QoS Aware Path Protection Schemes for MPLS Networks

B.Tech. Project Thesis

submitted towards the partial fulfillment

of requirements for the degree

of

Bachelor of Technology

in

Computer Science & Engineering

by

Ashish Gupta
98130

Ashish Gupta
98131

Under the Guidance of

Prof. B. N. Jain

Department of Computer Science and Engineering

Indian Institute Of Technology, Delhi

May 2002

Acknowledgment

We heartily thank our project supervisor *Dr. B. N. Jain* for providing us with invaluable help and guidance throughout the course of this project. He introduced us to the area and provided us with the required understanding and support. He constantly inspired us to work in new directions and without his valuable guidance, it would be impossible for us to successfully engage in this project.

We would also like to thank Dr. Naveen Garg and Dr. Sanjiva Prasad for their insightful comments regarding our work which helped us to improve our work significantly.

We would like to thank the faculty, staff and students of Computer Science Department for providing a rich, stimulating and above all fun environment in which to work and live.

We are also grateful to *Mr. S. Negi*, STA, incharge of Advanced Networking Lab for helping us out will all the technical difficulties in the lab.

Ashish Gupta(98130) Ashish Gupta(98131)

Certificate

This is to certify that the dissertation entitled **QoS Aware Path Protection Schemes for MPLS Networks** submitted by Ashish Gupta (98130) and Ashish Gupta (98131) in partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering to Indian Institute of Technology, Delhi, is a bonafide record of work carried out by them under my supervision and guidance.

This work has not been submitted elsewhere for the award of any degree or diploma.

Prof. B. N. Jain
Department of Computer Science & Engineering
Indian Institute of Technology, Delhi

Abstract

Multiprotocol label switching (MPLS) integrates the label swapping Forwarding paradigm with network layer routing to allow flexibility in the delivery of new routing services. To deliver reliable services, MPLS requires a set of procedures to provide protection of the traffic carried on the label switched paths. We discuss a new scheme for providing path protection using a segment-based approach. Existing schemes like Local Path Protection protect each node/link by providing a backup path for each. In the new approach we look at the entire path as a group of segments (each consisting of successive links) and provide protection for each segment separately. This scheme offers flexibility in providing path protection to the network operator in comparison to the existing schemes. It provides a protection configuration meeting given QoS constraints and provides conservation of resources at the same time by using lesser number of backup paths. We discuss various mechanisms for providing Path Protection like detection and notification in the context of Segment Based Approach. Then we present algorithms along with analysis for segmenting a primary path which assure satisfaction of various QOS constraints in case of failure like switch over time, end-to-end delay, jitter and reliability with the aim of using no more resources than necessary. A visualization system for the above algorithms has also been developed which aids in understanding the above algorithms. The simulation results show the advantages of the segment based algorithms over standard solutions like Local Path Protection. This advantages include conservation of protection resources while guaranteeing satisfaction of QoS constraints.

Contents

List of Figures

Chapter 1

Introduction

Network routing deployed in the Internet today is focused primarily on connectivity, and typically supports only one class of service, the best effort class. Multi-protocol label switching [1], on the other hand, by integrating a label-swapping framework with network layer routing allows flexibility in the delivery of new routing services, since it allows new services to be added without changing the basic forwarding mechanism. This enables more sophisticated features such as quality-of-service (QoS) and traffic engineering to be implemented. An important component of providing QoS, however, is the ability to do so reliably and efficiently. Although the current routing algorithms are very robust and survivable, the amount of time they take to recover from a failure can be significant, on the order of several seconds or minutes, causing serious disruption of service in the interim. This is unacceptable to many organizations that aim to provide a highly reliable service, and thus require recovery times on the order of tens of milliseconds.

Any breakdown of a network component must not affect the traffic streams drastically enough so as to effect the service requirements for that affected traffic. Hence, a robust scheme for fault - tolerance must be present to deal efficiently with failure scenarios.

1.1 Path Protection

The method of Path Protection provides an effective way to provide fault tolerance in the network to deal with router or link failures inside the network. Its importance and applicability is on the rise especially with the newer innovations like MPLS networks. The reason is that contributes significantly towards providing network reliability by enabling a faster recovery from failures than is possible with Layer 3 mechanisms alone. In IP networks for example, in case of a failure in the network , the source router on getting the failure notification must find out alternate paths to the destination using existing dynamic routing algorithms. After figuring out an alternate path, it can then send the packets over the new route. However the time consumed

[1]For an introduction to MPLS Networks, please refer to the appendix

by this procedure is usually of the order of seconds, which may be quite unacceptable for the newer network applications which involve transmission of multimedia or other real-time traffic over the Network. Thus, to deal with such scenarios, the scheme of path protection was introduced. In Path protection, an alternate backup path is pre-reserved along with the primary path during the setting up of the primary path itself. The packets initially are sent on the primary path. In case of a failure on the primary path, the source on getting the failure notification reroutes the traffic over the backup path, thus coping with the failure. This provides significant improvement over the Layer 3 mechanisms, in which a major time is spent on finding an alternate path to carry the traffic. In Path protection, this is not necessary as the backup path is persevered in the network.

An important point in this regard is that Path Protection is applicable to networks, which allow a priori reservation of paths. MPLS networks provide the possibility for traffic engineering and pre-reservation of paths, thus allowing Path Protection schemes to be easily implementable over MPLS networks.

Since MPLS binds packets to a route (or path) via the labels, and is likely to be the technology of choice in the future IP-based transport network, it is imperative that MPLS be able to provide protection and restoration of traffic. In fact, a protection priority could be used as a differentiating mechanism for premium services that require high reliability.

Some Advantages of Path Protection

Protection of traffic at the MPLS layer (called MPLS protection) is useful for a number of reasons.

- The most important is its ability to increase network reliability by enabling a faster response to failures than is possible with Layer 3 (or the IP layer) alone.
- Protection at the MPLS layer gives the provider the flexibility to choose the granularity at which traffic is protected, and to also choose the specific types of traffic that are protected.

1.2 Existing ideas in Path Protection

Since the major goal of path protection can be stated to be to provide quick recovery after a network failure (of the order of milliseconds), different schemes have been proposed for this, which differ in their capability and features for providing recovery from failure. Two major existing schemes are local path protection and global path protection.

In global path protection (see fig GPP), there is only one backup path starting from the source router to the destination router (egress router). So in case of a failure, the source router is responsible for switching over to the alternate backup path.

In local path protection (see fig LPP), the idea is to provide protection for each link or node separately, thus resulting in reservation of many backup paths. The difference in these two approaches is in terms of speed of recovery after failure and the amount of bandwidth consumed by the backup paths. Local path protection provides quick recovery, as in case of a failure only the upstream router needs to take the action of switching over the traffic to the backup path. But is has the disadvantage of much greater bandwidth requirements.

So, these two schemes lie at the two extreme ends of the spectrum and provide fixed solutions to the problem of path protection. However, a whole range of solutions is possible between these two extreme ends, which provide a lot of flexibility for providing path protection, not just fixed ones. The goal is to meet various QOS constraints like bounded switch over time after failure, reliability and so on after a failure on the primary path, while also conserving bandwidth required for the reservation of the backup paths (which may be more than one). In this paper, we look at a significantly general approach towards providing protection, which provides lots of flexibility in meeting the protection needs of the network for various traffic streams based on their service agreements.

1.3 The Segment based approach

The main advantage of this approach is that it is configurable according to the parameters (such as QOS constraints and bandwidth constraints) provided by the administrator and thus provides the flexibility to the administrator to choose between the trade-offs and select the most appropriate protection scheme, instead of having to rely upon just two fixed schemes for providing this protection.

The way this flexibility is introduced in path protection is to look at the primary path not just as a sequence of links but also as a sequence of segments, with each segments itself consisting of a sequence of links. As one can see in the fig (SBPP), the entire primary path can thus be broken down into variable sized segments. Now the key idea in our new approach is providing protection for each segment as a whole, instead of providing protection for the whole path or each link separately (corresponding to Global Path protection and Local Path Protection respectively). This will result in fewer backup paths, and may still be able to meet the given QOS constraints as required for a particular traffic stream, thus resulting in a better protection design than the extreme case of local path protection. Note that since the size of each segment need not be fixed and can vary, this allows us to decide the number of segments constituting the primary path and the size of these segments, thus providing flexibility in protection design. Also note that, that the segmentation based approach can also result in protection configuration corresponding to the extreme cases of local path protection (each segment consists of one link each) and global path protection (there is only one segment consisting of the entire path) as well. Thus these cases are encapsulated in the Segment Based approach and can be used as and when required.

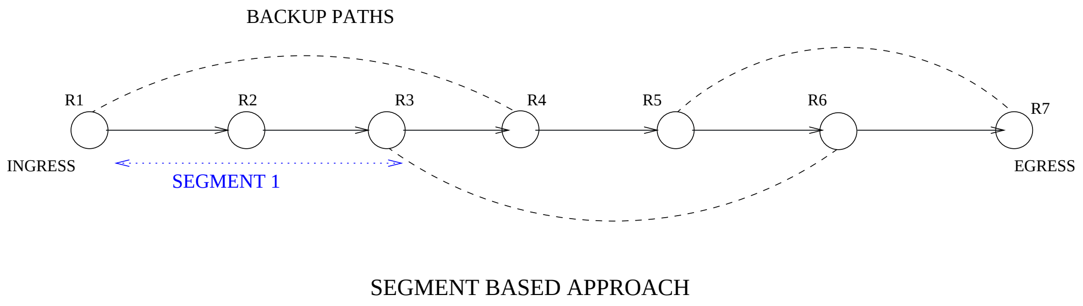


Figure 1.1: Segment Based Approach

Now the main problem that arises is that how to convert the SLA parameters (the QOS and bandwidth constraints) into an efficient segmentation of the entire path. In this project we investigate this problem and present analysis and new algorithms for providing segment-based protection configurations for various QoS constraints like bounded switch over time, end-to-end delay, jitter, reliability or combination of the above. Along with satisfaction of the QoS Constraints, these algorithms will also aim at conservative use of protection resources, using no more than necessary. We look at the mechanisms required for providing path protection. We also consider issues concerning creation of backup paths and possibility of sharing the backup bandwidth among multiple LSPs.

Our experimental Results show significant improvements in protection design, where the satisfaction of certain QoS constraints can be assured while using lesser number of backup paths. Improvement achieved depends on the QOS constraints required for the particular LSP.

Chapter 2

Objective

The main objective of our work is to develop Path Protection Schemes for MPLS Networks which guarantee satisfaction of certain QoS constraints while also conserving the resources used for providing protection.

Path Protection is important in MPLS networks to provide quick recovery in case of a node/link failure and ensure good performance for critical traffic. For achieving this various methods have been proposed, like Local Path Protection and Global Path Protection. However, they are specific in their solutions and do not provide much flexibility in meeting various QOS constraints to be satisfied during the recovery phase and after a failure in the network.

Our new proposed scheme (The Segment Based Approach) is generic in nature and allows lot of flexibility in meeting the QOS constraints (for example switch over delay at time of failure), striving not to use more resources (like bandwidth) than required for meeting the specified QOS constraints.

Design of a Path Protection scheme involves various steps. We plan to present all these steps along with simulation/analytic results to show the advantages of our new approach.

The project can be divided into following modules, which we plan to cover step by step:

- Study of MPLS Networks and the existing work in the area of Path Protection.
- Development of mechanisms for various aspects of Path Protection using Segment Based Approach, which include detection, notification and switching.
- Consideration of various QOS parameters like switch over delay, end-to-end delay and related protection design issues
- Development of algorithms for providing a protection configuration, using the segment based approach, which should be flexible and also helps in conserving resources used by protection resources.

- Experimental studies of various aspects of our scheme to study the advantages offered by the new approach and the performance of various algorithms.
- Simulation of the above algorithms using random network topologies
- Visualization of the above algorithms and Segment Based Protection in action.
- Implementation of Path protection in a real-world network setup.

Chapter 3

Methods for providing Fault Tolerance

To primary and the easiest method to provide fault tolerance in MPLS networks is rerouting the traffic to another path in case of a failure on the primary path (the initial path which was setup for transmission of packets from the ingress to the router)

For rerouting the traffic to an alternate path, in case of a failure, two possibilities exist:

3.1 Dynamic Path Rerouting

These protection mechanisms dynamically create protection entities for restoring traffic, based upon failure information, bandwidth allocation and optimized reroute assignment. Thus, upon detecting failure, the LSPs crossing a failed link or LSR are broken at the point of failure and reestablished using signaling.

3.1.1 Pros and Cons

These methods may increase resource utilization because capacity or bandwidth is not reserved beforehand and is available for use by other (possibly lower priority) traffic, when the protection path does not require this capacity. They may, however, require longer restoration times, since it is difficult to instantaneously switch over to a protection entity, following the detection of a failure.

3.2 Pre-negotiated Protection

These are dedicated protection mechanisms, where for each working path there exists a pre-established protection path, which is node and link disjoint with the primary path, but may merge with other working paths that are disjoint with the primary. The resources (bandwidth, buffers, processing) on the backup entity may be either predetermined and reserved beforehand (and unused), or may be allocated dynamically by displacing lower priority traffic that was allowed to use them in the absence of a failure on the working path.

3.2.1 Pros and cons

This is costlier in terms of the resources used (which can be alleviated to some extent by allowing backup path sharing, in which primary paths share the resources used for providing protection), but advantage offered by pre-negotiated path protection is that there is no delay involved in setting up an alternate protection path in case of a failure, thus reducing the switch over delay.

Our work focuses on the second approach i.e. Pre-negotiated Protection. In our project, we discuss strategies for setting up pre-reserved backup paths using our new approach.

In pre-negotiated path protection, two methods to reserve the protection paths are popular, namely Local Path Protection and Global Path Protection.

3.3 Global Path Protection

In Global Path Protection, an alternate or backup path is re-established starting at the source and terminating at the destination. Thus, protection is always activated on an end-to-end basis, irrespective of where along a working path a failure occurs. This method might be slower than the local repair method discussed below, since the failure information has to propagate all the way back to the source before a protection switch is accomplished. It has the advantage, however, requiring the configuration of only a single backup path for each working path, and the reservation of resources along only that path.

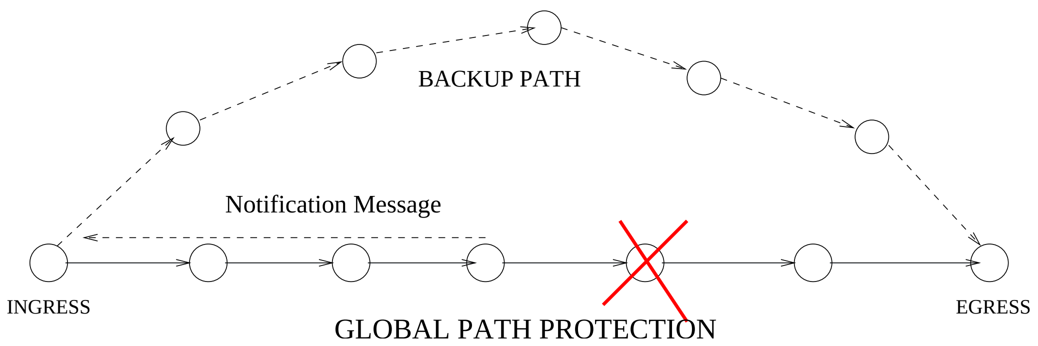


Figure 3.1: Global Path Protection

3.4 Local Path Protection

In local repair, an alternate path exists protecting each node/link on the primary path. Thus protection is activated by each LSR along the path in a distributed fashion on an as-needed basis. While this method has an advantage in terms of the time taken to react to a fault, it introduces the complication that every LSR along a working path may now have the function of switching over to backup path in case of a failure. If resources along each backup segment are reserved a priori and are kept unused, this could result in considerable inefficiency. One trade-off, therefore, is between speed of reaction to a failure and efficient use of resources.

