connect the top τ most connected nodes. For these nodes, 25% of their edges are connected randomly selected nodes with out degree 2. To create a fully connected topology, the remaining nodes are either connected to one of these nodes or connected to a node that can reach one of these τ nodes. The Inet-1.0 model has a second phase where the top most connected nodes are expanded into networks with τ nodes each. This phase is used to expand the top most connected autonomous systems into networks with router-level connectivity.

4.3 Simulation

4.3.1 Network Model

The network model used in the simulation is described below:

The simulation generates a random network topology based on the input parameters given. The first part of the simulation in which the paths(shortest path and/or QoS path) are graphically displayed, a single topology is generated using the Waxman topology generation method and a single request is considered. In the second part where the simulation is run with several topologies and several thousand requests, topology is generated using either the Waxman or the Inet topology generation method.

The simulation takes as input the following parameters to model the network topology:

- Number of nodes
- Number of requests
- Number of topologies [2]
- Topology generation type (Waxman / Inet)
- Minimum and Maximum bandwidth
- Minimum and Maximum delay
- For Waxman: the average outdegree α
- A random number seed which will affect the assignment of links in the Waxman model

The input parameters for a specific topology can be saved to a file and later the parameters could be read from the file in order to re-generate the same topology.

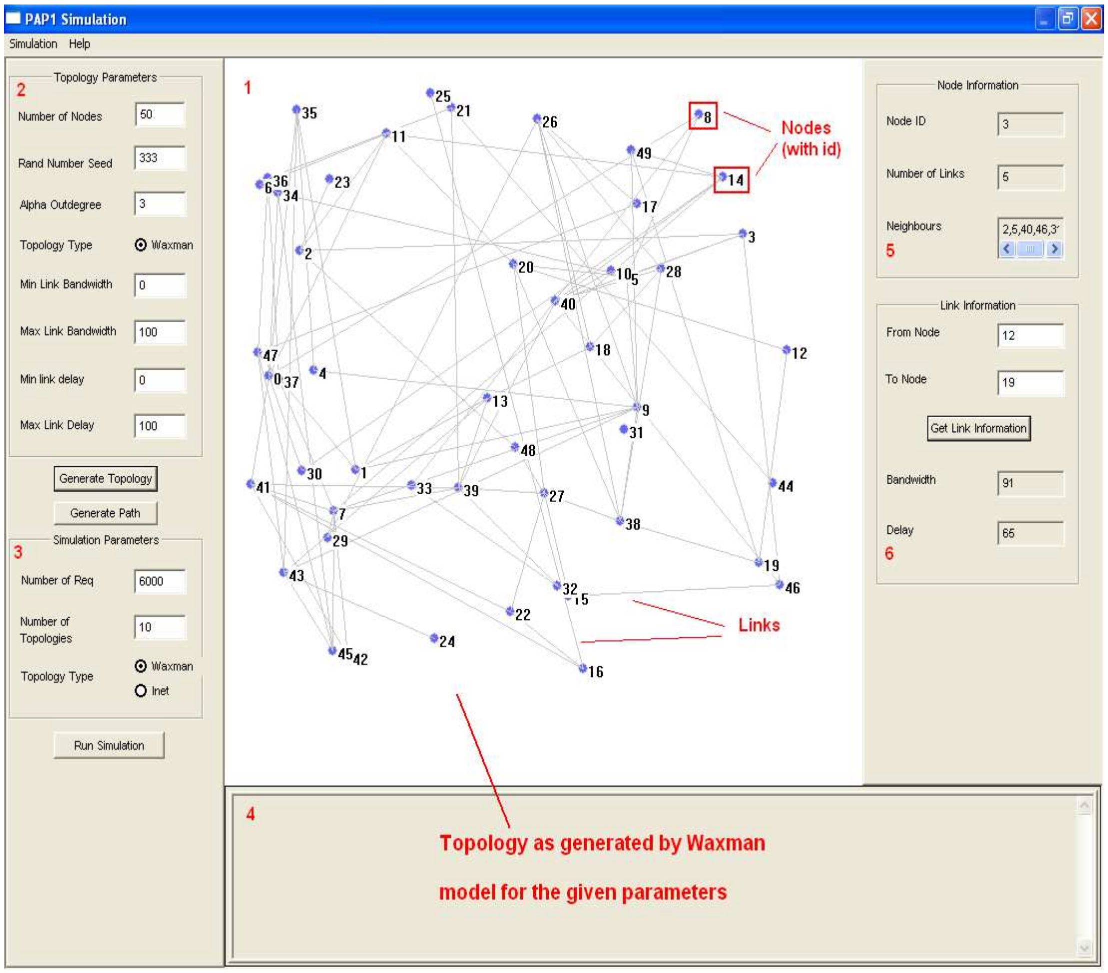
4.3.2 The Graphical User Interface

A view of the GUI that has been implemented to demonstrate the new routing architecture is shown in Figure 4–1.

The main parts of the user interface can be stated as

1. The main display window - The topology generated according to the given parameters is displayed in the main window.
2. Parameter Entry - The input parameters stated above for the topology generation are entered here
3. Multiple topology simulation parameters are entered below along with the choice of topology type (Waxman / inet) and the number of requests
4. A text window which displays the results of the simulation
5. The node information window which gives the properties of a selected node

[2] For the first part of the simulation, the number of topologies = 1 and number of requests = 1



Topology Parameters

Number of Nodes	50
Rand Number Seed	333
Alpha Outdegree	3
Topology Type	[✓ Waxman]
Min Link Bandwidth	0
Max Link Bandwidth	100
Min link delay	0
Max Link Delay	100

Simulation Parameters

Number of Req	6000
Number of Topologies	10
Topology Type	[✓ Waxman] Inet

Node Information

Node ID	3
Number of Links	5
Neighbours	2,5,40,46,3

Link Information

From Node	12
To Node	19
Bandwidth	91
Delay	65

Figure 4–1: A view of the GUI implemented with all the component windows

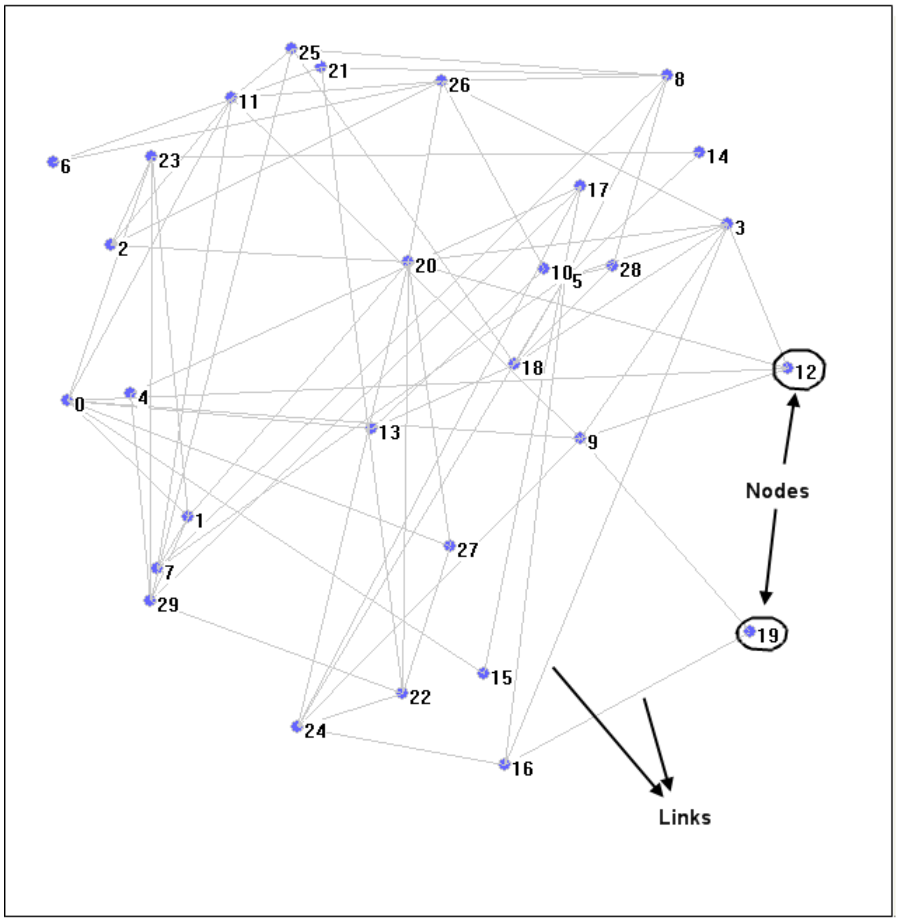


Figure 4–2: Network topology generated by Waxman method for 30 nodes with an average out-degree of 3

6. The link information window which gives the properties for a specified link

The individual parts of the user interface are now described in detail:

Topology display window. Depending upon the parameters entered for the simulation, a network topology is generated using the Waxman generation method. The nodes are placed in a $x - y$ plane corresponding to the blank white area seen on the display window.

Then using the Waxman probability function, the links are calculated and the nodes along with the links between them are displayed as shown in Figure 4–2. The nodes are represented by blue circles and the links between them by gray