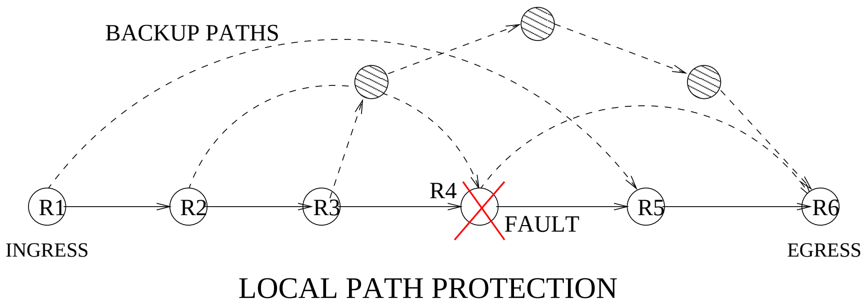


Figure 3.2: Local Path Protection

3.5 Protection Switching Options

Protection switching options refer to the relationship between the active working paths and backup paths, and define how the working entities are protected by the protection entities.

3.6 1+1 Protection

In 1+1 protection, the resources (bandwidth, buffers, processing capacity) on the backup path are fully reserved to carry only working traffic. In MPLS, this bandwidth may be considered wasted. Alternately, this bandwidth could be used to transmit an exact copy of the working traffic, with a selection between the traffic on the working and protection paths being made at the protection merge LSR.

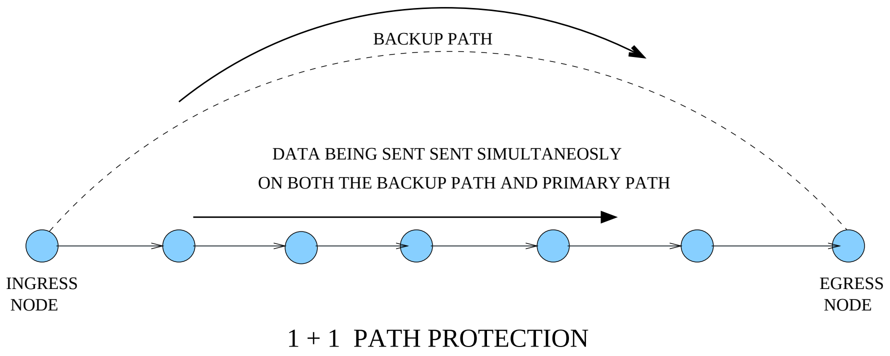


Figure 3.3: 1+1 Path Protection

3.7 1:1, 1:n, and n:m Protection

In 1:1 protection, the resources (bandwidth, buffers, and processing capacity) allocated on the protection path are fully available to preemptable low priority traffic when the protection path is not in use by the working traffic. In other words, in 1:1 protection, the working traffic normally travels only on the working path, and is switched to the protection path only when the working entity is unavailable. Once the protection switch is initiated, all the low priority traffic being carried on the protection path is discarded to free resources for the working traffic. This method affords a way to make efficient use of the backup path, since resources on the protection path are not locked and can be used by other traffic when the backup path is not being used to carry working traffic.

Similarly, in 1:n protection, up to n working paths are protected using only one backup path, while in m:n protection, up to n working paths are protected using up to m backup paths.

3.8 Some Existing propositions to Path Protection

Some of the schemes proposed for path protection are by Haskin, Hakim, Iwata, and others. . They differ in the location of the upstream node, which reroutes the traffic in case of a node/link failure along the downstream path. The basic idea of two of them is given below. One works on Global Path Protection scheme and the other uses Local Path Protection scheme.

3.8.1 Haskin's Scheme

The main idea of Haskin's work is to reverse traffic at the point of failure of the protected LSP back to the source switch of the protected LSP such that the traffic flow can be then redirected via a parallel LSP between source and destination. An alternative unidirectional label switched path is established from the destination switch to the source switch in the reverse direction of the protected path traversing through every protected switch between the destination switch and the source switch. The second part of this alternative path is set between the source switch and the destination switch along a transmission path that does not utilize any protected switches.

3.8.2 Iwata's Scheme

Proposes an enhanced scheme for fast rerouting, which provides means of quickly repairing LSPs in cases of link/node failures. It also provides for the sharing of backup LSPs and their resources and thus allows minimization of requirements for backup resources.

Chapter 4

The Segment Based Approach

4.1 Motivation for our New Scheme

An Internet draft on MPLS Path Protection[draft-makam-mpls-protection-00] states that

> MPLS protection switching should be designed into the existing protocol to give as much flexibility as possible to the network operator. In fact, the operator should have some alternatives to choose from when deciding what type of protection to implement per MPLS LSP. The most logical way to achieve this would be to use alternatives that are realizable by using the mechanisms currently defined in MPLS

The current schemes are quite rigid in their approach and do not offer much flexibility to the domain administrator regarding the reservation of backup paths for critical traffic and may not be adjustable to suit various requirements like tight confirmation to delay bounds and adaptation to the network metrics etc.

Here, we discuss a new approach towards path protection, which is very general and offers a lot of flexibility in offering Path Protection for traffic and has parameters which can be adjusted to suit the delay bound and amount of reservation. Moreover, the existing schemes like local rerouting and global path rerouting can be modeled as special cases of this approach thus covering previous schemes also.

4.2 Considering a new approach

In global path protection we see that the whole path is protected by another backup path. However as we have seen above, the delay for notifying the failure to the ingress router can be quite large. However, the advantage is that only one backup path is needed from the ingress node to the egress.

In local path protection, the neighboring node is responsible for rerouting the traffic.

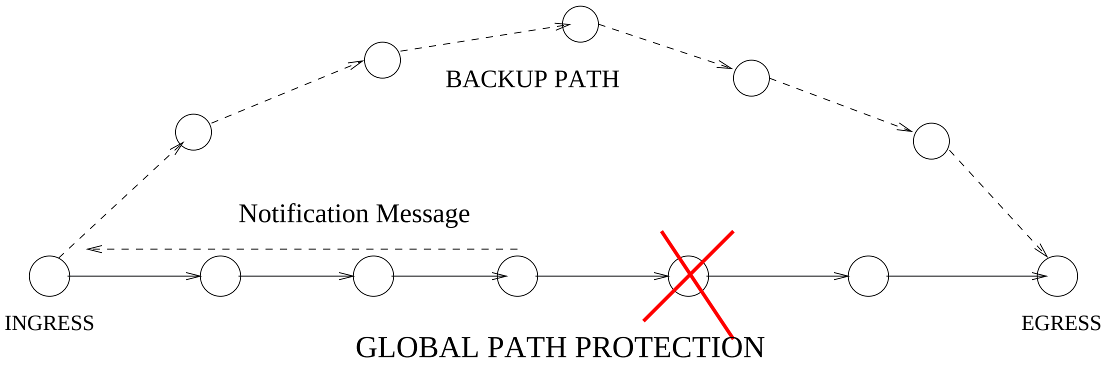


Figure 4.1: Global Path Protection

Here the delay in notifying the upstream router is very less. But here every link needs to be protected by a backup path. Thus number of backup paths needed may be very large thus using lot of resources used for protection. Thus the amount of resources used for providing local path protection can be substantial.

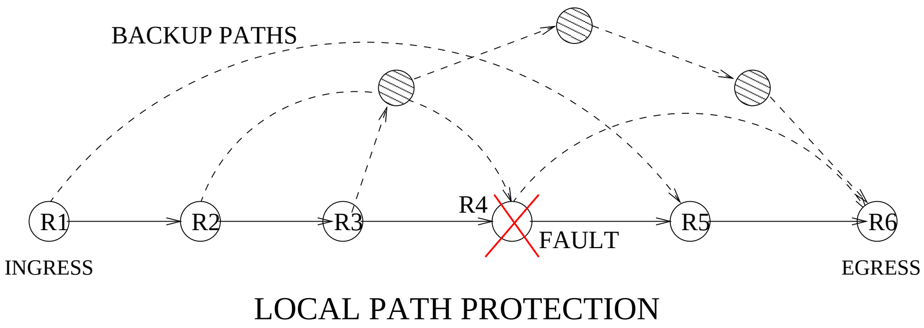


Figure 4.2: Local Path Protection

However, we suggest that these two solutions lie at two extremes and a whole range of other solutions can also be considered which provide more flexibility in setting up the backup paths and in prescribing a trade-off between the rerouting delay and the reservation of backup paths. The segment-based approach is such an approach, which provides such flexibility.

4.3 Introducing the Segment Based Approach

Let us understand how the Segment Based approach can be applied by an example. Consider a single primary path shown below from the ingress to the egress router.

The path consists of 7 links and 8 routers , with the link property specified for

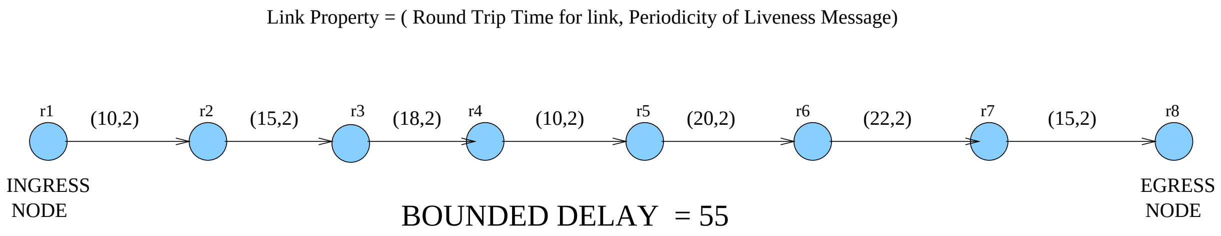


Figure 4.3: A Sample Primary Path

each link. Here we characterize each link by the pair (Round Trip Time (ms) , Periodicity of liveness message(ms)). RTT refers to the sum of delays encountered in transmission of a data packet from the upstream router to the downstream router and then back. RTT can be defined as One Way Delay (R_i, R_{i+1}) + One Way Delay $(R_i + 1, Ri)$ for the link connecting the routers R_i , R_{i+1}.

Liveness message refers to a mechanism for detecting faults in a network using a "Hello" message, which is exchanged periodically between neighboring routers in a network. Failure to receive a "Hello" message after a certain time-out period indicates that the neighboring link or router is down. We denote the periodicity by T_{test}.

Now, suppose we want to provide protection to the above primary path with the given QOS constraint that on a single failure on this path, the switch over time is no more than 55 milliseconds i.e. packets are lost for not more than 55 milliseconds in case of failure.

We shall later show by analysis that if router Ri is responsible for switching the traffic in case of a failure at the downstream router Rj, then the switch over time can be bounded by the expression

$RTT(R_i, R_j) + t_{test} < \delta$

Where δ is the upper bound on switch over time, 55 milliseconds in our case. We assume the above expression now for explaining the current example.

Please note that when R_j fails, its immediate upstream neighbor R_{j-1} sends the notification to R_i, and T_{test} is for routers (R_{j-1}, R_j).

Let us first of all consider the approach of providing protection by the Local Path Protection method. In this approach, we protect each link separately. If we provide protection using this approach, the following Path Protection configuration is setup.

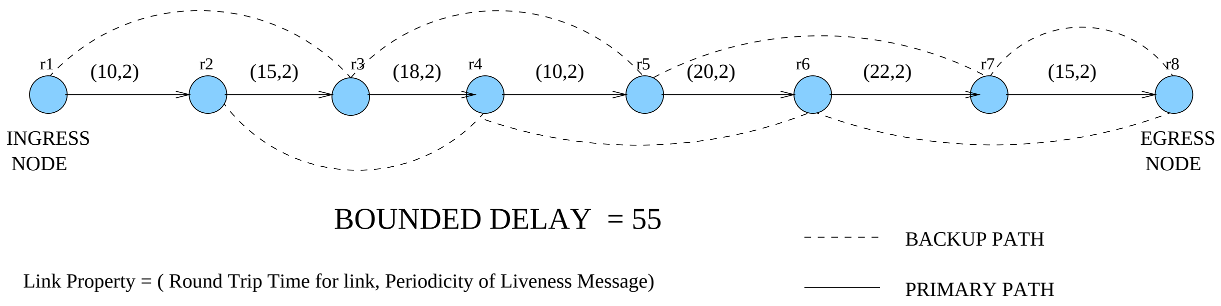


Figure 4.4: Protection using Local Based Approach

Here the number of backup paths established are 7, the same number as the number of links. We also note that the above configuration meets the switch time bound of 55 milliseconds very easily.

But the point to note here is that maybe the same constraint can be met by using lesser number of backup paths i.e. we can meet the constraints more "tightly" , and use less resources in the process. For example consider the protection configuration below:

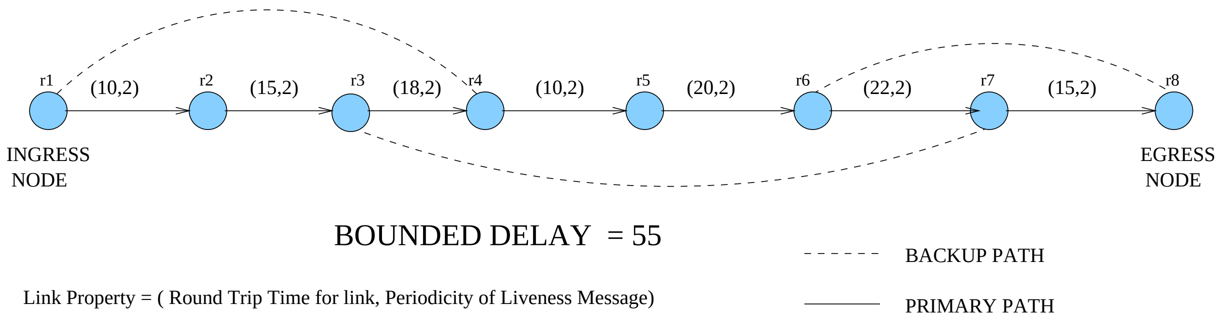


Figure 4.5: Another way to protect the path

This Protection configuration also meets the upper bound of 55 milliseconds (in a tighter way) but here only 3 backup paths are required instead of 7. Thus, this strategy appears to be much better than over-providing for protection, which uses much more resources.

The latter approach can be viewed as dividing the primary path into segments and then protecting each segment separately. Hence the name Segment Based Approach.

We considered the case of bounded Switch over time in the above example. But besides delay, one can also consider other constraints such as reliability measure, end-to-end delay etc. while segmenting the primary path.

Of course, we need to figure out appropriate algorithms to divide the primary path into segments in an optimal manner. Since various QOS constraints exist, different algorithms may need to be developed for each separately. Multiple QOS constraints may also be need to met together, thus calling for more complex algorithms. One of the aims of these algorithms is to minimize the number of segments, thus using lesser number of backup paths, while meeting the QOS constraints.