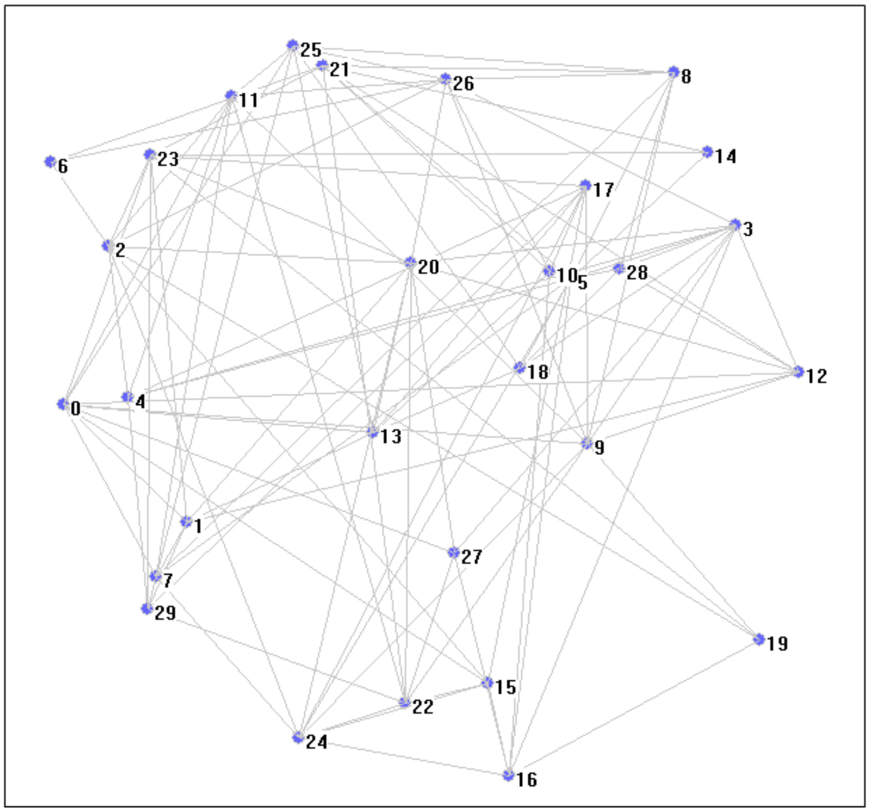


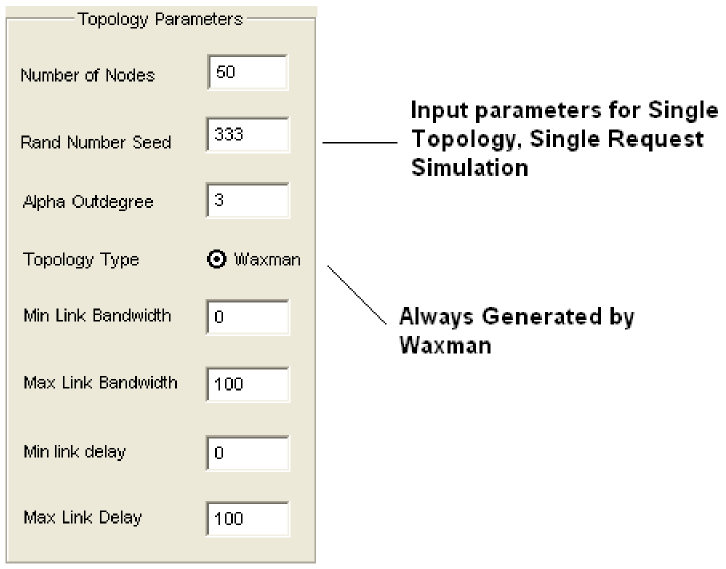
Figure 4–3: A denser network topology generated by Waxman method for 30 nodes with an average out-degree of 5

lines. Each node is assigned a unique identity with an integer starting from 0 to the number of nodes. Each node also has its corresponding ID displayed besides it.

As stated previously, a higher value for the α out degree value for Waxman will generate a denser network topology. This can be seen in Figure 4–3 which shows a topology with all the same parameters as in Figure 4–2 but with a α value of 5.

The Parameter entry window. The input parameters specified above will be entered by the user in the parameter entry window as shown in Figure 4–4. The default values for the input parameters are given in the following table

The bandwidth values for the links in the topology are assigned from the range $MinBandwidth - MaxBandwidth$ and the delay values are assigned from the range $MinDelay - Maxdelay$. The average out-degree value for Waxman is given



Topology Parameters

Number of Nodes: 50

Rand Number Seed: 333

Alpha Outdegree: 3

Topology Type: [✓ Waxman]

Min Link Bandwidth: 0

Max Link Bandwidth: 100

Min link delay: 0

Max Link Delay: 100

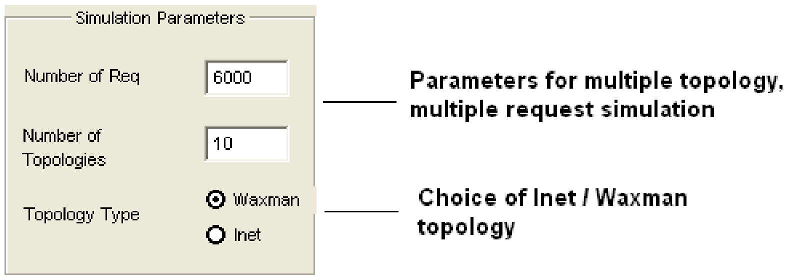
Input parameters for Single Topology, Single Request Simulation

Always Generated by Waxman

Figure 4–4: The parameter entry window with default values

Table 4–1: Default Values for Input Parameters

Parameter	Default Value
Number of Nodes	50
Random Number seed	333
Alpha out-degree (Waxman)	3
Minimum Link Bandwidth	0
Maximum Link Bandwidth	100
Minimum Link Delay	0
Maximum Link Delay	100


Simulation Parameters

Number of Req: 6000

Number of Topologies: 10

Topology Type: [✓ Waxman] Inet

Parameters for multiple topology, multiple request simulation
Choice of Inet / Waxman topology

Figure 4–5: The simulation parameter entry window with default values

by the parameter alpha. A higher value for α will result in denser topology while a lower value will give a sparse topology. These values are also used when multiple topology, multiple request simulation is run.

Simulation parameter entry window. Figure 4–5 shows the input boxes for the parameters required for the multiple topology, multiple request simulation. It takes as input the number of topologies to be generated, the number of requests and the choice of the topology generation method to be used ($Waxman/inet$). The other parameters are the taken from the parameter entry window as described above.

This generates a simulation with several topologies and chooses a source and destination node at random from the generated topology for each request. It generates several thousand requests as specified by the input parameter for that particular source and destination and then compares the results of the new protocol with the shortest path routing protocol, by calculating the averages of the comparison parameters for the thousands of requests. The default values for the window are given in the following table

Table 4–2: Default Values for Multiple Topology, Multiple Request Simulation Parameters

Parameter	Default Value
Number of Requests	6000
Number of topologies	10
Topology Type	Waxman

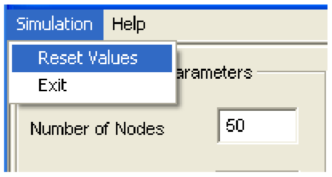


Figure 4–6: Reset parameter values to default for new simulation

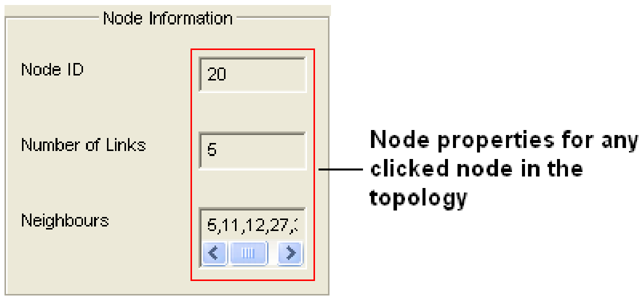


Figure 4–7: The node information window displaying properties of a clicked node

After a path is generated or the simulation is run, the values in both the parameter entry windows can be reset to the default values given in the tables by clicking "Reset" from the "Simulation" option on the GUI as shown in Figure 4–6

The Node information window. The node information window displays the following properties for a given node:

- Node ID
- Number of links for the node (its out degree)
- Its neighbors

The neighbors of the node are the IDs of the nodes in the topology to which it has a link. When any node in the topology is clicked, its corresponding properties would be displayed in the node information window. The window is shown in Figure 4–7.

The Link information window. The link information window accepts the start and end points of a node which are the node IDs of the nodes between which the

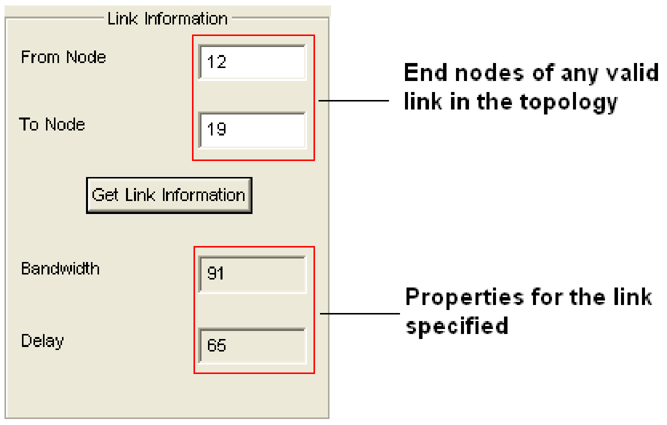

Link Information
From Node
12
To Node
19
Get Link Information
Bandwidth
91
Delay
65
End nodes of any valid link in the topology
Properties for the link specified

Figure 4–8: The link information window displaying properties of a specified link

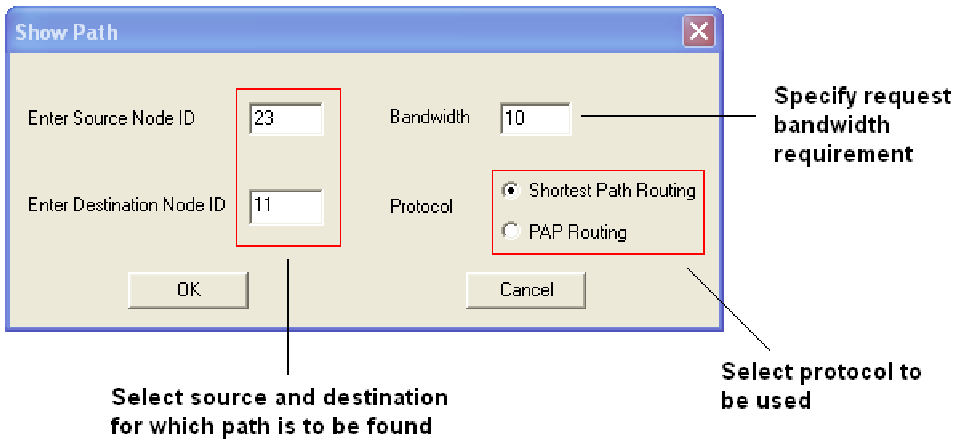
link exists. Given the "start" and "end" nodes(IDs) of a link, clicking the "Get Link Information" window will display the following properties of a link:

- Bandwidth
- Delay

The bandwidth for a link is taken from the range $MinBandwidth - MaxBandwidth$ and similarly the delay from the range $MinDelay - MaxDelay$, which are specified in the parameter entry window. This information is useful to check and verify whether a particular link can pass the acceptance test for the incoming request. The information can demonstrate how the new protocol chooses an alternate path over the shortest path when the shortest hop from the branching point cannot support the requested bandwidth. Figure 4–8 shows the link information window.

Selecting the source and destination nodes. For the single topology, single request simulation, once the topology has been generated, the source and destination for the generated request can be selected by clicking the "Generate Path" button on the parameter entry window.