We present these algorithms for various QOS constraints and their design approach in Chapter 7.

4.4 Terminology

Some new terminology introduced by the segment-based approach, which will be used in the later discussions, is listed below.

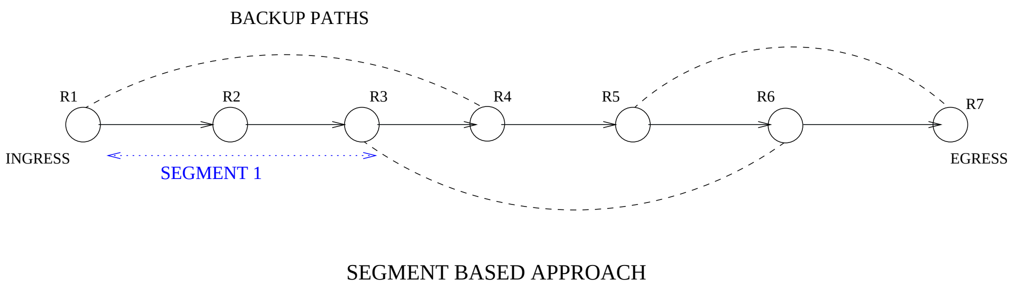


Figure 4.6: Segment Based Approach

Segment

A segment is defined as a collection of links and routers, which are protected by a single backup path. In the above figure, the nodes R2, R3 and the links (R_1, R_2) and (R_2, R_3) constitute one segment. Please note that if R1 fails, there is no way to protect, as R1 is the source router. Also the last router R3 is not a part of the next segment but the first segment indicated.

The above segment can be denoted as $< R_1, R_3 >$.

Segment Switch Router (SSR)

SSR or Segment Switch Router is the router, which is responsible for switching the traffic to a backup path in case of a failure in the segment. In the above figure, the routers R1, R3 and R5 serve as Segment Switch Routers since they are the origin of backup path for that particular segment and responsible for switching over the traffic.

Note that there is a SSR for each segment.

Chapter 5

Mechanisms of Fault Detection

Consider an LSP $< R_1, ..., R_n >$ where R_1 is the ingress router and is the R_n is the egress router. In the above figure, a segment $< R_2, R_6 >$ has been established. In case of a failure in this segment R_2 (which is the Segment Switch Router) is responsible for switching over the traffic to the alternate backup path in case of a failure.

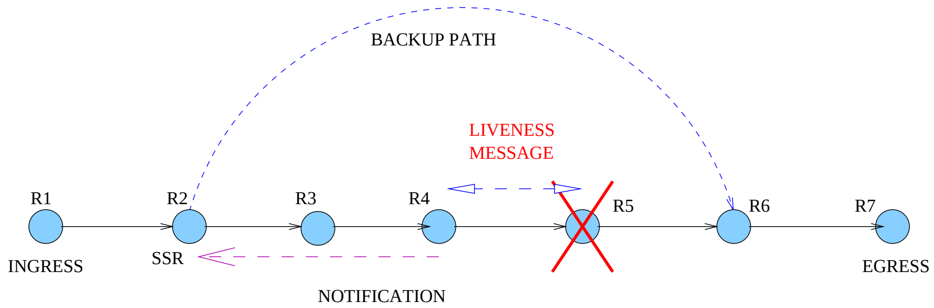


Figure 5.1: Mechanism for Fault Detection

Now for detecting the fault, there are two approaches with the Liveness message mechanism.

5.1 1st Approach

In this approach, the Liveness message is exchanged between the Segment Switch Router and the last router of that particular segment. In the above case, the Liveness message will be exchanged between routers R_2 and R_6.

However, with this approach there are some problems:

1. This test must be carried out separately for each LSP. In this scenario it is possible

that more than one Liveness message are being transmitted over the same link.

2. Using this method, extra signaling needs to be carried out to locate the fault.

5.2 2nd Approach

In a totally different approach, the Liveness message is exchanged along the working path between the peer LSRs. Each LSR sends the Liveness message to all its neighbors and in case of timeout, infers the failure of the downstream link or node.

The advantage of this approach is that the test can be carried out independent of each LSP that goes over the link connecting the two LSRs. Also locating the fault is trivial in this case as the immediate upstream LSR detects the fault in this case.

This approach does not address at the present the issue of how does the segment switch router for the segment, which has the responsibility to switch over the traffic receives the notification of node failure. We assume for now that a downstream node sends a "failure notification" to every upstream node. And that it does so for every LSP affected by the node failure.

By timing analysis of the two approaches above we can find out that there is no fundamental difference in the delay in detecting and notifying failure in the above two schemes. However, the later scheme is more efficient in that there is only one test message per link, and is independent of the LSPs that go over the link.

5.3 Signaling the Liveness Message

The Liveness messages can be encoded into a LDP packet, which provides the option of TLVs (Type-Length-Value) using which various kinds of control information can be signaled to the LSRs.

5.4 Fault Notification Mechanism

In this section, we look at schemes for fault notification. We assume that each node is responsible for detecting the failure of its neighboring nodes, and that such a failure signals a breakdown of all LSPs that pass through the failed node.

Let us consider a simple scheme, one that returns a notification to a specified upstream router as quickly as possible, even though it may be bandwidth intensive.

Consider an LSP between the ingress and egress routers , R_0 and R_n. We assume thatR_0 is the LSR which is responsible for switching traffic if there is a failure of any node $R_1, R_2, ..., R_{n-1}$. Let R_k be the router to detect failure of node R_{k+1} , then R_k

sends a notification message to R_{k-1} to signal that ”the remaining portion of the LSP has failed”. But , if R_k is itself R_0 , then it simply switches traffic over the backup path.

Now , let us consider the case of multiple LSPs being affected by the failure of a node. In this case , the LSR R_k which detects the fault does one of the two things :

1. If R_k is the ingress node then it switches traffic along the backup path.
2. Otherwise, it sends a notification to the upstream node corresponding to each particular LSP passing through that link(Thus, R_k sends multiple notification messages , one for each LSP).

5.5 Fault Notification in case of Multiple LSPs

After the fault has been detected and localized it must be notified to the SSR so that it can reroute the traffic affected by the fault to the reserved backup path for this segment. Earlier we have done the delay analysis for notification.

We can note that when a fault occurs within a segment, some nodes will be responsible for passing the notification to the appropriate upstream LSRs and each SSR must not pass it above but carry out the rerouting function (unless the SSR is also an internal node of another segment in the case where segments overlap for different primary paths). There must be some mechanism so that these operations can be smoothly carried out. The following issues need to be addressed:

1. Which information needs to be passed inside the notification message?
2. How will SSR proceed to switch the route when it receives the notification message?

Please note that although the SSR may be responsible for protecting and switching packets for many LSPs, only a subset of these may actually be affected due to a particular link/node failure. We need to communicate this effectively to the SSR so that it can switch packets only for the affected LSPs. All this is show in the example discussed below.

When LSR R_4 fails, R_3 detects the error. Then it must propagate the notification message to the SSR for this segment, which is R_1. Note that another LSP also has R_1 as its SSR but does not pass through the link $< R_3, R_4 >$. So R_1 must switch only the LSP, which passes through the link $< R_3, R_4 >$. For propagating this information to

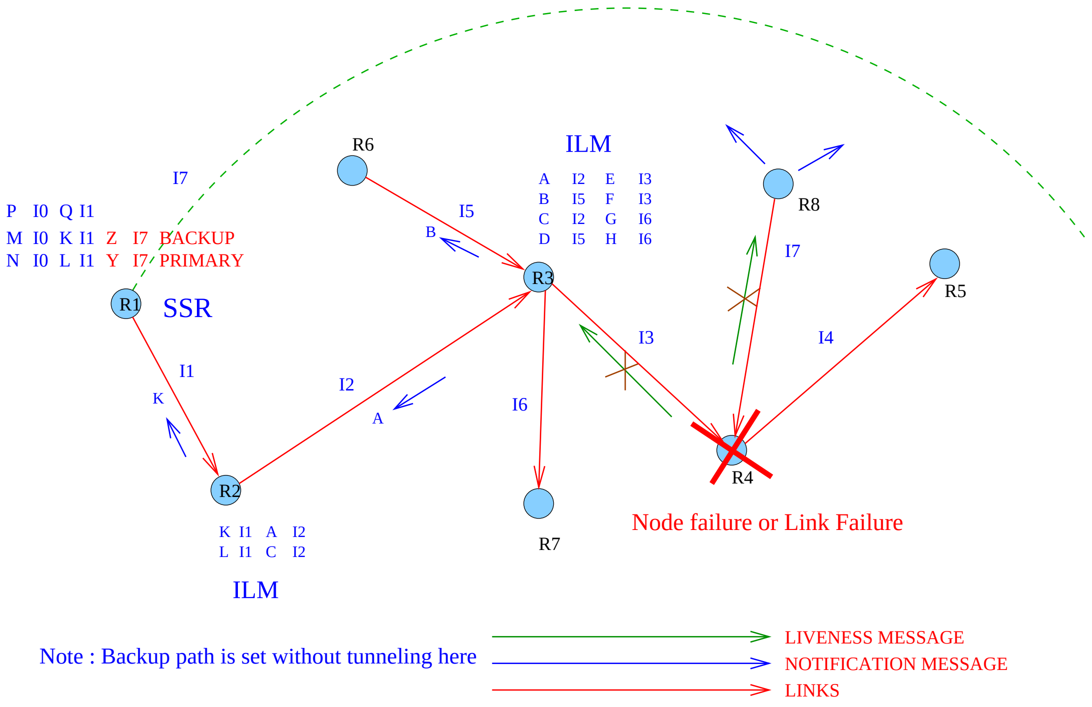


Figure 5.2: A Mechanism for Notification

R_1, R_3 sees which incoming labels are routed to $< R_3, R_4 >$ using the outgoing interface column in the NHLFE. [1] Then it forwards the notification message to containing the labels A and B on incoming interfaces I2 and I5 respectively to notify the LSRs R_2 and R_6 about the labels, which cannot be transmitted due to error on the primary path. Then R_2 sees that since label A cannot be transmitted, it again consults its NHLFE to see which incoming labels map to label A on I2. Thus it propagates Label K further on I1. R_1 on receiving label K in the notification message looks up all incoming labels in the NHLFE, which are mapped to label K on the outgoing interface I1. On consulting its NHLFE it sees that all incoming packets with label M should be rerouted since it is the SSR for them. After referring to rerouting information stored in its NHLFE, it reroutes all incoming packets with label M to interface I7.

[1] Next Hop Label Forwarding Entry: A table stored by each LSR which maintains the Label Swapping information

Chapter 6

Issues in Setting up of Backup Paths

6.1 Creation of Backup Paths

Creating the backup paths for each segment is an important step in the whole process. One advantage of the segment-based approach is that while still meeting the delay bound, one need not create a backup path for each node/link (as in local restoration) but instead must provide a backup path for the whole segment. This may result in less resources being consumed by the reservation of the backup path and thus increase the capacity of the network while still meeting the delay bounds or other criteria. Indeed our experimental results, which are presented later, confirm this.

Generally, the set of constraints and objectives that went into deciding the specific route for the given LSP will also be the ones to be used to determine suitability if the backup path. But, there will also be additional constraints such as:

Consider an LSP $< R_1, ..., R_i, ..., R_n >$ and a segment $< R_i, ..., R_j >$.

1. The backup path must be node-disjoint with all the nodes on the entire segment , other than R_i. If any of the nodes $R_{i+1}, ..., R_j$ fails, R_i switches traffic over a backup path which is node disjoint from $R_{i+1}, ..., R_j$. It can terminate at any of the nodes, $R_{j+1}, ..., R_n$.

2. The reserved backup paths must not create loops in the path. The backup paths created for a segment must be node-disjoint from $R_1, ...R_{i-1}$ along with the above condition. Otherwise, a loop will be formed and packets, which are rerouted along this backup path, will loop around forever.

3. When creating a backup path for a segment, we may also need to consider the length of the backup path. If the end-to-end delay from the SSR to the final node of the backup path is very high then this will affect the rerouting delay

also. We call this Backup Path Delay (BPD). We must ensure that BPD is not very large otherwise the rerouting delay may exceed the specified bounds and affect some performance metrics such as jitter. This is further discussed later.

6.2 Creating backup paths using tunnels

The MPLS Architecture Specification gives the concept of tunnels, using which a hierarchy of LSP paths can be created and packets can be sent from one LSR to another non-peer LSR using a tunnel.

The backup paths can be created using tunnels. The backup path will be at a higher level than the primary path.

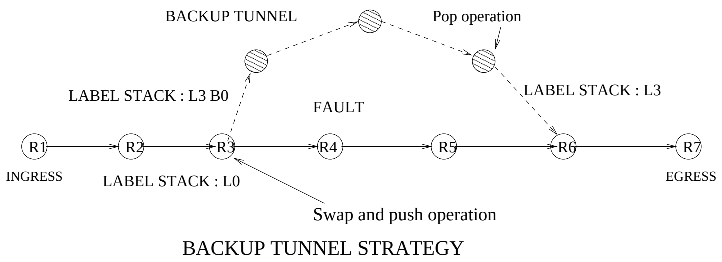


Figure 6.1: Creating Backup Paths using tunnels

After creating the backup paths and reserving bandwidth for them, each SSR can route the packets in case of a failure, to a backup tunnel whose information is already available to the SSR. To send the packet over the backup tunnel, the SSR needs to do swap and push operations over the label stack.

In the swap operation, the SSR will pop the current label and push a label such that the LSR on the primary path where the backup tunnels ends perceives the packets coming from its upstream LSR also the primary path.

Then the SSR needs to push another label corresponding to the new LSP tunnel. Thus the backup tunnel uses this new layer of the label stack to route the packet to the end of the tunnel. The penultimate LSR of the backup tunnel then pops the label at this layer, thus making the label at the lower layer visible. Then the next LSR, which is on the primary path of the LSP, perceives as it has received the packet from its upstream LSR on the primary path.

6.3 Changes to the NHLFE (Next Hop Label Forwarding Entry)

The SSR also needs to store information corresponding to the rerouting of traffic to the backup tunnels, in addition to the usual Label Swapping information.

The SSR needs to keep a Boolean field, which will indicate whether the incoming packets for a particular LSP are using the primary path or the backup path. In addition to this field, it also needs to store two labels. One of these is the label, which will be perceived by the end LSR of tunnel and the other label (the tunnel routing label) is added on top of the label stack to route the packets along the backup tunnel. The table also needs to store the outgoing interface of the backup tunnel.

6.4 Sharing of backup paths

Since we assume pre-allocation of backup path resources, primary traffic will not be able to use the resources reserved by the backup paths. Therefore, sharing of backup paths is highly desired otherwise backup paths may consume a large part of the available networking resources.

However, there are also some issues, which need to be discussed. Sharing assumes that there will be at most one fault in the network at one time. Otherwise, if two faults occur which use the same backup paths, their bandwidth cannot be shared. Sharing can be done in two ways:

1. Multiple LSPs, same segment If a SSR is responsible for switching k LSPs in case of a failure, the backup paths created should be able to handle the traffic of all the LSPs. In this case, there can not be any sharing since all the traffic must be rerouted together.

   If the bandwidth used by the k LSPs is $b_1, ..., b_k$ then the bandwidth required on the backup path is $\sum_{i=1}^{k} b_i$.

2. Multiple LSPs, different segments In this case, since only one fault at a time is assumed, we can share the backup path among these LSPs. Then the bandwidth required on the backup path is $\max(b_1, b_2, ..., b_k)$ for the case that the LSP with the highest bandwidth requirement develops a fault in the corresponding segment for this backup path.

Chapter 7

Algorithms for various QoS constraints

7.1 Introduction

This part of the project deals with the design of algorithms for providing efficient fault tolerant mechanisms in case of a network component failure. In case of a failure, we need to deal with in a manner which is transparent to the user and which also meets some specified performance constraints. We will study various performance constraints and then we will present algorithms to deal with a network failure while meeting the specified constraints. Its importance lies in providing good performance for today's critical applications like VoIP, video transmission, other real-time traffic even in case of a failure. Thus the goal is to provide reliable networks with certain performance bounds.

7.1.1 Approach

For providing efficient fault tolerance, we will be designing algorithms using the idea of path protection. Though many possibilities exist to take care of a failure in a network (like dynamic routing etc.), path protection provides very low rerouting delays as the alternate routes are already reserved. However, the setup of these alternate routes needs to be considered according the network performance criteria such as conservation of bandwidth and other QoS parameters. We will discuss how to setup the primary and alternate routes to meet the desired criteria.

7.1.2 Segment Based Path Protection

Protection of the primary path has to be provided in a manner such that it meets the performance constraints. Existing protection schemes like Local path protection and Global Path Protection are special cases of protection and don't provide much flexibility in meeting performance constraints. Here, we consider a new approach to protection called Segment Based Path Protection. Here we don't follow a fixed paradigm for allocating alternate paths but instead allow partial path protection.

The structure/configuration of the backup paths which will be reserved for providing protection to the primary path depends on the performance constraints to be met. We need to figure out methods which tell us the specific number and the configuration of the backup paths which will meet the required performance constraints.

In the Segment Based Approach, We consider the idea of dividing the entire primary path into segments (based on the performance criteria to be met) and then protect these segments with alternate routes. This gives us a lot of flexibility in providing protection as the freedom to choose the segments gives us flexibility in meeting the constraints. Other issues are also involved such as conservative use of backup bandwidth while meeting the performance constraints at the same time. As we shall see, there exists a trade-off between the amount of bandwidth used up by the backup paths vs meeting the performance constraints. So we would like to meet these constraints as tightly as possible, so as not to use excess of resources than otherwise required.

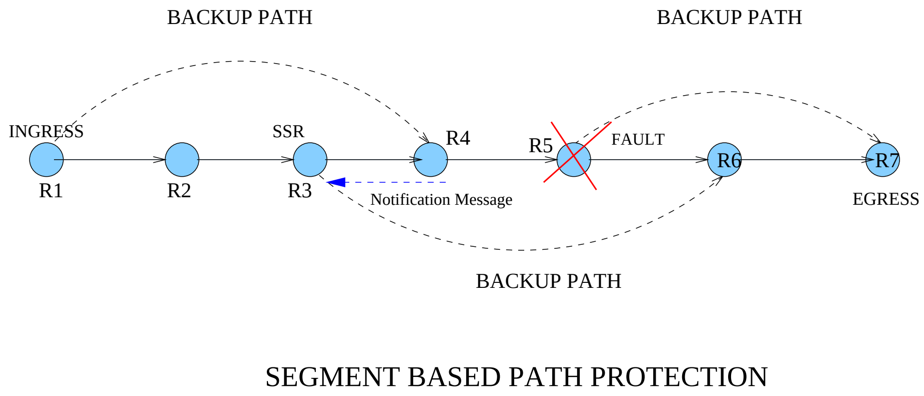


Figure 7.1: Segment Based Path Protection

In this report , we shall discuss in detail how we can provide various types of performance guarantees using the segment based approach.

7.1.3 Assumptions

We assume one fault at a time in the network. When a fault occurs, the backup path will not be used for a very long period of time and the ingress router tries to find another primary path as soon as possible if the current primary path doesn't recover.

7.1.4 Performance Parameters

- **Switch-over time bounds:** Switch-over time is the time for which the packets along the LSP will be dropped in case of a failure of link/node.

- **End-to-End Delay:** This is the propagation time of a packet to reach the destination node from the source.
- **Reliability:** This is a measure of the probability of nodes being reachable to each other.
- **Jitter:** Jitter is the deviation from the ideal timing of receiving a packet at the destination.
- **Conservation of Network Resources:** To provide fault tolerance, we need to use additional network resources to meet the specified criteria. However, a trade-off exists between the satisfiability of the criteria and the amount of resources used. We need to ensure optimal network resource usage while providing fault tolerance.

The first four are relevant from the service users' point of view. The last parameter is relevant for the service provider.

7.2 Modeling the network

We will be using a graph-theoretic approach to model the network and to describe the algorithms. The routing devices will be modeled as nodes and the links as edges. The edges will be directed to constrain the transmission direction of packets along the links. Let $N = (V, E)$ describe the given network , where V is a set of n tuples , each tuple denoting a router and its associated properties. E is a set of m tuples , each tuple denoting a link and its associated properties such as delay, reliability etc.

LSP
LSP will be designated as the tuple $< R_0, .., R_i, .., R_n >$ where R_i is the i^{th} router along the path.

Segments
A path is divided into several non-overlapping segments , which cover the path completely. The Segment Switch Router will be denoted as S_i which is the i^{th} SSR on that path. Please see the diagram to see the notation in use.

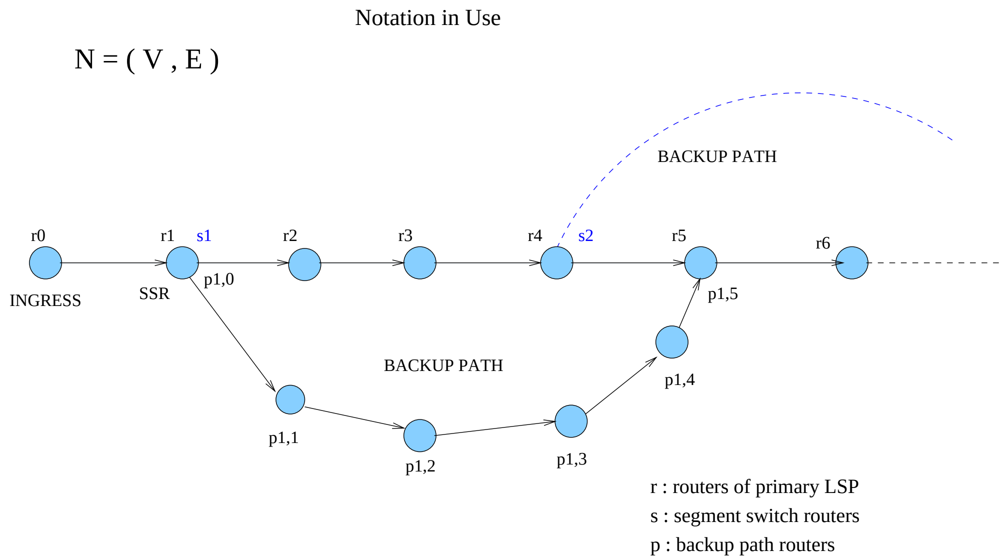


Figure 7.2: Modeling the Network

By saying $R_i < R_j$ we mean that R_i occurs before R_j in the LSP $< R_0, ..., R_n >$ i.e. $i < j$.

Backup Paths
Each segment will be protected by a backup path. The backup path for the i^{th} segment (originating at S_i) will be denoted by $\{< P_{i_0}, ..., P_{i_n} >, i = 0, ..., k-1\}$ for a

backup path consisting of $n+1$ routers in the network. Note that P_{i_0} is same as S_i.

7.3 Switch-over Time

7.3.1 Introduction

In case of a network element failure , some packets will be lost along the LSP to which that element belongs. We want to minimize this loss. In this section we will describe an algorithm which will provide bounds to the amount of packet loss.

Switch-Over Time is an important parameter in communications. It refers to the time for which the packets will be dropped in case of a failure. For example in a Voice conservation over a packet network , it refers to the time for which the voice packets will not reach the destination because of being dropped due to some failure in the network. For this much time , the destination user will not hear anything.

We will focus on meeting bounds on Switch-over time in this section. We want to protect the primary path so that in case of a failure along the path, packets are dropped for a time no more than delta. As described earlier, we use the new concept of segments. We want to divide the whole path into segments such that each segment is protected independently by an alternate path and in case of a failure in a particular segment, the traffic can be switched at the head node of that segment along the alternate path. However, the configuration of alternate routes and segment should be such that the packet loss time bounds are respected.

Conserving Protection Resources
While meeting the Packet loss bounds, we also want to conserve the amount of bandwidth used. Therefore we want to use the least amount of protection resources (i.e. the bandwidth used by the pre-allocation alternate routes used for protecting the segments). For this , the least number of segments are desired so that we can minimize the number of alternate routes used. Experimental results have shown that over a random network , lesser number of segments(hence larger segments) use less protection resources.

7.3.2 Analysis

In this section we derive the expression assumed previously: $RTT(R_i, R_j) + T_{test} < \delta$, where delta is the upper bound on switch over time.

Consider the above figure showing a situation in which one of the router has failed (router R_5). In the above case, R_2 is the segment switch router of the segment to which R_5 belongs hence it is responsible for switching over the traffic to the backup path shown.

For this to happen, R_2 must somehow know of the failure, which has occurred down-

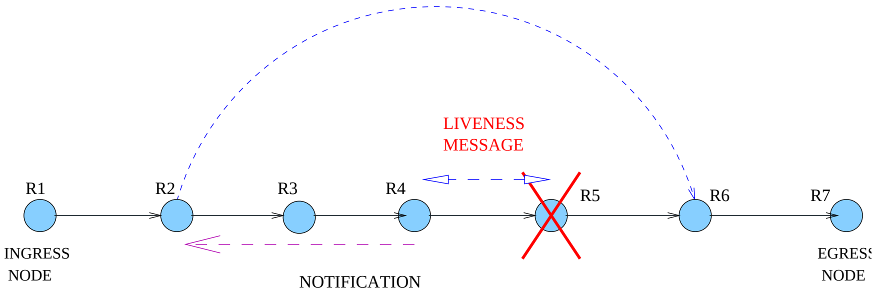


Figure 7.3: Notification Mechanism

stream. As discussed previously, its upstream neighbor R_4 will detect the failure. Since it is not the SSR, it will notify its upstream neighbor R_3, which in turn notifies R_2. Now when R_2 received the notification, since it is the SSR, it switches over the traffic for the failed segment to the backup path.

Obviously, some data packets will be lost in the above scenario. This time (known as the switch over time) for which the data packets are dropped is bounded by QoS constraints, thus we need to make sure that the whole mechanism involving detection, notification etc. doesn't exceed this bound. Below, we analyze the switch over time more closely in terms of known network parameters, so that we can come up with a closed expression, which is vital for the development of an algorithm for segmenting the primary path.

We assume the following parameters:
T_{test} : Periodicity of the Liveness message which is exchanged between the neighboring routers (for example here R_4 and R_5)
$OWD(R_i, R_j)$: One Way Delay is the transmission delay for a data packet from router R_i to reach router R_j.
$RTT(R_i, R_j)$: Round Trip Time for the router pair (R_i, R_j). Note that RTT$(R_i, R_j) = OWD(R_i, R_j) + OWD(R_j, R_i)$.

Let t be the time when the node R_5 fails. Then in the worst case , the Liveness message sent by R_4 reaches R_5 at $t + t_{test}$. After $OWD(R_5, R_4) + \epsilon$, R_4 will come to know that there is a problem with the downstream link or node. Thus it detects the failure at time $t + t_{test} + OWD(R_5, R_4)$. After detecting the failure, it sends the notification to its upstream router and in this way it reaches the Segment Switch Router R_2 after time $OWD(R_4, R_2)$. (Note here that we neglect the processing times involved in the router for forwarding the notification to the appropriate link). Thus, router R_2 which is responsible for switching over the traffic receives the notification at time $t + t_{test} + OWD(R_5, R_4) + OWD(R_4, R_2) + \epsilon$ which is equivalent to $t + t_{test} + OWD(R_5, R_2) + \epsilon$ (in the worst case). For this time period, the packets entering the segment will be dropped as R_5 is down.

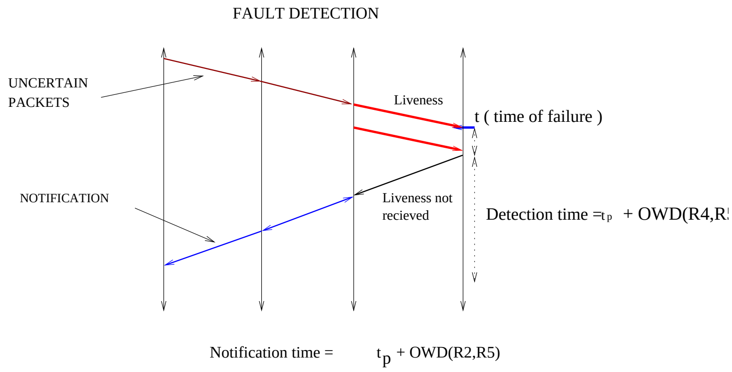


Figure 7.4: Finding switch-over time

The above figure shows the analysis in a timing diagram. Vertical axis denotes time. The worst case will occur when the router goes down, immediately after receiving a Liveness message from its upstream router.

Now, also note that since the router fails at time t , the fate of the packets which enter this segment after $t - OWD(R_2, R_5)$ is uncertain. By the time, they will reach the router R_5, it would already have gone down. This is also shown in the above figure.

Thus, it can be seen that the from time $t - OWD(R_2, R_5)$ to time $t + t_{test} + OWD(R_5, R_2) + \epsilon$, the packets will be dropped. This the total time for which packets will be dropped is $t_{test} + OWD(R_5, R_2) + OWD(R_2, R_5) + \epsilon$ which is equivalent to $t_{test} + RTT(R_2, R_5) + \epsilon$.

For a general expression, for the segment $< R_i, R_{i+1}, ..., R_j >$, in the worst case (when R_j fails) , the packets will be lost for time $t_{test} + RTT(R_i, R_j) + \epsilon$.

7.3.3 An interesting comparison of SBPP to Global Path Protection

It is desirable to look upon an LSP as consisting of one or more segments, and protecting each segment with a backup path if a router along a segment fails. The motivation behind this comes from the fact that the tine it takes to detect failure along the entire LSP and to notify the same to the ingress router may be unacceptably large.

It can be shown that if node R_k fails then the delay in determining the fault has occurred may be as large as $t_{test}(R_{k-1}, R_k) + RTT(R_0, R_k)$, where $t_{test}(R_{k-1}, R_k)$ is the periodicity of testing reachability, and $RTT(R_0, R_k)$ is the round-trip delay between the ingress router R_0 and the failed node R_k.

Maximum delay in detecting fault

Maximum delay in detecting fault is defined as follows : if a packet is sent byR_0 at time t, then in the worst case, it is only at time $t + t_{test}(R_{k-1}, R_k) + RTT(R_0, R_k)$ that it is determined that a downstream router is unreachable.