The following window is displayed as shown in figure 4–9. The source and destination nodes can be specified along with the bandwidth requirement for the

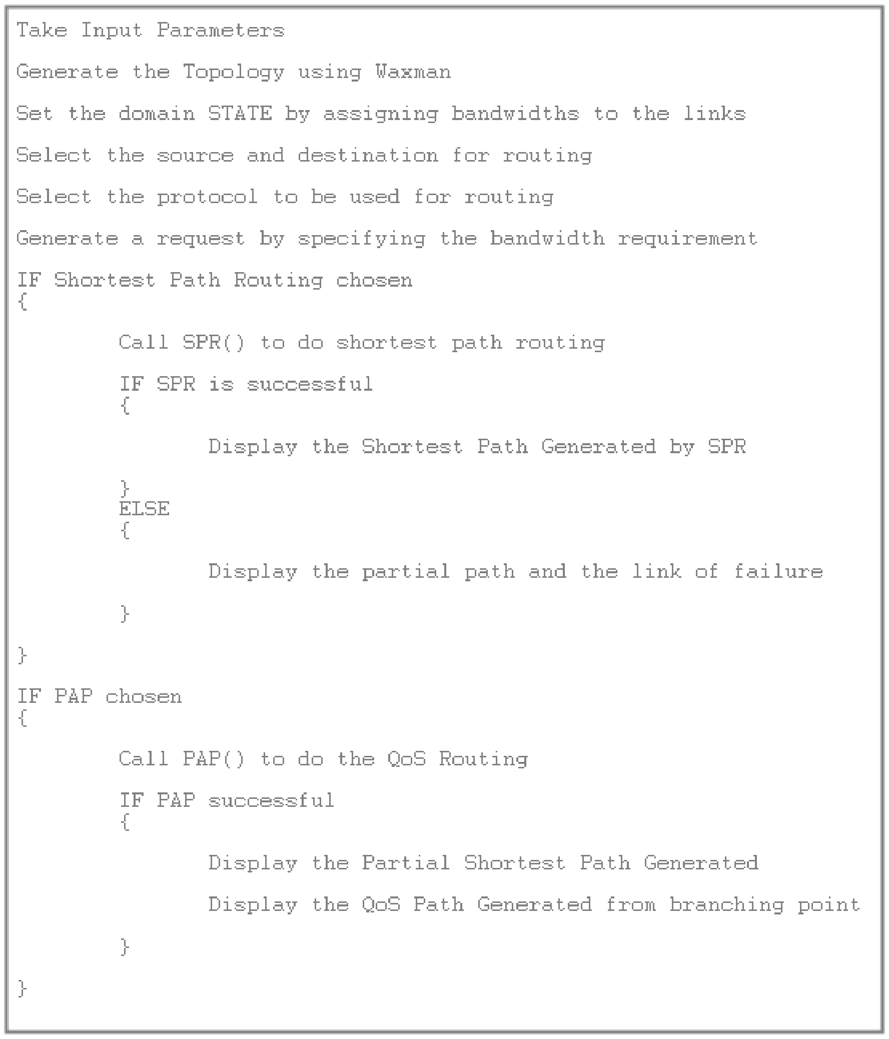

Show Path
OK
Cancel
Specify request bandwidth requirement
Select source and destination for which path is to be found
Select protocol to be used

Enter Source Node ID: 23
Enter Destination Node ID: 11
Bandwidth: 10
Protocol: [✓ Shortest Path Routing] PAP Routing

Figure 4–9: Selecting the source and destination nodes. The bandwidth requirement for the request is also specified.

request to be routed from the source to the destination. The request corresponds to an EF traffic stream to be routed from the source to the destination with the specified bandwidth requirement. The requirement states that every hop(link) on the generated path from the source up to the destination should have a bandwidth greater than the requirement specified, to ensure guaranteed delivery of the the EF traffic stream. The link that has a bandwidth less than the specified required bandwidth for the request cannot be used on the path.The protocol to be used for the routing can be chosen here.

- If $ShortestPathRouting$ protocol is chosen, then if any of the links in the shortest path calculated by the protocol, does not pass the acceptance test, then the protocol fails and the partial path from the source to the node of failure is shown. The link at which the test fails is also highlighted. This link is called as an $InfeasibleLink$ as stated in the PAP protocol description in chapter 3. An example later in this chapter will demonstrate such a case.
- If the new PAP QoS protocol is chosen, it starts of with calculating the shortest path between the source and destination nodes. But if any link along the path fails to satisfy the EF request, then the QoS protocol is triggered as described in chapter 3. It looks for alternates paths from the branching point which would satisfy the EF request. If it finds any, then the alternate path is shown along with the partial shortest path. If the protocol is not able to find even an alternate path, then only the partial path is displayed with the link of failure highlighted.



```
Take Input Parameters

Generate the Topology using Waxman

Set the domain STATE by assigning bandwidths to the links

Select the source and destination for routing

Select the protocol to be used for routing

Generate a request by specifying the bandwidth requirement

IF Shortest Path Routing chosen
{
        Call SPR() to do shortest path routing

        IF SPR is successful
        {
               Display the Shortest Path Generated by SPR

        }
        ELSE
        {
               Display the partial path and the link of failure

        }

}

IF PAP chosen
{
        Call PAP() to do the QoS Routing

        IF PAP successful
        {
               Display the Partial Shortest Path Generated

               Display the QoS Path Generated from branching point

        }

}
```

Figure 4–10: Algorithm for path generation by PAP / SPR

Algorithm.The algorithm to generate and display a path between the selected source and destination is shown as in Figure 4–10

SPR and PAP are successful, if they find a path such that all links on that path have bandwidths assigned $\geq$ required bandwidth.

4.4 Path Generation by Shortest Path Routing–Example 1

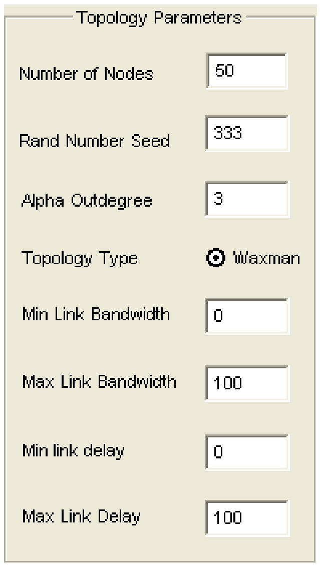
An example path generation by shortest path routing is first shown.

4.4.1 Input Parameters

The input parameters for the sample path generation are as given in the table below.

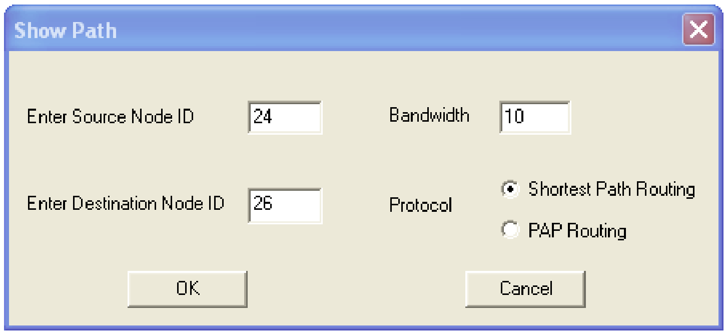
Table 4–3: Parameter Values for Example SPR Simulation

Parameter	Value
Number of Nodes	50
Random Number seed	333
Topology Type	Waxman
Alpha Out degree	3
Minimum Bandwidth	0
Maximum Bandwidth	100
Minimum Delay	0
Maximum Delay	100


Topology Parameters

Number of Nodes	50
Rand Number Seed	333
Alpha Outdegree	3
Topology Type	● Waxman
Min Link Bandwidth	0
Max Link Bandwidth	100
Min link delay	0
Max Link Delay	100

Figure 4–11: The parameter entry window for the example SPR path generation


Show Path

Enter Source Node ID: 24
Bandwidth: 10
Enter Destination Node ID: 26
Protocol: [✓ Shortest Path Routing] PAP Routing

Figure 4–12: Selecting the source and destination for the example SPR simulation

The parameter input window corresponding to the values above is shown in Figure 4–11.

Given these parameters, a topology is generated using the Waxman model for 50 nodes and an average out-degree of 3.

4.4.2 Selecting Source and Destination

Once, the topology has been generated, the source and destination nodes are selected, for the EF stream has to be routed. The EF bandwidth requirement is also specified and the protocol to be used which would be Shortest Path Routing in this case. As an illustration of the path generation, the following table gives the values selected:

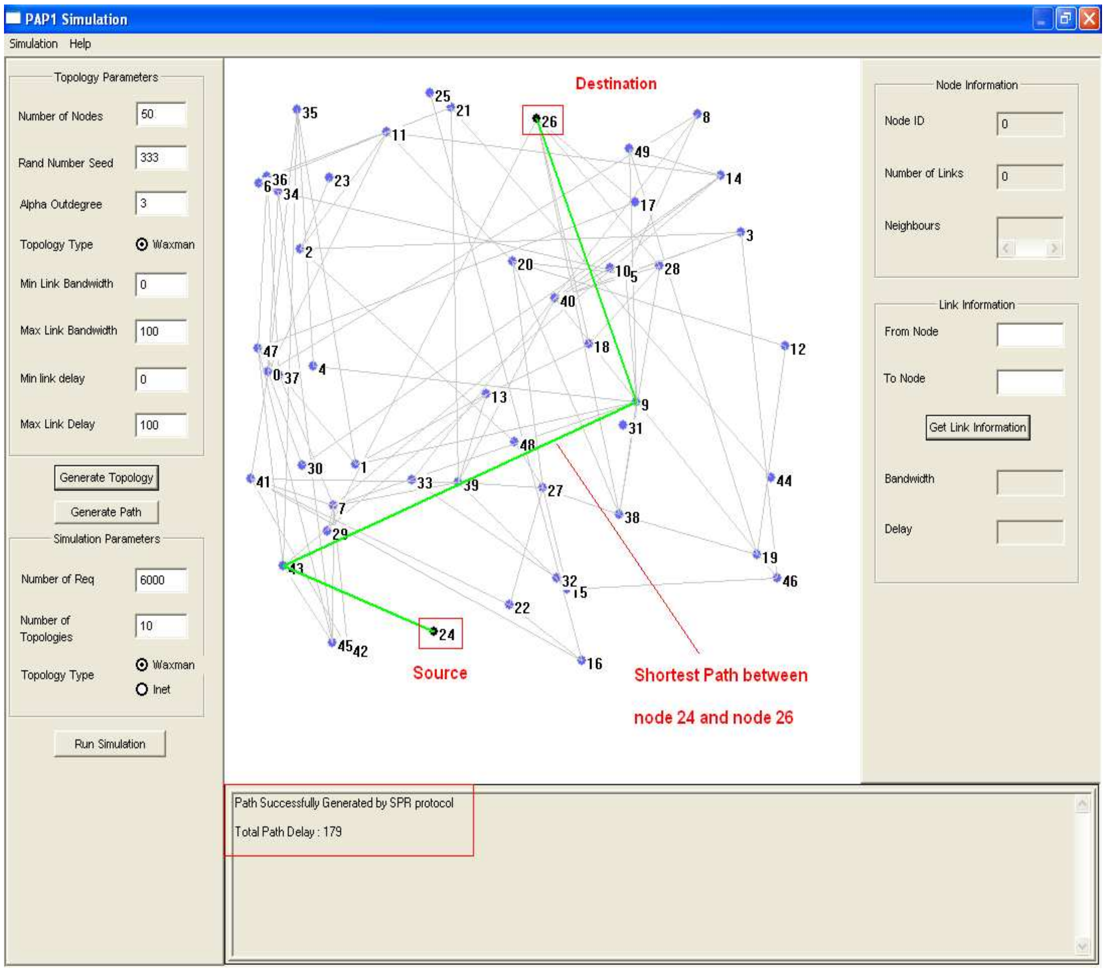
Table 4–4: Parameter Values for Example SPR Simulation Source and Destination

Parameter	Value
Source Node ID	24
Destination Node ID	26
Bandwidth	10
Protocol	SPR

The window corresponding to the selections made is as shown in Figure 4–12

4.4.3 SPR Successful

When the ”OK” button is clicked, the Shortest Path generation algorithm will calculate a shortest path by the Dijkstra’s algorithm from the source to the



Number of Nodes: 50
Rand Number Seed: 333
Alpha Outdegree: 3
Topology Type: [✓ Waxman]
Min Link Bandwidth: 0
Max Link Bandwidth: 100
Min link delay: 0
Max Link Delay: 100

Number of Req: 6000
Number of Topologies: 10
Topology Type: [✓ Waxman] Inet

Node ID: 0
Number of Links: 0
Neighbours:

```
Path Successfully Generated by SPR protocol
Total Path Delay : 179
```

Figure 4–13: SPR protocol successful.The shortest path generated for the specified source and destination is shown in GREEN.The total delay is shown in the text window below

destination. If the acceptance test is passed by all the links along the shortest path, then the protocol is successful. In this case, the topology window will display the calculated shortest path with a green highlighting, showing all the links along the path. For the example simulation input specified above, the shortest path calculated is as shown in Figure 4–13. The source and the destination have been shown in the figure. The green line represents the shortest path between the source and the destination nodes, by which the EF traffic will be routed with guaranteed bandwidth.

This case results when the Shortest Path routing is successful and the EF traffic stream is routed successfully from the source to the destination. If the routing is unsuccessful, the partial path is displayed along with the link of failure. An example will be dealt with in the next section.

4.5 Path Generation by PAP Routing Protocol–Example 1

In the previous section, successful routing using the Shortest Path routing was described. The new routing protocol described will come into picture when the Shortest Path Routing fails and the PAP QoS protocol is triggered. Hence in order to see an example of the new routing protocol, the case in which the shortest path routing fails is first considered.

4.5.1 Input Parameters

Here, a different scenario is considered with the input parameters taken as shown in the table below:

Table 4–5: Parameter Values for Example PAP Simulation–Example 1

Parameter	Value
Number of Nodes	45
Random Number seed	333
Topology Type	Waxman
Alpha Out degree	2
Minimum Bandwidth	0
Maximum Bandwidth	100
Minimum Delay	0
Maximum Delay	100

4.5.2 Selecting the Source and Destination

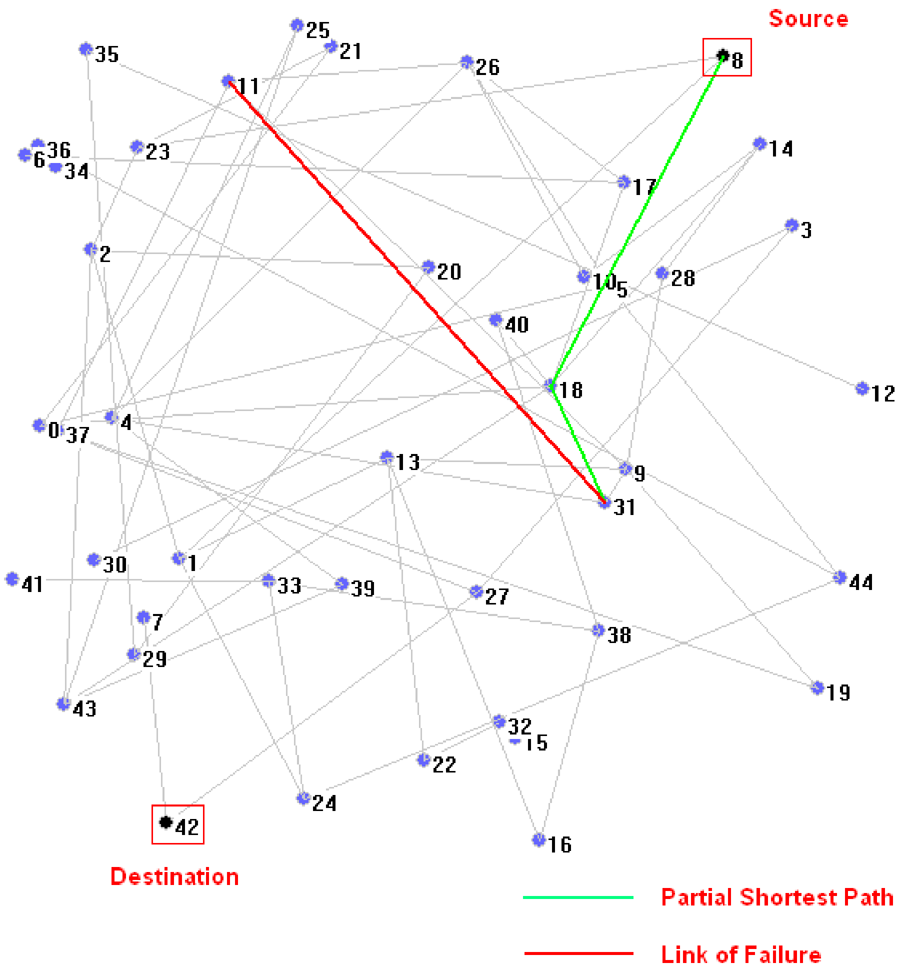
The values for the source, destination and bandwidth requirements for the generated topology are given by the table shown below:

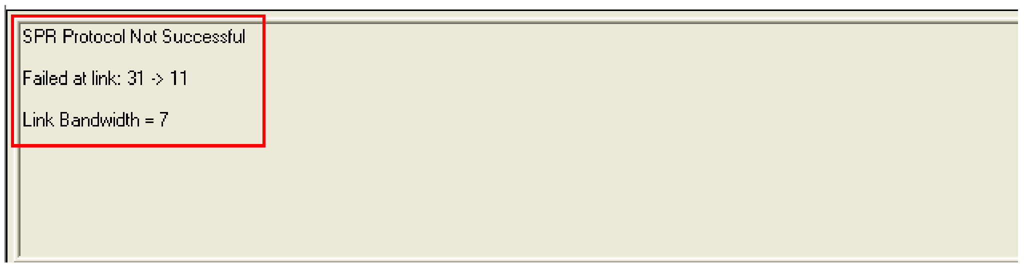
4.5.3 PAP Successful

When the OK button is clicked, the output that is displayed in the topology window is as shown in Figure 4–14.

Table 4–6: Parameter Values for Example PAP Simulation–Example 1

Parameter	Value
Source Node ID	8
Destination Node ID	42
Bandwidth	10
Protocol	SPR

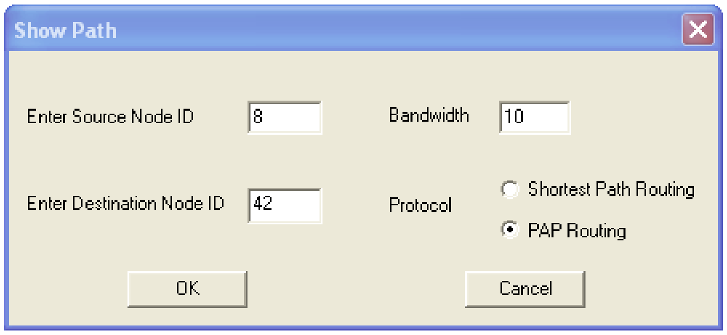



SPR Protocol Not Successful

Failed at link: 31 -> 11

Link Bandwidth = 7

Figure 4–14: Example 1: SPR protocol fails. Green line shows partial shortest path and red shows link of failure



Enter Source Node ID: 8
Bandwidth: 10
Enter Destination Node ID: 42
Protocol: Shortest Path Routing [✓ PAP Routing]

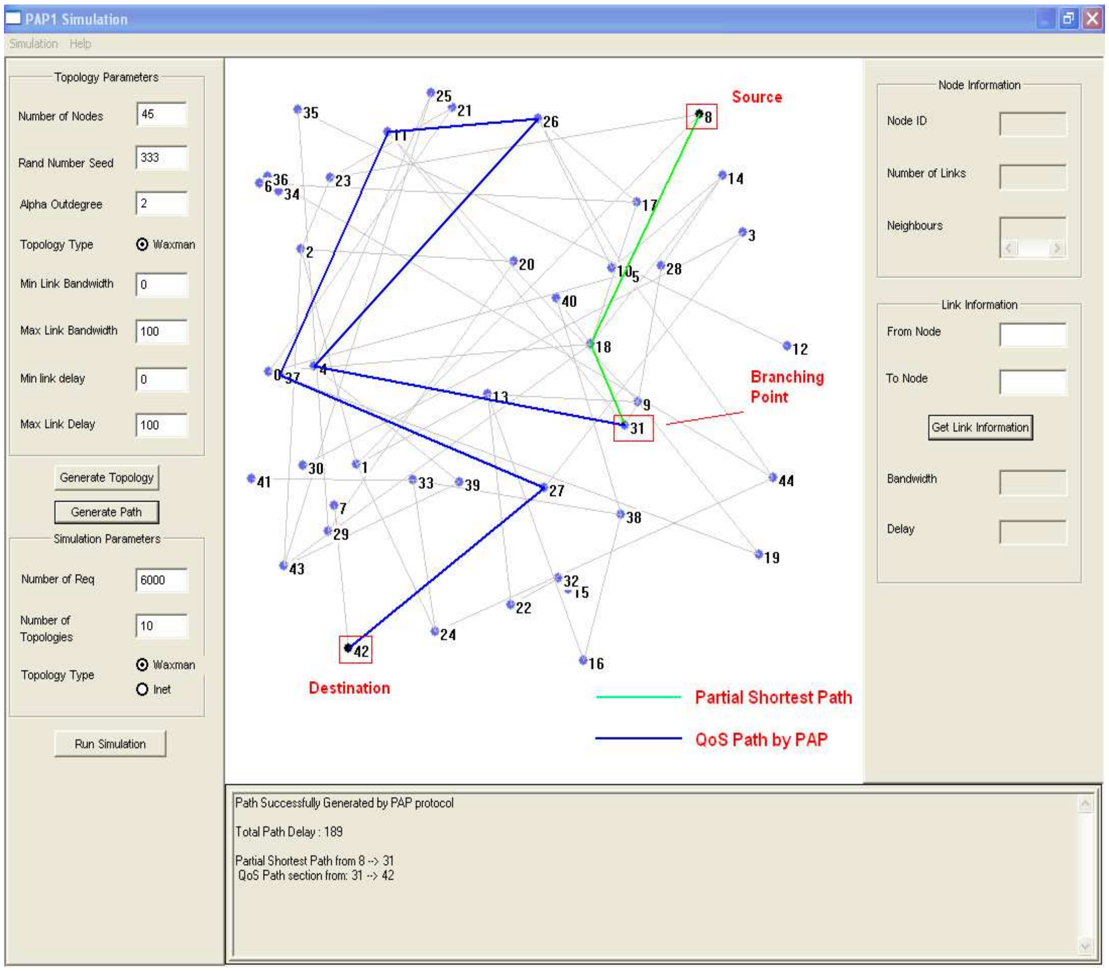
Figure 4–15: Using PAP protocol for the same topology with the same parameters. The EF request has the same bandwidth requirement for the same source and destination