As a result , if a router detects failure at time t and switches traffic to an alternate path, then under worst-case conditions, one may lose packets sent by the ingress router over a time interval $(t - t_{test}(R_{k-1}, R_k) - RTT(R_0, R_k), t)$

For the present we assume that t_{test} is the same for all links. In that case, one may lose packets over $(t - t_{test} - RTT(R_0, R_k), t)$

We also refer to $t_{test} + RTT(R_0, R_k)$ as the bound on switch-over time.

It can be also be established that even under the most favorable conditions one would lose all packets sent by the ingress router over the time interval, $(t - RTT(R_0, R_k), t)$ where t is the time the ingress router expects to receive a response from R_k, but does not since R_k fails.

From the above discussion, we can clearly see that there are two ways to reduce the bound on switch-over time: by sending test messages more frequently, or by reducing the round-trip delay. Clearly, there are limitation to doing either.

Alternatively, if an LSP were broken down into a number of segments, and if there was a way to switch traffic whenever there is a breakdown in a segment, then the switch-over time is bounded by $t_{test} + RTT(< R^i, R^{i+1} >)$ for the segment $(< R^i, R^{i+1} >)$. As a consequence , one can bound the switch over time to delta by ensuring that for each segment:
$t_{test} + RTT(< R^i, R^{i+1} >) <= \delta$.

Therefore the idea of dividing the LSP into segments may help in providing tight delay bounds compared to the global protection scheme where the maximum delay in detecting the fault may be unacceptable.

7.3.4 Problem Statement

Given a LSP $< R_0, ..., R_i, ..., R_n >$, and an upper bound δ, identify k segments $\{< S_i, S_{i+1} >, i = 0, 1, ..., k-1\}$, such that

- $S_0 = R_0$
- $S_k = R_n$
- $RTT(< S_i, S_{i+1} >) + t_{test} <= \delta$

- k is minimum.

where RTT is the round-trip time and t_{test} is the periodicity of liveness messages which is same for all links.

7.3.5 Approach

We need to know the basis on which we will divide the path into segments. In case of a failure within a segment, we want to notify the head node as early as possible so that the head node can switch traffic to an alternate route pre-allocated for this purpose. In case of a failure , some time will be spent on detecting the fault and then notifying it to the head node.

By earlier analysis , it was shown that if a failure occurs at time t, then the head node can be specified at time $t + RTT(S_i, S_{i+1}) + T_{test}$ in the worst case, where S_i refers to the head node of the i^{th} segment and T_{test} is the periodicity value of the liveness messages exchanged by adjacent routers to test the "upness" of their peers.

Hence, we have to divide the path into segments such that for each segment the condition: $RTT(S_i, S_{i+1}) + T_{test} < \delta$ is satisfied where delta is the maximum permissible packet loss time.

If we meet this requirement for each segment, then we can satisfy the packet loss requirement for the entire path. We also want the fewest number of segments to use less protection resources overall.

Hence, we describe a greedy algorithm to identify the segments. At each step it chooses the largest segment possible such that it satisfies the packet loss time bounds. This way it segments the entire path into such segments.

7.3.6 Algorithm

$SegmentAlgo(< R_0..R_n >, \delta, RTT)$
{
    $segment \leftarrow 0;$
    $first \leftarrow 0;$
    $while(first < n)do$
    {
        $S_{segment} \leftarrow R_{first};$
        $segment \leftarrow segment + 1;$
        $identify\_next\_segment(first, n, last, \delta, RTT);$
        $first \leftarrow last;$
    }
    $S_{segment} \leftarrow R_n;$
}

```
identify_next_segment(first, n, last, δ, RTT)
{
    last ← first + 1;
    RTT ← t_test + RTT(R_first, R_last);
    next ← last + 1;
    while(next <= n)and(RTT + RTT(R_last, R_next) <= δ)do
    {
        RTT ← RTT + RTT(last, next);
        last ← next;
        next ← next + 1;
    }
}
```

7.3.7 Proofs

Minimum Number of Segments

Proof Statement: The algorithm in Section 7.3.6 finds minimum number of segments.

Let the segments generated by the algorithm in Section 7.3.6 be $\{< S_i, S_{i+1} >, i = 0, 1, ..., k-1\}$, and suppose that there exists a segmentation $\{< S'_i, S'_{i+1} >, i = 0, 1, ..., n-1\}$ where $n < k$ and which satisfy the given constraints on the LSP $< R_0, ..., R_n >$

If we prove that $S_i >= S'_i$ for all i then we will get a contradiction, as then $S_{k-1} \geq S'_{k-1}$ and the number of segments required $n >= k$. This can be proved easily using induction.

Base Case: Since $RTT(< S_0, S_1 >) + t_{test} <= \delta$ and $RTT(< S_0, S_1 + m >) + t_{test} > \delta$,for all $m > 0$, $S'_1 <= S_1$ to satisfy the constraint on δ.

Induction Hypothesis: Let $S_p >= S'_p$ for some p, $0 < p <= k-1$.

Induction Step:
From the algorithm
$RTT(< S_p, S_{p+1} >) + t_{test} <= \delta$ and
$RTT(< S_p, S_{p+1} + m1 >) + t_{test} > \delta$ for all $m1 > 0$.
Therefore
$RTT(< S_p - m2, S_{p+1} + m1 >) + t_{test} > \delta$ for all $m1 > 0$ and $m2 >= 0$.
From the induction hypothesis
$S'_p = S_p - m2$ for some $m2 >= 0$.
Therefore
$RTT(< S'_p, S_{p+1} + m1 >) + t_{test} > \delta$ for all $m1 > 0$.

$\Rightarrow S'_{p+1} < S_{p+1} + 1$
$\Rightarrow S'_{p+1} <= S_{p+1}$.

7.4 Consideration of Backup Paths

7.4.1 Introduction

In the algorithm presented in the previous chapter, an algorithm was developed for dividing the primary path into minimum number of segments while meeting the constraints at the same time.

However, there was no consideration of reserving alternate routes while constructing the segments and the network topology (e.g. availability of protection resources , network layout) was not taken into account. But in some cases , it may not be possible to construct backup routes for the segments if a backup route doesn't exist for a particular segment after applying the above algorithm (or even a alternate path exists it may not be possible to reserve protection resources over it). So , while dividing the path into segments, we must also take into consideration the reservation of backup paths. One example to illustrate this is show below:

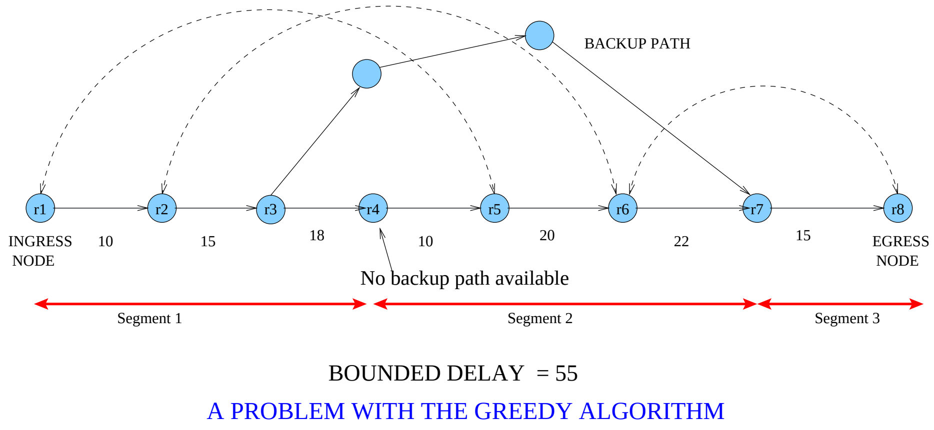


Figure 7.5: Problem with Greedy Algorithm

Here we see that for one of the SSRs there is no backup path available. If we use the greedy algorithm, we may not be able to give admission to the new LSP. However if we allow some tweaking for placement of the SSRs, we may be able to find a backup path.

Also, even if a backup path may exist for a SSR, bandwidth on that path may already be reserved for protection of other segments hence that backup path may be unusable. Hence we need to change the SSR to find the possibility of another backup path.

Therefore, we present a new algorithm, which takes into consideration the above problem and follows an "Adaptive Segment Creation" process. The algorithm adjusts the segments in the LSP to ensure availability of backup paths, and also ensures minimum number of segments

7.4.2 Problem Statement

Given a Network N(V,E), a LSP $< R_0, ..., R_i, ..., R_n >$ and an upper bound δ, identify k segments $\{< S_i, S_{i+1} >, i = 0, ..., k-1\}$ and backup paths $\{< P_{i_0}, ..., P_{i_m} >, i = 0, ..., k-1\}$, such that

- $S_0 = R_0$
- $S_k = R_n$
- $RTT(< S_i, S_{i+1} >) + t_{test} <= \delta$
- For each $S_i, i = 0, ..., k-1$ there exists a path $< P_{i_0}, ..., P_{i_m} >$ such that
  - $P_{i_j} \; \varepsilon \; V$ for $j = 0, ..., m$
  - $(P_{i_j}, P_{i_{j+1}}) \; \varepsilon \; E$ for $j = 0, ..., m-1$
  - $\{R_0, .., R_n\} \cap \{P_{i_1}, .., P_{i_{m-1}}\} = \phi.$
  - $P_{i_0} = S_i$
  - $P_{i_m} \; \varepsilon \; \{R_{j+1}, ..., R_n\}$ where $R_j = S_{i+1}$
- k is minimum.

where RTT is the round-trip time and t_{test} is the periodicity of liveness messages which is same for all links.

7.4.3 Approach

Our goal is to segment the primary path into segments meeting the switch over time constraints while also reserving the backup paths.

This algorithm is a modification of the previous algorithm in which the process of finding the alternate routes is combined with the process of segmenting the primary path. We follow an adaptive process of segmenting the primary path in which the segment size may not be the largest possible according to the constraints but may actually be shorter , to accommodate the formation of alternate paths for protecting that segment.

To segment the entire path, we start from the egress router and find the largest possible segment towards the ingress router which will meet the switch over time constraint. Let the SSR of this segment be S_i corresponding to Router R_i. Then we try to find a backup path from Si which will protect this path. Note that this backup path needs to be disjoint from this segment. If we are able to establish a backup path then we continue otherwise we change the segment size and try to find out the backup path again. Using this approach we segment the entire primary path.

7.4.4 Algorithm

$SegmentAlgo(< R_0..R_n >, \delta, RTT, N)$
$\{$
$\quad segment \leftarrow 0;$
$\quad last \leftarrow n;$
$\quad while(last > 0)do$
$\quad \{$
$\quad\quad S_{segment} \leftarrow R_{last};$
$\quad\quad segment \leftarrow segment + 1;$
$\quad\quad identify\_next\_segment(first, n, last, \delta, RTT);$
$\quad\quad while(!find\_backup\_path(N, first, last, < P_{segment_0}, ..., P_{segment_m} >)and(first! = last))$
$\quad\quad \{$
$\quad\quad\quad first \leftarrow first + 1;$
$\quad\quad \}$
$\quad\quad if(first = last)$
$\quad\quad \{$
$\quad\quad\quad print($"Unable to make backup paths"$);$
$\quad\quad\quad exit(0);$
$\quad\quad \}$
$\quad\quad last \leftarrow first;$
$\quad \}$
$\quad S_{segment} \leftarrow R_0;$
$\}$

$identify\_next\_segment(first, n, last, \delta, RTT)$
$\{$
$\quad first \leftarrow last - 1;$
$\quad RTT \leftarrow t_{test} + RTT(R_{first}, R_{last});$
$\quad next \leftarrow first - 1;$
$\quad while(first >= 0)and(RTT + RTT(R_{next}, R_{first}) <= \delta)do$
$\quad \{$
$\quad\quad RTT \leftarrow RTT + RTT(next, first);$
$\quad\quad first \leftarrow next;$
$\quad\quad next \leftarrow next - 1;$
$\quad \}$
$\}$

$find\_backup\_path(N, first, last, < P_{segment_0}, ..., P_{segment_m} >)$
$\{$
$\quad N.V \leftarrow N.V - (\{R_0, ..., R_{last}\} - \{R_{first}\});$
$\quad N.V \leftarrow N.V \cup \{a\};$
$\quad N.E \leftarrow N.E \cup \{(v, a) : v \epsilon \{R_{last+1}, ..., R_n\}\};$
$\quad$ Find a path $< T_0, ..., T_{m+1} >$ from R_{first} to R_n and assign $P_{segment_i} \leftarrow T_i;$

If(no path exists from R_{first} to R_n)
    return $false$;
else
    return $true$;
}

Note: The path can be found using a simple DFS or a BFS algorithm.

7.4.5 Proofs

Minimum Number of Segments

Proof Statement: The above algorithm finds minimum number of segments.

The above proof follows exactly the same lines as the proof given in Section 7.3.7

7.5 End-To-End Delay

7.5.1 Introduction

End-to-End delay indicates the time a packet takes from the ingress to reach the egress. It is an important multimedia parameter and can be especially important in two-way communication. For example , in a voice conversation, it determines the time delay between the the time a person starts speaking and the other person starts hearing.

The following figure indicates how do the users perceive the effect of end-to-end delay.

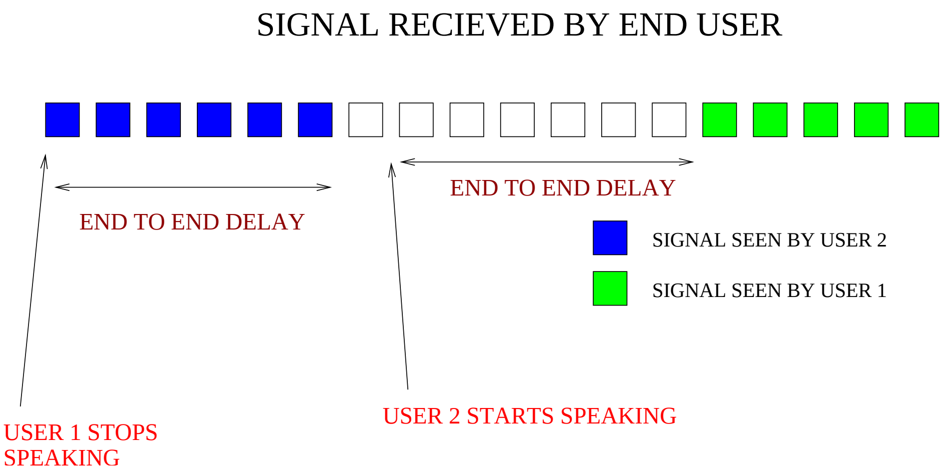


Figure 7.6: Perceived effect of End-to-End delay

7.5.2 Problem Statement

Given a Network N(V,E), a LSP $< R_0, ..., R_i, ..., R_n >$, an upper bound δ on switch-over time, and an upper bound η on end-to-end delay, identify k segments $\{< S_i, S_{i+1} >, i = 0, ..., k-1\}$ and backup paths $\{< P_{i_0}, ..., P_{i_m} >, i = 0, ..., k-1\}$, such that

- $S_0 = R_0$
- $S_k = R_n$
- $RTT(< S_i, S_{i+1} >) + t_{test} <= \delta$
- For each $S_i, i = 0, ..., k-1$ there exists a path $< P_{i_0}, ..., P_{i_m} >$ such that

- $P_{i_j} \ \varepsilon \ V$ for $j = 0, ..., m$
- $(P_{i_j}, P_{i_{j+1}}) \ \varepsilon \ E$ for $j = 0, ..., m-1$
- $\{R_0, .., R_n\} \cap \{P_{i_1}, .., P_{i_{m-1}}\} = \phi$.
- $P_{i_0} = S_i$
- $P_{i_m} \ \varepsilon \ \{R_{j+1}, ..., R_n\}$ where $R_j = S_{i+1}$

- $OWD(< R_0, .., S_i >) + OWD(< P_{i_0}, .., P_{i_m}) + OWD(< P_{i_m}, ..., R_n >) <= \eta$ for $i = 0, ..., k-1$
- k is minimum.

where RTT is the round-trip time, OWD is one-way delay and t_{test} is the periodicity of liveness messages which is same for all links.