As seen in Figure 4–14, the shortest path routing fails because, it has link 31 $\rightarrow$ 11 on the shortest path which cannot pass the acceptance test. The bandwidth for link 31 $\rightarrow$ 11 is 7 whereas the requested bandwidth for the EF traffic was 10. Hence SPR fails at node 31. Link 31 $\rightarrow$ 11 is the $InfeasibleLink$ as described in chapter 3. Figure 4–14 shows the partial shortest path with *green* color and the infeasible link with *red*. Also seen in the figure is the bandwidth of the link at which the routing failed.

This necessitates the use of a QoS protocol which will guarantee that the EF request will be routed to the destination node 42. The PAP protocol is then triggered which looks for alternate paths at node 31(branching point) which have bandwidth greater than 10 and which can route the EF request. To see how it works, the same scenario is considered with the same values for input parameters. The PAP protocol is now considered for the same source and destination and the same bandwidth requirement. Figure 4–15 show this.

When the OK button is clicked, the PAP protocol gets invoked when SPR fails at node 31 as shown above. The path calculated by PAP protocol is as seen in Figure 4–16.

The figure shows the path as generated by the PAP routing protocol, with the shortest path section and the QoS path section shown with different colors. The



Topology Parameters

Number of Nodes: 45
Rand Number Seed: 333
Alpha Outdegree: 2
Topology Type: [✓ Waxman]
Min Link Bandwidth: 0
Max Link Bandwidth: 100
Min link delay: 0
Max Link Delay: 100

Simulation Parameters

Number of Req: 6000
Number of Topologies: 10
Topology Type: [✓ Waxman] Inet

Node Information

Node ID:
Number of Links:
Neighbours:

Link Information

From Node:
To Node:
Bandwidth:
Delay:

```
Path Successfully Generated by PAP protocol

Total Path Delay : 189

Partial Shortest Path from 8 --> 31
 QoS Path section from: 31 --> 42
```

Figure 4–16: PAP successfully routes the same request for which SPR fails at link 31 → 11. QoS path is seen in BLUE, starting from the branching point (Node 31)

text window below displays the total delay and the different path sections by node numbers.

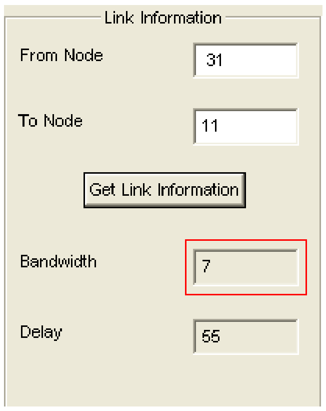
At node 31, SPR chooses path 31 → 11, which fails, since it has a bandwidth (7) less than the required bandwidth. The PAP protocol is triggered, which searches for an alternate path from the branching point node 31. It finds a link 31 → 4 which has a bandwidth (47) greater than the required bandwidth (10). This can be verified by finding the link information for links 31 → 11 and 31 → 4 from the link information window. This is shown in Figures 4–17 and 4–18. Hence the PAP protocol will find the alternate path from node 4 onwards till it reaches the destination node 42. It can be seen that the path shifts back to the remaining shortest path which is from 11 → 42 later. Since the links on the shortest path from 11 onwards can support the request, the PAP protocol follows the shortest path again. After reaching node 42, an accept message will be sent to the source thus forming this new QoS path from node 8 to 42 will be followed for all EF traffic from node 8 to node 42. RT entries are deposited at the intermediate routers accordingly. The branching point in this case is node 31 as shown in Figure 4–16.

There can also be a possibility that the QoS protocol does not find an alternate path at the branching point which would support the EF request by ensuring a bandwidth greater than the requested bandwidth. In such a case, the source node is notified accordingly, and the PAP protocol also fails. A corresponding message is displayed.

The next example shows a case where the SPR fails, PAP is successful, and PAP chooses a path which is totally different from the initial shortest path which SPR would have chosen.

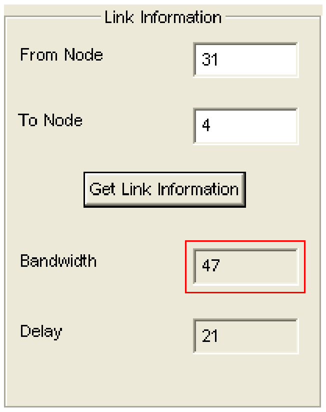
4.6 Path Generation by PAP Routing Protocol–Example 2

Here, a second example is presented to demonstrate the PAP routing protocol. The difference between this example and the previous one is that in the previous



From Node: 31
To Node: 11
Bandwidth: 7
Delay: 55

Figure 4–17: Link information for link 31 → 11 to demonstrate why SPR fails at the branching point.



From Node: 31
To Node: 4
Bandwidth: 47
Delay: 21

Figure 4–18: Link information for link 31 → 4 to demonstrate why this alternate path is chosen by PAP.

case, PAP shifts back to the shortest path after choosing an alternate path over the SPR link of failure. Here in this example, the PAP follows a different path altogether to the destination.

4.6.1 Input Parameters

Here, a different scenario is considered with the input parameters taken as shown in the table below:

Table 4–7: Parameter Values for Example PAP Simulation–Example 2

Parameter	Value
Number of Nodes	45
Random Number seed	333
Topology Type	Waxman
Alpha Out degree	3
Minimum Bandwidth	0
Maximum Bandwidth	100
Minimum Delay	0
Maximum Delay	100

4.6.2 Selecting the Source and Destination

The values for the source, destination and bandwidth requirements for the generated topology are given by the table shown below:

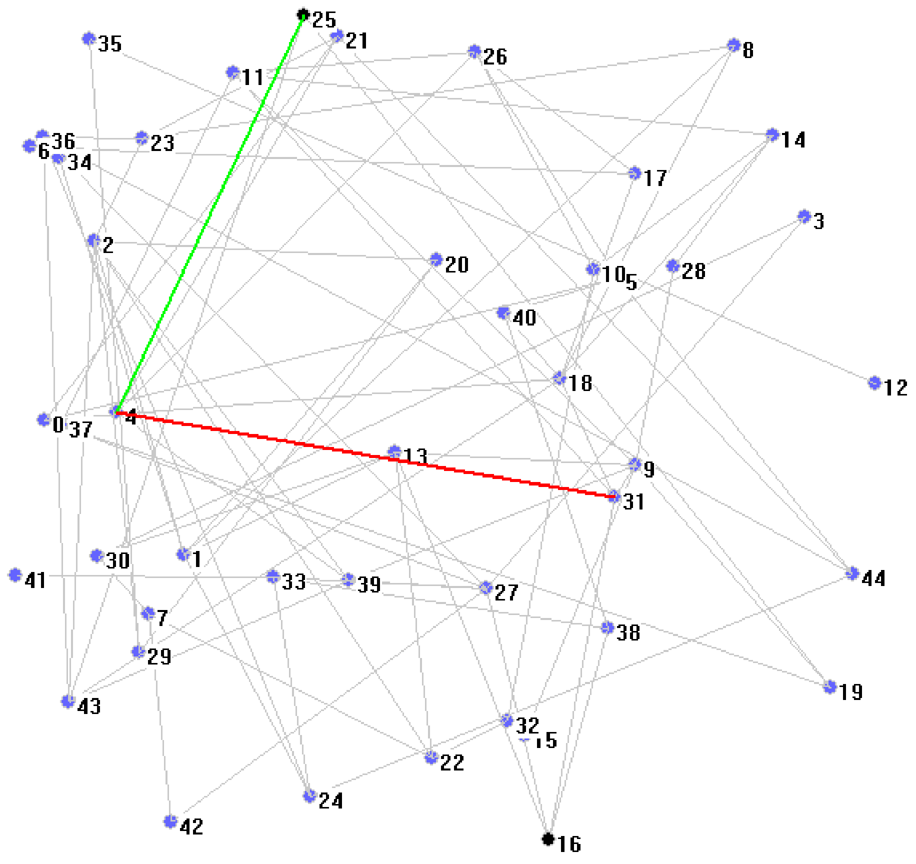
Table 4–8: Parameter Values for Example PAP Simulation–Example 2

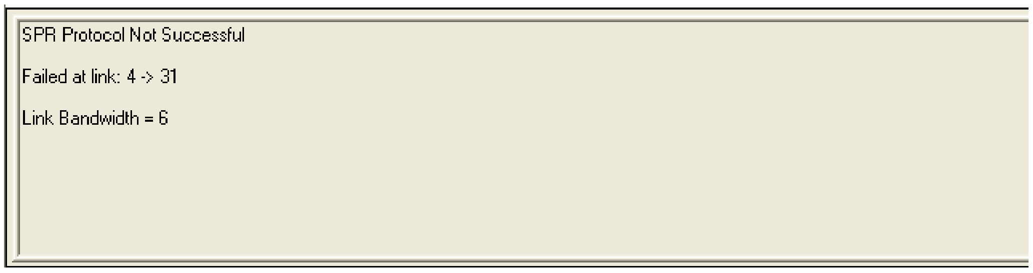
Parameter	Value
Source Node ID	25
Destination Node ID	16
Bandwidth	25
Protocol	SPR

4.6.3 PAP Successful

When the OK button is clicked, the output that is displayed in the topology window is as shown in Figure 4–19.

As seen in Figure 4–19, the shortest path routing fails because, it has link 4 $\rightarrow$ 31 on the shortest path which cannot pass the acceptance test. The bandwidth

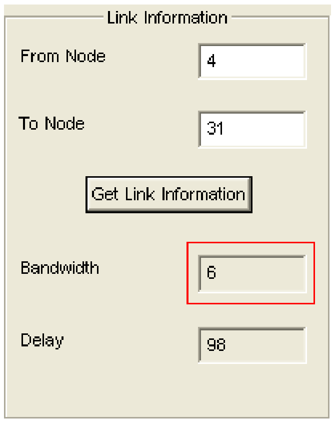



SPR Protocol Not Successful

Failed at link: 4 -> 31

Link Bandwidth = 6

Figure 4–19: Example 2: SPR protocol fails. Green line shows partial shortest path and red shows link of failure


Link Information

From Node 4

To Node 31

Bandwidth 6

Delay 98

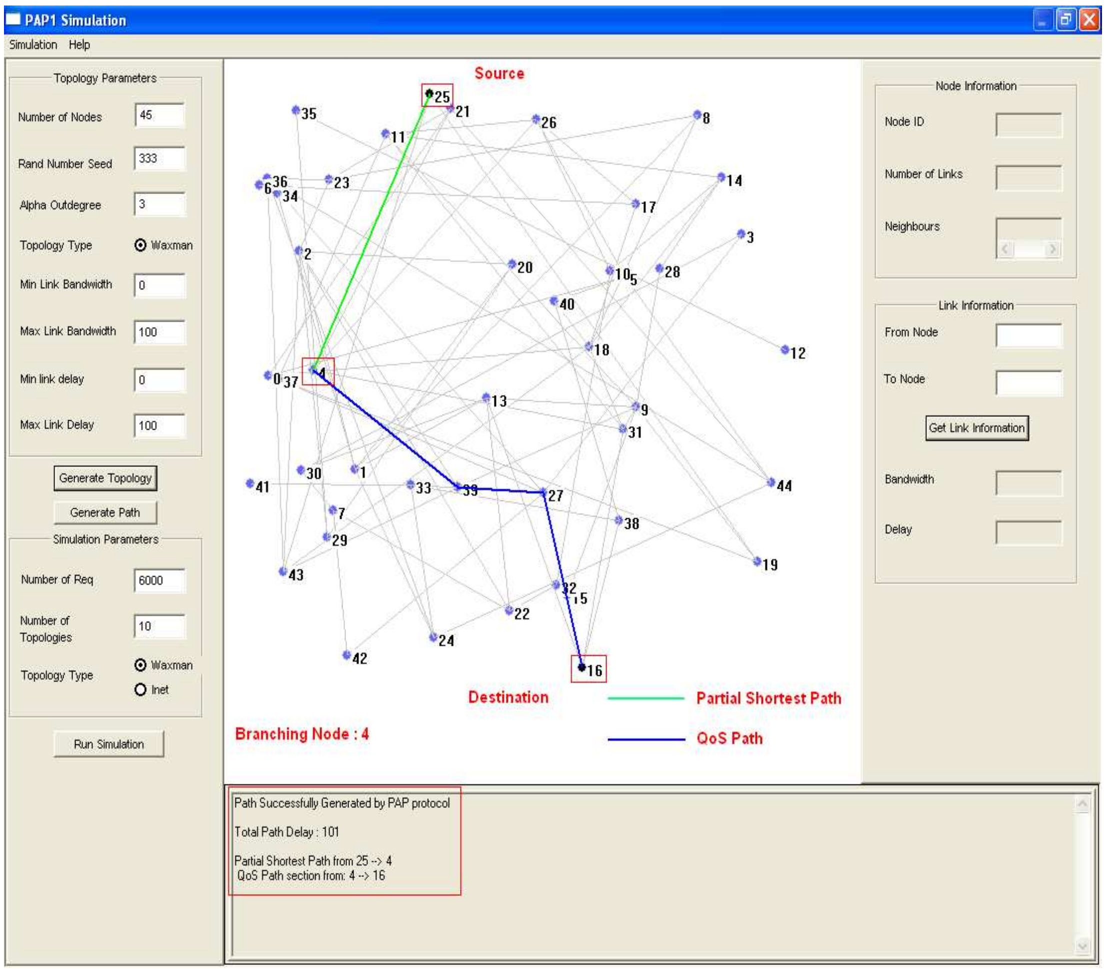
Figure 4–20: Link 4 $\longrightarrow$ 31 has a bandwidth 6.Hence SPR fails at this link

for link 4 $\longrightarrow$ 31 is 6 (Figure 4–20), whereas the requested bandwidth was 25. Hence SPR fails at node 4. Figure 4–19 shows the partial shortest path with *green* color and the link of failure with *red*. Also seen in the figure is the bandwidth of the link at which the routing failed.

The PAP protocol is triggered, which looks for alternate paths at node 4 which is the branching point in this case. For the same source and destination and bandwidth requirement, the PAP protocol generates the path as shown in Figure 4–21.

In this case, however, the QoS path is completely different from the shortest path after the branching point unlike example 1. The path from the branching point is 4 $\longrightarrow$ 39 $\longrightarrow$ 27 $\longrightarrow$ 16; since link 4 $\longrightarrow$ 39 has a link bandwidth of 79 as shown in Figure 4–22. The initial link chosen by shortest path routing at which it fails (4 $\longrightarrow$31) has a bandwidth of 6 as shown in Figure 4–20

The above two examples demonstrate the use of the new routing protocol especially for the EF traffic in case of Differentiated Services, which requires guaranteed delivery. In such cases, the new protocol will try to find out alternate paths in order to guarantee the flow of EF request even if shortest path routing fails at a certain point.



Topology Parameters

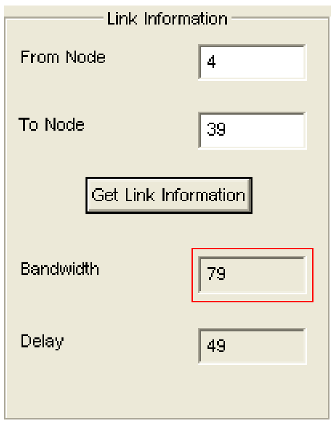
Number of Nodes	45
Rand Number Seed	333
Alpha Outdegree	3
Topology Type	[✓ Waxman]
Min Link Bandwidth	0
Max Link Bandwidth	100
Min link delay	0
Max Link Delay	100

Simulation Parameters

Number of Req	6000
Number of Topologies	10
Topology Type	[✓ Waxman] Inet

```
Path Successfully Generated by PAP protocol
Total Path Delay : 101
Partial Shortest Path from 25 --> 4
 QoS Path section from: 4 --> 16
```

Figure 4–21: PAP successfully routes the same request for which SPR fails at link 4 → 31. QoS path is seen in blue, starting from the branching point (Node 4)



From Node	4
To Node	39
Bandwidth	79
Delay	49

Figure 4–22: Link 4 → 39 has a bandwidth 79. Hence PAP chooses it for the alternate path.

4.7 Multiple Request Multiple Topology Simulation

The first part of the simulation discussed in the previous section generated a single topology and then compared the Shortest Path Routing and the new PAP protocol by generating a single request with a specified bandwidth requirement. The source and destination were chosen by the user and then the paths as generated by the two protocols were displayed graphically on the topology generated.

In the second part of the simulation, multiple topologies are generated. For each topology, a random source and destination node are selected from the topology and several thousands of requests are generated to be routed from the selected source to destination. Both the protocols are run for each topology and each request and then the comparison results are displayed in the results window below. The two protocols are compared on the basis of $LinkSuccessRatio$ which is the percentage of requests that are admitted. The results of comparison will be considered in the next chapter.

4.7.1 Simulation Steps

This section describes step wise how the multiple topology, multiple request simulation has been designed.

Input. The simulation takes as input the same parameters as the simulation in the previous section, with the following additional parameters:

- Number of Topologies
- Number of Requests
- Topology Type (Inet/Waxman)

Algorithm:.

The simulation follows the algorithm given in Figure 4–23

The requests that are generated for each topology are not based on the bandwidth requirement, but on the link success probability for each link in the topology. For each value of link success probability ranging between 0.1 and

```
Take Input Parameters

For every topology DO
{
      Generate Topology

      Generate the specified number of requests

      For link success probability from 0.1 to 1.0 DO
      {
            For each request DO
            {

                  Select a random Source and destination

                  Set the link probabilities for all links

                  Call SPR() to do shortest path routing

                  If SPR is successful
                  {
                         Collect results for this request
                         (path and other statistics)
                  }

                  Call PAP to apply the PAP routing protocol

                  If PAP is successful
                  {
                         Collect results for this request
                         (path and other statistics)
                  }

             }

      }

}

For link success probability from 0.1 to 1 DO
{
      Compute the link success ratio for both protocols

      Compute the average RT entries for PAP
}

Display the results
```

Figure 4–23: Algorithm for the multiple topology, multiple request simulation

1.0(in increments of 0.1), each link is assigned a link success probability which decides whether the link will pass the acceptance test or not.[3] This domain state is calculated each time for each request generated for each topology.Thus for each different request generated, the link success probabilities will be different and hence for each of the 10 iterations of the simulation run for link probability values from 0.1 to 1.0,

the average success ratio is calculated by the formula

$$\text{successRatio} = \frac{numSuccess}{numRequests}$$

where

numSuccess = number of successful routing attempts

numRequests = number of requests generated

The average number of RT entries deposited is calculated by the formula

$$\text{avgRTentries} = \frac{numRTentries}{numSuccess}$$

where

numSuccess = number of successful routing attempts

numRTentries = number of routing entries deposited for the successful routing attempt

The final values for these two parameters for values of link success probabilities ranging from from 0.1 to 1.0, are then displayed. These values form the basis for comparison for the two protocols.

4.7.2 Multiple Topology, Multiple Request Simulation–Example

This section gives an example of a Multiple Topology, Multiple Request Simulation.

[3] The bandwidth in this case is 0 for all links

Input Parameters.