7.5.3 Approach

To include end-to-end delay we need to consider the delay of each of the individual links along a route from the ingress to the egress. For the primary LSP , the end-to-end delay will simply be the sum of the delays of each of the individual links.

However, when we provide protection using alternate routes, we need to make sure that the new path thus formed (in case of a failure) also meets the end-to-end delay constraints. To achieve this we need to take the delay characteristics of the alternate route into consideration.

While setting up a backup path for a segment , we need to make sure that in case that particular segment fails, the end-to-end delay characteristics of the new route from the ingress to the egress which includes this protection path, meets the specified constraints. We need to ensure this for all segments and all backup paths established.

Specifically, if the end-to-end to delay of the protection path is $T2$ and the end-to-end delay of the portion of the primary LSP which the protection path covers is $T1$, then the difference $T2 - T1$ is the additional end-to-end delay incurred by the packets. So we need to make sure that the Primary LSP $end\!-\!to\!-\!enddelay + T2 - T1$ is less than the required end-to-end delay bound for each backup path.

7.5.4 Algorithm

$SegmentAlgo(< R_0..R_n >, \delta, RTT, N)$
{
  $segment \leftarrow 0;$
  $last \leftarrow n;$
  $while(last > 0)do$
  {
    $S_{segment} \leftarrow R_{last};$

```
        segment ← segment + 1;
        identify_next_segment(first, n, last, δ, RTT);
        while(!find_backup_path(N, first, last, < P_segment_0, ..., P_segment_m >)and(first! =
last))
        {
            first ← first + 1;
        }
        if(first = last)
        {
            print("Unable to make backup paths");
            exit(0);
        }
        last ← first;
    }
    S_segment ← R_0;
}

identify_next_segment(first, n, last, δ, RTT)
{
    first ← last − 1;
    RTT ← t_test + RTT(R_first, R_last);
    next ← first − 1;
    while(first >= 0)and(RTT + RTT(R_next, R_first) <= δ)do
    {
        RTT ← RTT + RTT(next, first);
        first ← next;
        next ← next − 1;
    }
}

find_backup_path(N, first, last, < P_segment_0, ..., P_segment_m >)
{
    N.V ← N.V − ({R_0, ..., R_last} − {R_first});
    N.V ← N.V ∪ {a};
    N.E ← N.E ∪ {(v, a) : vϵ{R_last+1, ..., R_n}};
    Find the shortest path < T_0, ..., T_m+1 > from R_first to R_n and assign P_segment_i ←
T_i;
    If(no path exists from R_first to R_n)
        return false;
    else
        return true;
}
```

7.5.5 Proofs

Minimum Number of Segments

Proof Statement: The above algorithm finds minimum number of segments.

The above proof follows exactly the same lines as the proof given in Section 7.3.7

7.6 Jitter

7.6.1 Introduction

Jitter (the deviation from the scheduled arrival time of data packets) is an important QoS metric which affects multimedia communications. When the packets are being rerouted through the backup path, we need to make sure that the backup path doesn't introduce extra jitter and violate the Jitter constraints.

Jitter is also a link property like delay, and can be modeled in exactly in the same way as end-to-end delay.

7.7 Dynamic Programming Approach to Multi-Constraint Satisfaction

7.7.1 Introduction

In this section we consider an algorithm for providing a protection configuration for a scenario where there are multiple QoS constraints required to be met. So while setting up the backup paths, we need to make sure that in case the packets are switched over to a backup path, no QoS constraint is violated.

For doing this , we consider a dynamic programming approach. Since while searching for the backup path , we need to find a path which satisfies two constraints than one, a dynamic programming approach is needed to do do.

7.7.2 Problem Statement

Given a Network N(V,E), a LSP $< R_0, ..., R_i, ..., R_n >$, an upper bound δ on switch-over time, an upper bound η on end-to-end delay and an upper bound Ω on jitter, identify k segments $\{< S_i, S_{i+1} >, i = 0, ..., k-1\}$ and backup paths $\{< P_{i_0}, ..., P_{i_m} >, i = 0, ..., k-1\}$, such that

- $S_0 = R_0$
- $S_k = R_n$
- $RTT(< S_i, S_{i+1} >) + t_{test} <= \delta$
- For each $S_i, i = 0, ..., k-1$ there exists a path $< P_{i_0}, ..., P_{i_m} >$ such that
  - $P_{i_j} \ \varepsilon \ V$ for $j = 0, ..., m$
  - $(P_{i_j}, P_{i_{j+1}}) \ \varepsilon \ E$ for $j = 0, ..., m-1$
  - $\{R_0, .., R_n\} \cap \{P_{i_1}, .., P_{i_{m-1}}\} = \phi$.
  - $P_{i_0} = S_i$
  - $P_{i_m} \ \varepsilon \ \{R_{j+1}, ..., R_n\}$ where $R_j = S_{i+1}$
- $OWD(< R_0, .., S_i >) + OWD(< P_{i_0}, .., P_{i_m}) + OWD(< P_{i_m}, ..., R_n >) <= \eta$ for $i = 0, ..., k-1$
- $Jitter(< R_0, .., S_i >) + Jitter(< P_{i_0}, .., P_{i_m}) + Jitter(< P_{i_m}, ..., R_n >) <= \Omega$ for $i = 0, ..., k-1$
- k is minimum.

where RTT is the round-trip time and t_{test} is the periodicity of liveness messages which is same for all links.

7.7.3 Approach

The algorithm for the above problem is very similar to the algorithm given in the previous section. The technique to find the segment head is exactly the same. The only change is in the way in which we find the backup paths as there is an additional constraint to be satisfied. This problem lies in the well known category of finding shortest path with constraint satisfaction. A dynamic programming approach can be used for such a purpose.

7.7.4 Algorithm

$SegmentAlgo(< R_0..R_n >, \delta, RTT, N)$
{
    $segment \leftarrow 0;$
    $last \leftarrow n;$
    $while(last > 0)do$
    {
        $S_{segment} \leftarrow R_{last};$
        $segment \leftarrow segment + 1;$
        $identify\_next\_segment(first, n, last, \delta, RTT);$
        $while(!find\_backup\_path(N, first, last, < P_{segment_0}, ..., P_{segment_m} >)and(first! = last))$
        {
            $first \leftarrow first + 1;$
        }
        $if(first = last)$
        {
            $print$("Unable to make backup paths");
            $exit(0);$
        }
        $last \leftarrow first;$
    }
    $S_{segment} \leftarrow R_0;$
}

$identify\_next\_segment(first, n, last, \delta, RTT)$
{
    $first \leftarrow last - 1;$
    $RTT \leftarrow t_{test} + RTT(R_{first}, R_{last});$
    $next \leftarrow first - 1;$
    $while(first >= 0)and(RTT + RTT(R_{next}, R_{first}) <= \delta)do$
    {
        $RTT \leftarrow RTT + RTT(next, first);$
        $first \leftarrow next;$
        $next \leftarrow next - 1;$

}
}

$find\_backup\_path(N, first, last, < P_{segment_0}, ..., P_{segment_m} >)$
{
$N.V \leftarrow N.V - (\{R_0, ..., R_{last}\} - \{R_{first}\});$
$N.V \leftarrow N.V \cup \{a\};$
$N.E \leftarrow N.E \cup \{(v, a) : v\epsilon\{R_{last+1}, ..., R_n\}\};$
$< P_{segment_0}, ..., P_{segment_m} >=$
$FindPath(<>, R_{first}, R_n, \eta - RTT(< R_0, R_{first} >), \Omega - Jitter(< R_0, R_{first} >))$
If(no such path exists from R_{first} to R_n)
return $false$;
else
return $true$;
}

$FindPath(< P_0, .., P_k >, R_i, R_j, \eta, \Omega)$
{ $if(i == j)$
$return < P_0, .., P_k >;$
$else$
$return \quad MIN\{FindPath(< P_0, .., P_k, R_i >, R_k, R_j, \eta - Delay(R_i, R_k), \Omega - Jitter(R_i, R_k))$
for all edges (R_i, R_k)
such that $\eta - Delay(R_i, R_k) > 0, \Omega - Jitter(R_i, R_k) > 0.\}$
}

7.7.5 Proofs

Minimum Number of Segments

Proof Statement: The above algorithm finds minimum number of segments.

The above proof follows exactly the same lines as the proof given in Section 7.3.7

7.8 Reliability

7.8.1 Introduction

Reliability is an important parameter in computer networks. Providing backup path helps in improving the reliability of the path which is setup between the ingress and the egress nodes. For Example, consider two nodes with primary path reliability p_e.

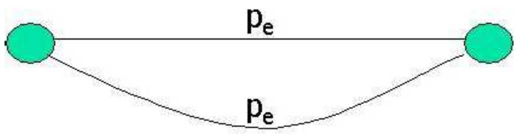


Figure 7.7: Path between 2 nodes

If we provide an additional backup path (as shown in the figure) with reliability p_e, the reliability of the path becomes $2p_e - p_e^2$.

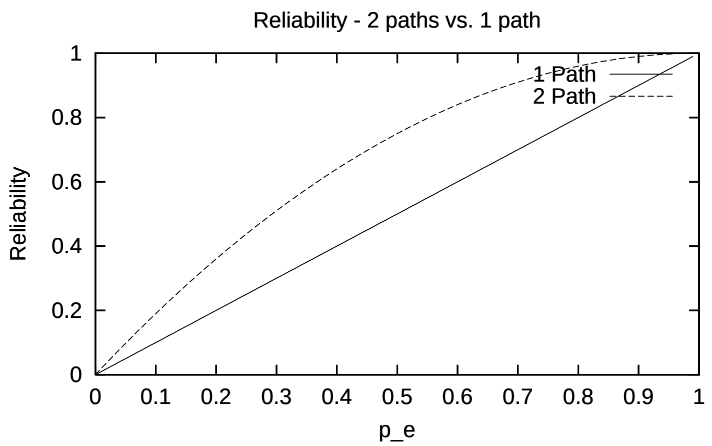


Figure 7.8: Reliability for one path vs. two paths

As seen from the above graph, path protection helps in improving reliability. In our approach, for providing path-protection, the size of a segment effects the reliability of the path formed between an ingress and egress routers.

A small theoretical analysis with a few assumptions can show that smaller segments result in higher reliability.

Segment Size vs. Reliability

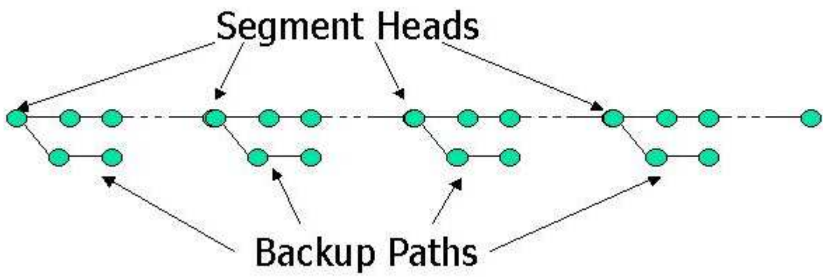


Figure 7.9: Primary and Backup Paths

Consider a network N(V,E) with link reliability p for each link. A primary path with n links is setup and the size of each segment be k. Therefore, the number of segments is $\frac{n}{k}$. Assume the size of the backup path for each segment be the same as the size of the segment, i.e. k. Also assume that there is no sharing of backup paths.

Since there are k links in each segment and its backup path the reliability of a segment is $2p^k - p^{2k}$. Since there are $\frac{n}{k}$ segments, the reliability of the path is $(2p^k - p^{2k})^{\frac{n}{k}}$. This value is plotted with the segment size in the graph below.

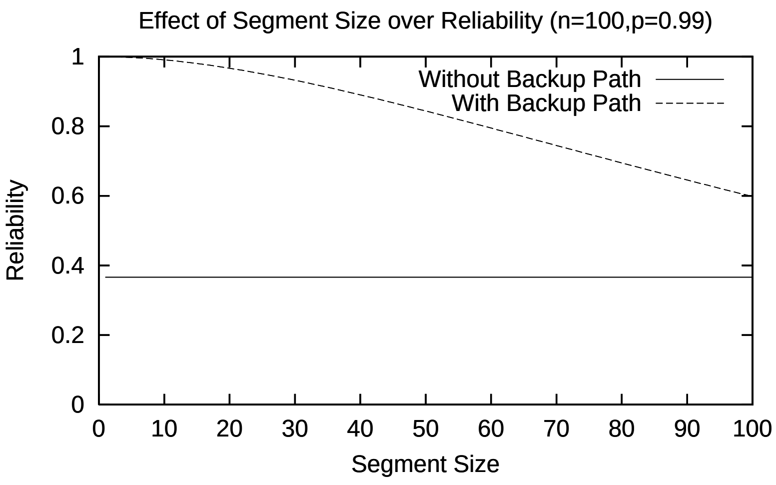


Figure 7.10: Reliability vs. Segment Size

As seen from the above graph, the reliability is much higher when we are providing path protection. Also with the increase in segment size the reliability decreases.

Calculating Reliability

The reliability of a computer network can often be modeled by a probabilistic graph $G = (V, E)$ with perfectly reliable nodes, and imperfect edges, where V is the set of n nodes and E is the set of m edges representing bidirectional communication links. We denote the reliability of the corresponding graph between the ingress and egress node by R(G). Most of the proposed approaches come from one of the following reliability formula:

- **Subgraph Counting :**

  $$R(G) = \sum_{i=0}^{m} N_i(G) q^i (1-q)^{m-i}$$

  where $N_i(G)$ is the number of subgraphs of G containing i edges with a path from ingress node to the egress node, and q is the reliability of an edge.

- **Factoring :** For an edge e in graph G

  $$R(G) = qR(G.e) + (1-q)R(G-e)$$

  where G.e is the graph G contracting edge e related to G with edge e working, while $G - e$ is G removing e related to G with edge e failed.

Any of these two definitions is suitable for getting the accurate expression for reliability. However, since these are exponential time algorithms, we use bounding expressions to estimate reliability.

Graph Transformations

It is often useful to perform a few graph transformations before using the above formulas, which helps in reducing the size of the graph and also helps dealing with node failures.

Dealing with Node Failure

An unreliable node A with reliability p_n can be replaced by two reliable nodes A_1 and A_2 with the link (from A_1 to A_2) reliability p_n. All incoming links to A are directed to A_1 and all outgoing links from A can be directed out from A_2. To find reliability between 2 nodes A and B in the original graph, we find the reliability between A_1 and B_2 in the transformed graph.

Merging

These transformations can be used to reduce the size of the graph.