The following table gives the input parameters for the example simulation:

Table 4–9: Parameter Values for Multiple Topology Multiple Request Simulation–Example

Parameter	Value
Number of Nodes	50
Random Number seed	333
Topology Type	Waxman
Alpha Out degree	3
Minimum Bandwidth	0
Maximum Bandwidth	100
Minimum Delay	0
Maximum Delay	100
Number of Topologies	10
Number of Requests	6000

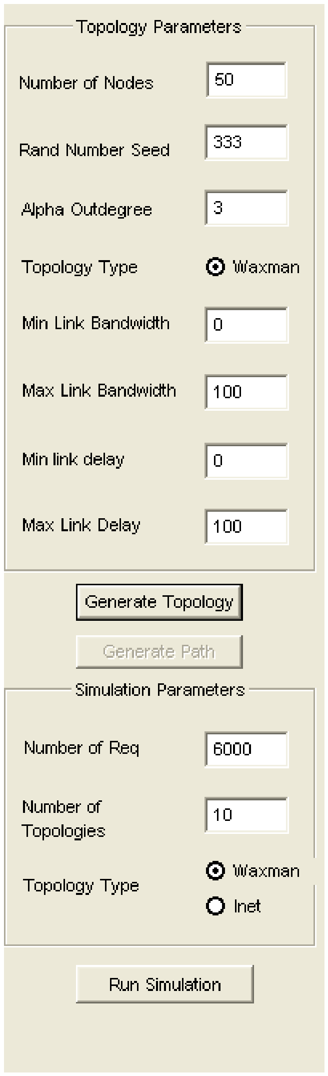
This is shown in Figure 4–24.

Running the simulation. When the "Run Simulation" button is clicked, the simulation is triggered, which then follows the algorithm given in the previous section. After the requests are generated and processed for all topologies, the results are displayed in the text window as shown in figure 4–25.

As seen in the Figure 4–25, the two protocols are compared on the basis of link success ratio with values for link success probability ranging from 0.1 to 1.0. This example shows the simulation run with the Waxman topology generation method. The simulation can also be run with the Inet topology generation method.

The results for the simulation with same parameter values but run with Inet topology generation method are as shown in Figure 4–26.

This chapter described the simulation that has been implemented to demonstrate the new routing architecture and the use of the new PAP routing protocol. It also presented its advantages over the traditional Shortest Path Routing. It was seen that for traffic like EF traffic in Differentiated Services domains, guaranteed



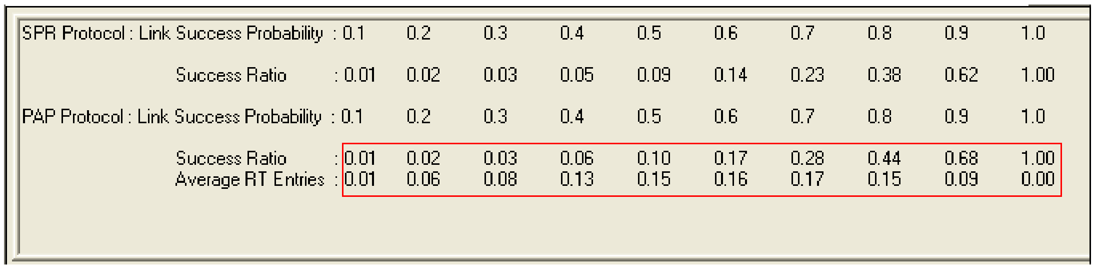
Topology Parameters

Number of Nodes	50
Rand Number Seed	333
Alpha Outdegree	3
Topology Type	[✓ Waxman]
Min Link Bandwidth	0
Max Link Bandwidth	100
Min link delay	0
Max Link Delay	100

Simulation Parameters

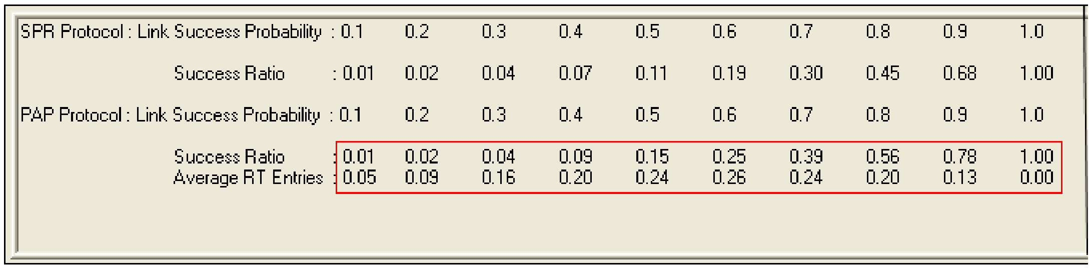
Number of Req	6000
Number of Topologies	10
Topology Type	[✓ Waxman] Inet

Figure 4–24: The input parameters for the multiple topology, multiple request simulation example



SPR Protocol : Link Success Probability	: 0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Success Ratio	: 0.01	0.02	0.03	0.05	0.09	0.14	0.23	0.38	0.62	1.00
PAP Protocol : Link Success Probability	: 0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Success Ratio	: 0.01	0.02	0.03	0.06	0.10	0.17	0.28	0.44	0.68	1.00
Average RT Entries	: 0.01	0.06	0.08	0.13	0.15	0.16	0.17	0.15	0.09	0.00

Figure 4–25: The results of the multiple topology, multiple request simulation example using Waxman. The protocols are compared on the basis of link success ratio and average number of RT entries.



SPR Protocol : Link Success Probability	: 0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Success Ratio	: 0.01	0.02	0.04	0.07	0.11	0.19	0.30	0.45	0.68	1.00
PAP Protocol : Link Success Probability	: 0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Success Ratio	: 0.01	0.02	0.04	0.09	0.15	0.25	0.39	0.56	0.78	1.00
Average RT Entries	: 0.05	0.09	0.16	0.20	0.24	0.26	0.24	0.20	0.13	0.00

Figure 4–26: The results of the multiple topology, multiple request simulation example using INET

delivery can be assured with the new protocol, by searching for alternate paths in case the shortest path routing fails.

4.8 Summary

The chapter describes the simulation and the GUI that has been implemented to demonstrate the new routing architecture. The $SingleTopology, SingleRequest$ simulation displays the paths as generated by the shortest routing protocol and the PAP protocol for a selected source and destination for a requested EF traffic stream. The $MultipleTopology, MultipleRequest$ simulation gives statistical results to compare the PAP protocol with the shortest path routing by generating several thousand EF stream requests on a number of topologies. The next chapter will discuss the results of comparison between the two protocols.

CHAPTER 5
CONCLUSIONS AND FUTURE SCOPE

The previous chapter described the simulation which demonstrated the use of the new routing architecture and a sample protocol(PAP) that can be used with the architecture to guarantee timely delivery of EF traffic in Differentiated Services domains.

5.1 Comparison

The last section in the previous chapter described the comparison of the PAP protocol with the traditional Shortest Path Routing. Several topologies were generated with several requests routed on randomly selected source and destination nodes. The protocols were compared on the following two parameters:

1. Average Link Success Ratio
2. Average Number of RT Entries

The link success probability values were varied from 0.1 to 1.0 with increments of 0.1 and for each value, the above mentioned two parameters were calculated. Figure 4–25 and Figure 4–26 shows the values for these parameters for Waxman and Inet model respectively. The average result of the 6000 requests gives a data point.

5.1.1 Link Success Ratio

The values for the link success ratio have been given in the following table for the example simulation given in the previous chapter

From the values in the table, we observe that the link success ration for the proposed QoS routing algorithm (PAP) is higher than the link success ratio for Shortest Path Routing for link success probability values over 0.3, thus showing

Table 5–1: Link Success Ratio Values for PAP and SPR–Using Waxman

Link Success Probability	SPR	PAP
0.1	0.01	0.01
0.2	0.02	0.02
0.3	0.03	0.03
0.4	0.05	0.06
0.5	0.09	0.10
0.6	0.14	0.17
0.7	0.23	0.28
0.8	0.38	0.44
0.9	0.62	0.68
1.0	1.00	1.00

Table 5–2: Link Success Ratio Values for PAP and SPR–Using Inet

Link Success Probability	SPR	PAP
0.1	0.01	0.01
0.2	0.02	0.02
0.3	0.04	0.04
0.4	0.07	0.07
0.5	0.11	0.15
0.6	0.19	0.25
0.7	0.30	0.39
0.8	0.45	0.56
0.9	0.68	0.78
1.0	1.00	1.00

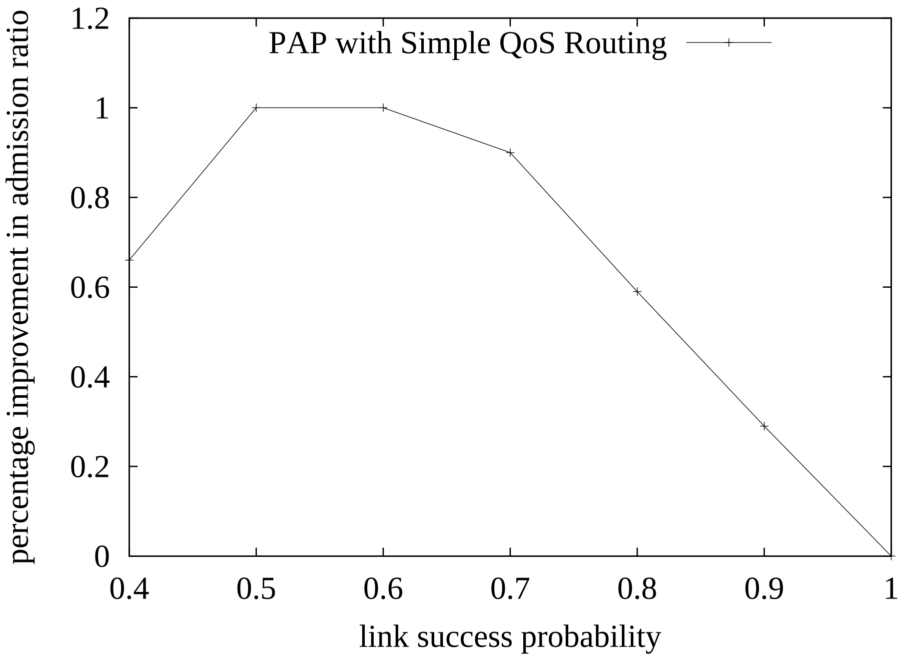


Figure 5–1: Admission ratios per admitted flow

that the new protocol does improve the number of requests that are admitted successfully. Comparing with traditional routing, PAP thus improves the admission ratio (percentage of requests that are admitted) significantly, up to 100%. Figure 5–1 shows this.

5.1.2 Average Number of RT Entries

The new protocol achieves this improvement in admission ratio at a small cost. This can be seen from the values of the Average Number of RT entries deposited for the PAP protocol. The table below gives the value for Average RT Entries for the example simulation described in the previous chapter.