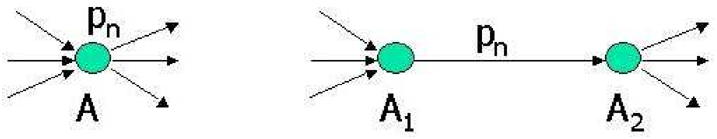


Figure 7.11: Graph Transformation

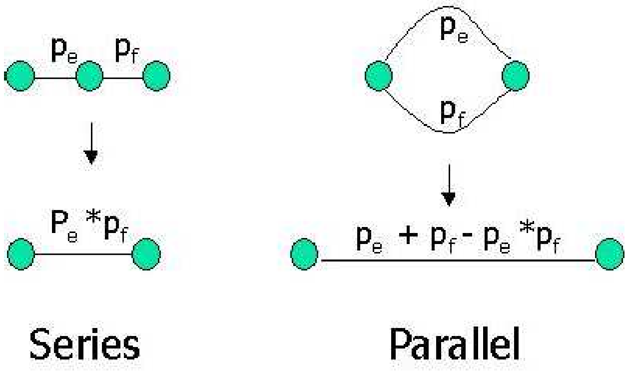


Figure 7.12: Graph Transformation

Algorithm to compute reliability

Even though the 2-node reliability problem is NP complete, an important observation is that the definition of existence of path is very different in our case. This can be explained using a diagram below.

As can be seen from the above figure, theoretically a path exists between the ingress

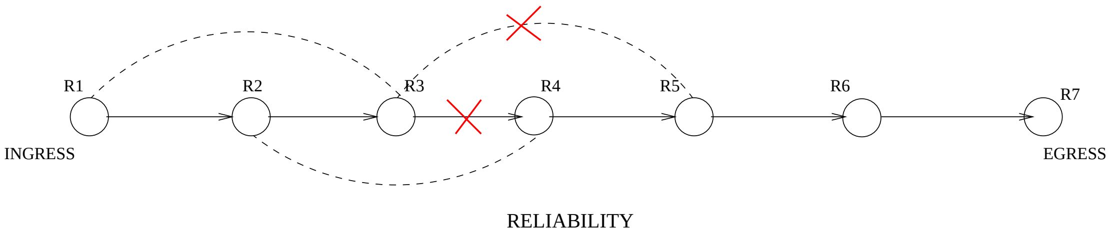


Figure 7.13: Difference in definition of path existence

node and the egress node, but according to our definition path does not exist as on the failure of link from R_3 to R_4, the packets can not be rerouted through SSR R_3 as the backup path has also failed. This observation was used for finding the exact value for reliability. The main concept is that for the LSP to work, either the primary path or the backup path should be working.

Let there be k segments, denoted by $< S_1, .., S_k >$. where a segment consists of those links that are protected by a single backup up.

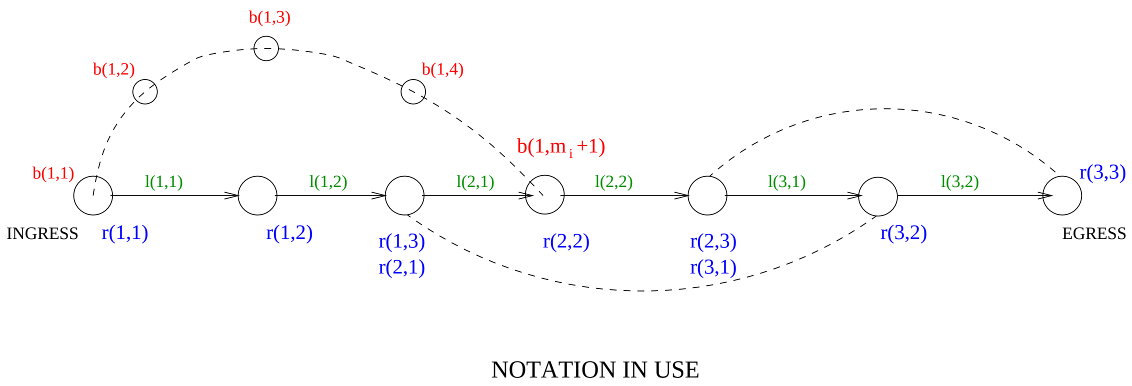


Figure 7.14: Notations used

Let the segment S_i contain n_i links, $< L_{i,1}, .., L_{i,n_i} >$ where $L_{i,j}$ is a link between routers $R_{i,j}$ and router $R_{i,j+1}$. Note that $R_{i,n_i+1} = R_{i+1,1}$.

Let $B_{i,1}, .., B_{i,m_i+1}$ be the backup path for segment S_i, where

- $B_{i,1} = R_{i,1}$
- $B_{i,m_i+1} = R_{x,j}$ where either $x > i + 1$ or $x = i + 1$ and $j > 1$. Let $Dest(i)$ be defined as the segment number where the backup path from segment S_i meets the primary path i.e. $Dest(i) = x$.

Let $BL_{i,j}$ is a link between routers $B_{i,j}$ and router $B_{i,j+1}$.

Let the reliability of any link $L_{i,j}$ be denoted as $p_{i,j}$. Therefore the probability of no links in segment S_i has failed is

$$P_i = \prod_{a=1}^{a=n_i} p_{i,a} \tag{7.1}$$

Let the reliability of any link $BL_{i,j}$ be denoted as $q_{i,j}$. Therefore the probability that the backup path for segment S_i can be used is

$$Q_i = \left(\prod_{a=1}^{a=m_i} q_{i,a} \right) \times \left(\prod_{a=j}^{a=n_{Dest(i)}} p_{Dest(i),a} \right) \tag{7.2}$$

Let $G_{i,j}$ be the probability that there is no failure in the primary path in any segment from S_i to S_{j-1} and there is a failure in segment S_j.

$$G_{i,j} = \left(\prod_{a=i}^{a=j-1} P_a \right) \times (1 - P_j) \tag{7.3}$$

Let $Rel(i)$ be the reliability of path starting from segment i. We want to find the value for $Rel(0)$. We can define $Rel(i)$ as:

$$Rel(k+1) = 1 \tag{7.4}$$

$$Rel(i) = \sum_{a=i}^{a=k} (G_{i,a} \times Q_a \times Rel(Dest(i) + 1)) + \prod_{a=i}^{a=k} P_a \tag{7.5}$$

The number of operations required for computing reliability using the above formula is $O(\sum_{i=1}^{i=k} n_i + \sum_{i=1}^{i=k} m_i + k^2)$ i.e. linear with respect to total number of edges and square with respect to number of segments.

7.8.2 Problem Statement

Given a Network N(V,E), a LSP $< R_0, ..., R_i, ..., R_n >$, an upper bound δ on switch-over time, and minimum reliability requirement r, identify k segments $\{< S_i, S_{i+1} >, i = 0, ..., k-1\}$ and backup paths $\{< P_{i_0}, ..., P_{i_m} >, i = 0, ..., k-1\}$, such that

- $S_0 = R_0$
- $S_k = R_n$
- $RTT(< S_i, S_{i+1} >) + t_{test} <= \delta$
- For each $S_i, i = 0, ..., k-1$ there exists a path $< P_{i_0}, ..., P_{i_m} >$ such that
  - $P_{i_j} \ \varepsilon \ V$ for $j = 0, ..., m$
  - $(P_{i_j}, P_{i_{j+1}}) \ \varepsilon \ E$ for $j = 0, ..., m-1$
  - $\{R_0, .., R_n\} \cap \{P_{i_1}, .., P_{i_{m-1}}\} = \phi$.
  - $P_{i_0} = S_i$
  - $P_{i_m} \ \varepsilon \ \{R_{j+1}, ..., R_n\}$ where $R_j = S_{i+1}$
- Minimum path reliability is r
- k is minimum.

where RTT is the round-trip time and t_{test} is the periodicity of liveness messages which is same for all links.

7.8.3 Approach

Using the formula for the reliability, it is possible to find most reliable set of backup paths for minimum number of segments. However given a bound on reliability, it is not possible to find the most optimal solution with respect to the number of segments. This is because the reliability of the path increases with increase in the number of segments. However we don't know before hand the size of the segment which will be required for obtaining a solution with reliability constraint satisfied. Therefore we find the most optimal solution with respect to number of segments, with maximum reliability and then if it does not satisfy the reliability constraint we use heuristic to divide a segment into two to improve reliability.

Chapter 8

Experimental Results

In this section we present simulation results which show the advantages of the segment based approach. For this we implemented a simulator in C++ which reads or generates network topologies and then simulates random LSP requests to the network with specified bandwidth requirements and bound on switch over time and reserves primary and backup path for the LSP on the Network.

An LSP request is generated by specifying an ingress router and the egress router, amount of bandwidth to be reserved along with the QoS constraints like bounded switch over time in case of failure. The primary LSP is setup using Djiktra's Shortest Path Algorithm between the ingress and the egress. All the edges which don't satisfy the bandwidth requirement for the primary path are not included while searching for the shortest path.

We then study the effect of using the segment based approach to setup up protection paths on the amount of resources used by these. We compare it the Local Path Protection scheme which can be modeled using the segment based approach with segment size of one.

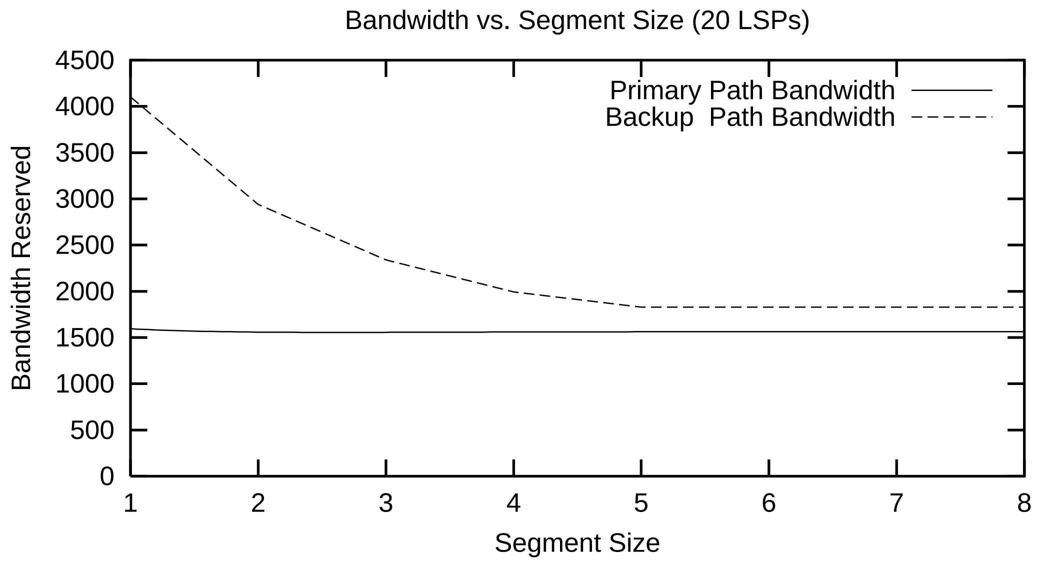
Then we also study the effect of sharing on the protection resources. As mentioned earlier, since we are assuming only one fault in the network, we can conserve more resources by sharing the backup paths among the primary paths.

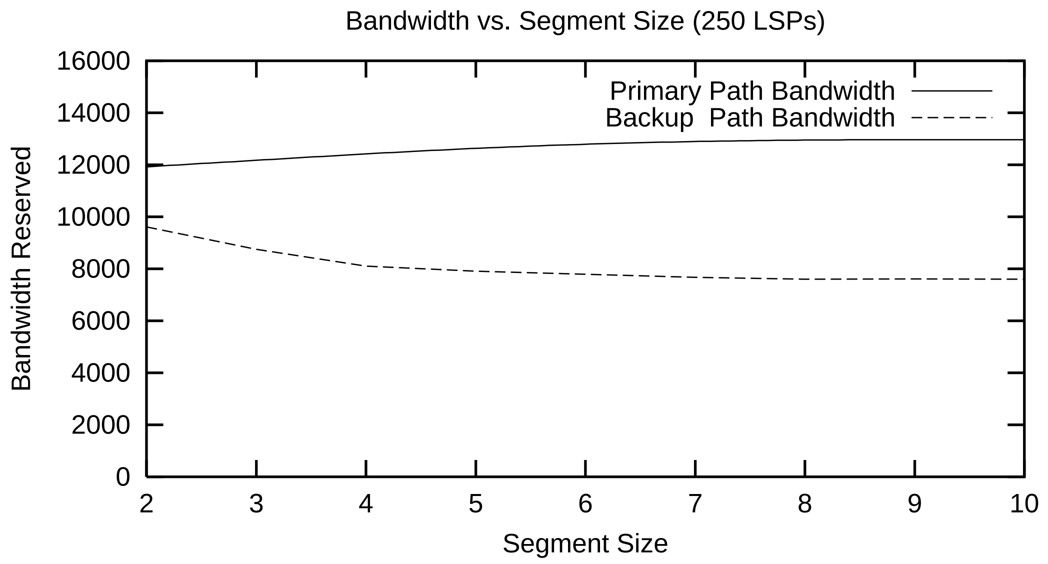
8.1 Experimental Setup

We performed experiments on a graph with 100 nodes and 200 links. All links were assumed to be symmetric. The capacity of the link was assigned randomly between 50 and 100 units(uniform distribution). The round trip delay was set to 10 ms. The periodicity of the liveness message T_{test} was set to 2 ms.

By fixing the round trip delay of links to 10ms. and periodicity to 2ms we can divide the primary LSP into segment of s nodes each by specifying the maximum allowed switch over time to 10*s+2 ms. for every LSP. This observation was used to setup the LSP with the required segment size.

8.2 Variation of Primary and Backup Path Bandwidth with the change in segment size





As expected and can be seen from the graphs the segment size has no effect on the Primary Path Bandwidth. But as the segment size increase the total Backup Path Bandwidth required decreases. This is because as the segment size increases, the number of segments decreases and therefore number of backup paths required to protect the primary path decrease. This can be seen from the two graphs.

Another observation from the two graphs is the difference in the total bandwidth reserved for the primary and the backup path. This depends on two factors

1. Backup Paths are longer than the Primary Path: Both primary and backup paths are setup using the Djiktra's algorithm. But primary path is setup before the backup path and the two cannot share any link. Therefore the primary path is shorter. As a result the total bandwidth reserved on the network for the primary paths will be less than the backup paths.