From the values in the tables, it can be seen that PAP incurs a very small cost with respect to the number of RT entries deposited. In the worst case, it deposits 0.51 QRT entry per admitted flow on average. The reason for the less-than-one average is that, for requests that can be supported by the primary paths, PAP is equivalent to traditional routing (TRT only), and no QRT entry is required.

Table 5–3: Average RT Entries Values for PAP–Using Waxman

Link Success Probability	Average RT Entries
0.1	0.01
0.2	0.06
0.3	0.08
0.4	0.13
0.5	0.15
0.6	0.16
0.7	0.17
0.8	0.15
0.9	0.09
1.0	0.00

Table 5–4: Average RT Entries Values for PAP–Using Inet

Link Success Probability	Average RT Entries
0.1	0.05
0.2	0.09
0.3	0.16
0.4	0.20
0.5	0.24
0.6	0.26
0.7	0.24
0.8	0.20
0.9	0.13
1.0	0.00

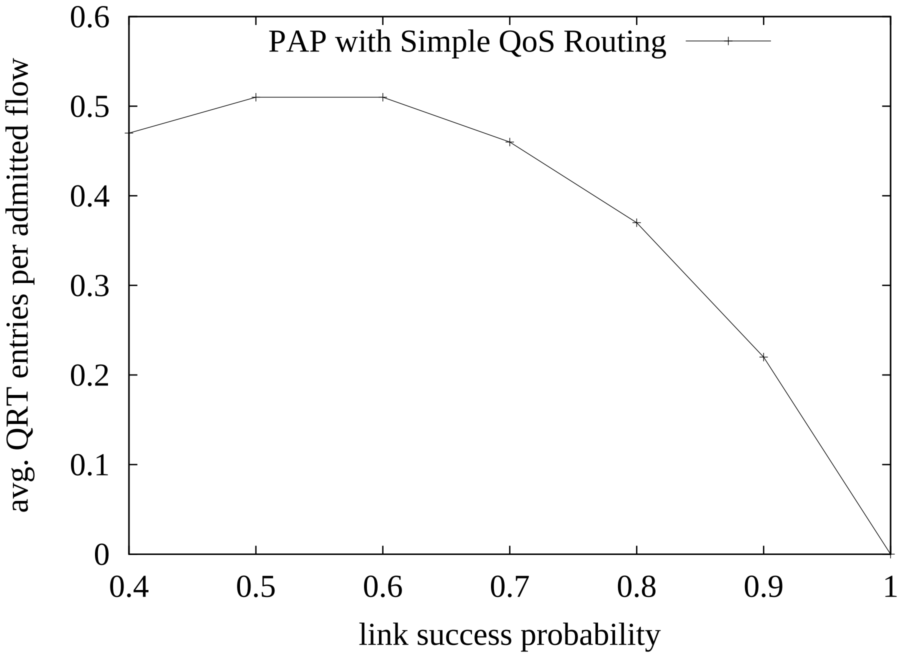


Figure 5–2: QRT entries for PAP per admitted flow

This improvement can be seen in Figure 5–2.

5.2 Conclusion

In the paper by Dr. Chen *et al.*[2], a new routing architecture for DiffServ domains has been proposed. It integrates the traditional routing, QoS routing, packet forwarding, and admission control. The admission control and QoS routing ensure that the EF traffic admitted to the network is limited under the maximum allowed quota so that the EF traffic always receives assured bandwidth and low delay. Much care has been taken to reduce the size of the QoS routing tables. In particular, the proposed QoS routing protocol requires only one table entry at a branching point to store an alternative routing path.

The simulation and the GUI that has been implemented helps demonstrating the advantages of the new protocol over the traditional protocols like the shortest path routing. It displays graphically how the protocol searches for alternate paths when the shortest path routing fails at a certain node in the network. With the help of the GUI, this can be verified by looking at the link bandwidths for the links of failure or the alternate links chosen by the new protocol. Comparing these two protocols on the basis of the link success ratio gives a view of how the new protocol improves admission control at a very low cost. The GUI gives a graphical view of the paths generated thus giving an idea of the QoS routing in the internet world, providing more opportunities for research in this area.

5.3 Future Scope

This thesis discusses a new protocol and its implementation for the DiffServ domains but it deals with the Expedite Forwarding Traffic PHB only. In chapter 2, two classes of PHBs were seen for the Differentiated Services domains:

1. Expedite Forwarding (EF)
2. Assured Forwarding (AF)

The protocol proposed can be extended or modified to handle AF traffic also. AF traffic defines different forwarding classes each with three different drop precendences. This is mainly to provide different levels of services to customers and applications. Considering this, the protocol and/or the implementation can be modified / extended to handle AF traffic class for each of the forwarding classes that it defines.

Also, even for the EF traffic class, the protocol can be compared with several other traditional routing protocols apart from the Shortest Path Routing and this can be demonstrated by enhancing and extending the implementation / GUI. This could give a clear view of whether the new routing architecture does provide a considerable advantage over most of the currently used routing protocols.

In this implementation, the PAP routing protocol uses the basic approach of finding a shortest path from the source to the destination(shortest path algorithm) even after finding an alternate path. It tries to shift back to the original shortest path to see if it can now support the requested bandwidth. If being compared with other routing protocols, the method of finding the route to the destination after the alternate path has been found from the branching node, could be made more efficient than using the shortest path only. The approach could be recursively applied at each hop along the path, which would be completely independent of the initial shortest path, thus suggesting the use of multiple branching points instead of one.

The simulation uses only the Waxman and Inet topology generation methods. Waxman has been used for the simplicity of displaying the topology since Waxman provides well defined co-ordinates for the nodes that could be plotted. But there are many other network topology generators which could be used in order to simulate a more realistic network topology to model the DiffServ domain. Some of the popular network topology generators that could be used are:

1. The NS-2 simulator from Stanford University[4],
2. The GT-ITM topology generator from Georgia Institute of Technology[13],
3. The BRITE topology generator from Boston University[7],

Such enhancements could open new ways for further research in the proposed protocol and its implementation.

REFERENCES

[1] Berghoff, G., "Introduction to QoS and Differentiated Services," *Application to Nokia's IP RAN*, May 2002, Available at URL:
www-i4.informatik.rwth-aachen.de/Kolleg/Vortraege/Berghoff.pdf
Last Accessed: March 2004

[2] Chen, S., Ling, Y. and Chen, S., "A New Routing Architecture for DiffServ Domains" *International Conference on Communications, Internet and Information Technology (CIIT 2003)*, Scottsdale, AZ, Nov 2003

[3] Cheng, J., Chen, Q. and Jamin, S., "Inet: Intenet Toplogy Generator," *Technical Report* UM-CSE-TR-433 – 00, Available at URL:
http://topology.eecs.umich.edu/inet/inet-2.0.pdf
Last Accessed: March 2004

[4] Greis, M., "Tutorial for the Network Simulator "ns"," *Information Sciences Institute*, Available at URL:
http://www.isi.edu/nsnam/ns/tutorial/index.html
Last Accessed: March 2004

[5] Kasari, A., "An Architecture for Differentiated Services," *Research Seminar on IP Quality Of Service*, Department of Computer Science,University of Helsinki, Oct. 2000, Available at URL:
http://www.cs.helsinki.fi/u/kraatika/Courses/QoS00a/kasari.pdf
Last Accessed: March 2004

[6] Manner, J., "Short Overview of Differentiated Services," Oct 2003, Available at URL:
www.cs.helsinki.fi/u/jmanner/Courses/seminar_papers/diffserv.pdf
Last Accessed: April 2004

[7] Medina, A., Lakhima, A., Matta, I. and Byers, J., "BRITE: Universal Topology Generation from a User's Perspective," *Computer Science Department, Boston University*, Apr 2001, Available at URL:
http://www.cs.bu.edu/brite/user_manual/BritePaper.html
Last Accessed: March 2004

[8] Microsoft Corporation, "Microsoft Windows 2000 Server: A Short Overview of QoS Mechanisms and their Interoperation" *White Paper*, Nov 1999, Available at URL:
http://www.microsoft.com/windows2000/docs/QoSMech.doc

Last Accessed: March 2004

[9] Nortel Networks, "Introduction to Quality of Service(QoS)," *White Paper on Quality of Service*, 2003, Available at URL:
www.nortelnetworks.com/products/02/bstk/switches/bps/collateral/56058.25_022403.pdf
Last Accessed: March 2004

[10] Schildt, H., *MFC programming from the ground up*, 2nd Edition, Osborne/McGraw-Hill, New York, 1998.

[11] Tangmunarunkit, H., Govindan, R., Jamin, S., Shenker, S. and Willinger, W., "Network Topologies,Power Laws and Heirarchy," *Technical Report* USC-CS-01 – 746, Available at URL:
http://www.isi.edu/~hongsuda/publication/ccrNote01_2.ps
Last Accessed: March 2004

[12] Tangmunarunkit, H., Govindan, R., Jamin, S., Shenker, S. and Willinger, W., "Network Topology Generators: Degree Based vs. Structural," in *Proceedings of ACM SIGCOMM'02*, Pittsburgh, PA, Aug 2002. Available at URL:
http://www.acm.org/sigcomm/sigcomm2002/papers/topogen.pdf
Last Accessed: March 2004

[13] Thomas, M., Edwards, E. and Bhattacharjee, S., "Modeling Topology of Large Internetworks," *College of Computing, Georgia Institute of Technology*, May 1997, Available at URL:
http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html
Last Accessed: March 2004

[14] Wilkins, S., Garg, S. and Meyyammai, S., *MFC Development using Microsoft Visual C++*, Microsoft Press, Redmond, WA, 2000.

[15] Zhang, P., Kantola, R. and Aalto, S., "QoS Routing for DiffServ Networks: Issues and Solutions (Draft)," *Technical Report, Networking Laboratory, Helsinki University of Technology*, Oct 2002, Available at URL:
www.netlab.hut.fi/tutkimus/ironet/papers/report-qosr-diffserv.ps
Last Accessed: March 2004

BIOGRAPHICAL SKETCH

Hrishikesh Nulkar was born on April 27^{th}, 1981, in the city of Pune, Maharashtra , India. He received his bachelor's degree, Bachelor of Engineering, in Computer Science and Engineering from the Pune University, India in June 2002.

In August 2002, he joined the University of Florida to pursue a master's degree in computer science and engineering. During his master's study he has been a Teaching Assistant at the Department of Computer and Information Science and Engineering at the University of Florida. His research interests include computer networks, network security. He expects to complete his degree in May 2004.