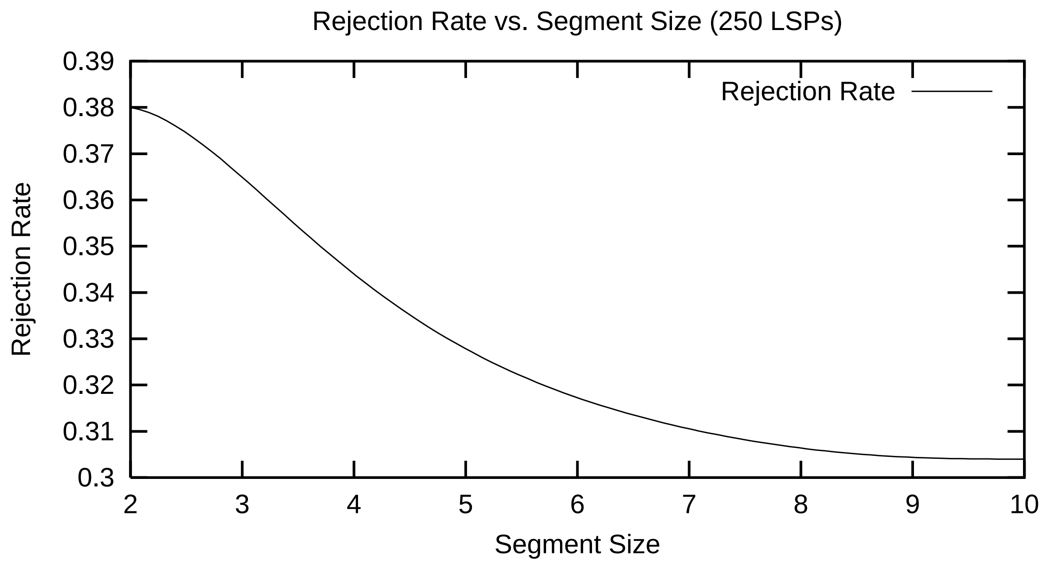
2. Backup paths can share links but Primary Paths can not: The backup bandwidth can be shared between the two links. This results in a decrease in the total backup bandwidth reserved.

Effects of Sharing

For the 20 LSP experiment the bandwidth required for setting up backup path is more than that for primary path. This is because with 20 LSPs on such a big network there will not be much sharing of backup paths and the path required to setup a backup path are longer than primary paths.

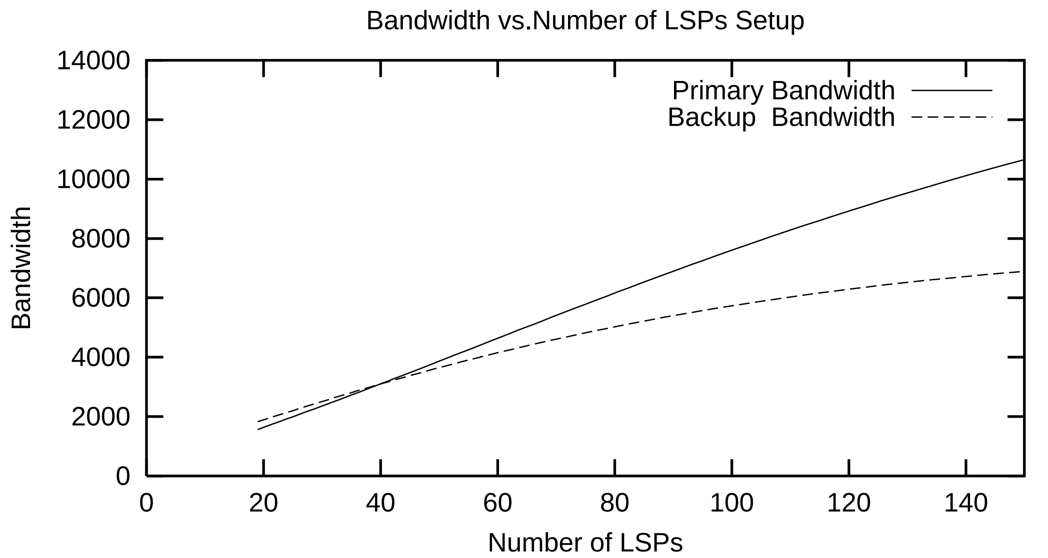
However, for the 250 LSP experiment the bandwidth required for setting up primary path is more than the total backup path reserved. This is because of the increased amount of sharing of backup links, which is possible. Even though the backup paths are longer the sharing effect dominates.

8.3 Change in Rejection Rate with the Segment Size



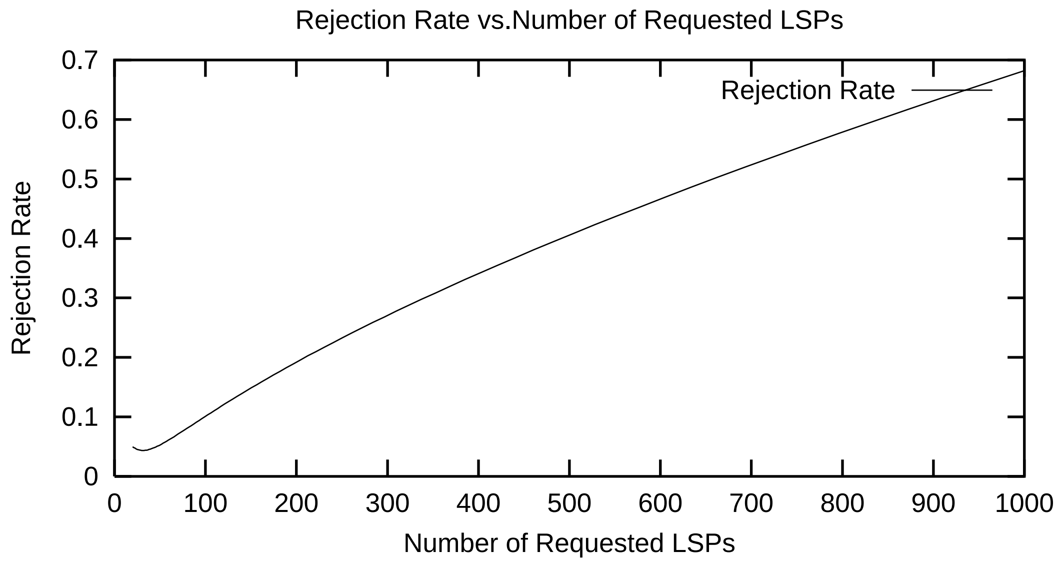
As the segment size increases the total bandwidth reserved (Primary and Backup) reduces. This reflects in the rejection rate. As can be seen from the above graph the rejection rate for segments of size 2 is 0.38 whereas the rejection rate is 0.3 for LSPs setup with segment size 10. Thus more LSP requests can be accommodated using the segment based approach.

8.4 Variation in Bandwidth Reserved with the Number of LSPs



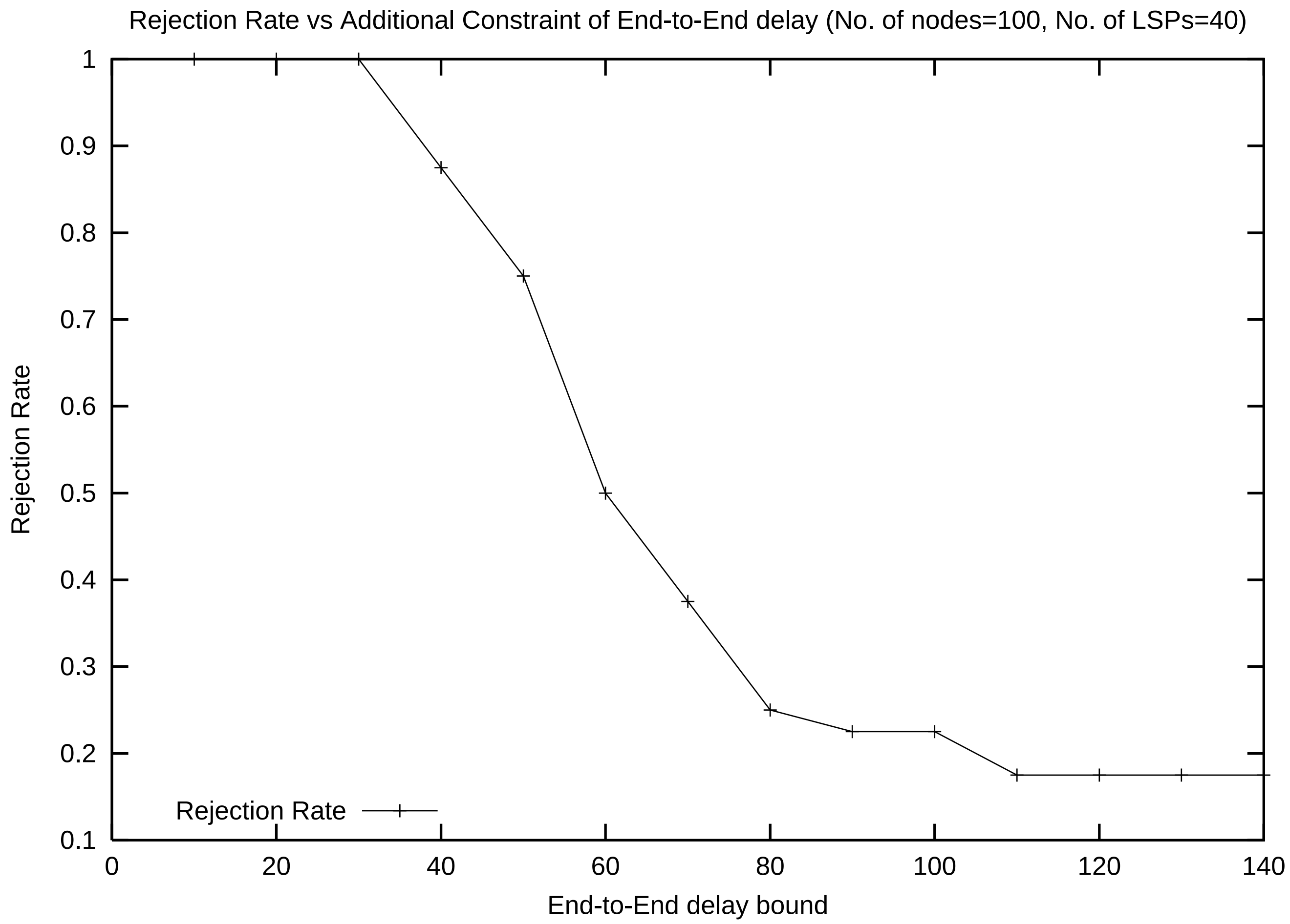
As expected with the increase in the number of LSPs the bandwidth reserved increases. The crossover in the graph indicates the effect of the 2 factors stated in section 6.2. When the number of LSPs is less than 40 Factor 1 dominates and the backup path reserves more bandwidth. As the number of LSP setup becomes more than 40 sharing will increase and the factor 2 starts to dominate.

8.5 Change in Rejection Rate with the Number of LSPs



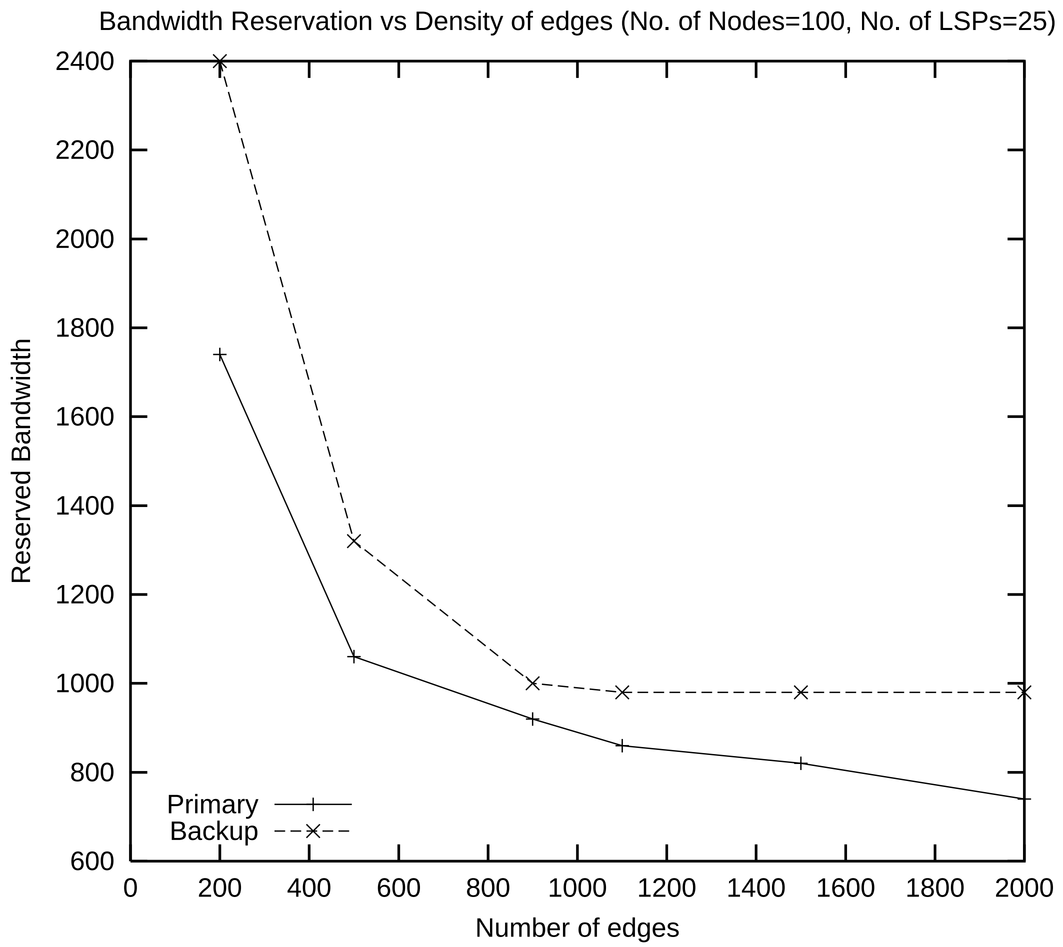
With the increase in the number of LSPs the bandwidth reserved on the network increases and more and more LSP request will be rejected because of the inability to setup primary path or the backup path for the request.

8.6 Effect of multiple constraints: End-to-End delay and Switch over time



If we also consider the QoS constraint of end-to-end delay along with bounded switch over time, we see that stringent end-to-end delay increases the rejection rate as it may not be possible to satisfy the additional constraint even if the bounded switch over time constraint can be satisfied. As we increase the permissible end-to-end delay, the rejection rate falls.

8.7 Effect of Density of the graph on the bandwidth reserved



As the density of the graph increases, we note that the primary paths as well as the backup paths become shorter. Thus the amount of bandwidth reserved by these also reduces.

Chapter 9

Visualization

In addition to development of algorithms and a simulator to simulate the segment-based algorithms, a visualization system was also developed as a part of the project.

Visualization shows the functioning of the Adaptive segment based bounded switch over time algorithm on a network modeled as a graph. It shows the various steps taken by the algorithm with the passage of time, with various visual cues such as changing colors of elements (nodes and links), zoom in/out, flashing and so on.

9.1 Advantages of Visualization

Some of the advantages offered by visualization of the algorithms are:

1. It aids in understanding how the algorithm actually works on a network. Using visual information, one can more easily assimilate the various steps of the algorithm, and their appropriate sequence etc.

2. The visualization is not static but of dynamic nature i.e. it is closely coupled with the simulator and visualization actually reflects all the steps of the simulation. As the simulation changes according to the input parameters, the output visualization will also change and show what happened in that particular simulation. Therefore one can visualize various situations by changing the input to the simulator.

3. It also helps in establishing in correctness of the algorithms, which are visualized. Since the visualization is dynamic and not static in nature and reflects the working of the algorithm, one can verify the various steps being taken by the algorithm.

Types of Visualization implemented

We have implemented two aspects of the visualization:

1. Visualization of the Adaptive Bounded Switch Over Time Algorithm.

Here, we show all the steps as listed in the algorithm previously, using visual cues. Thus one can clearly understand the various steps.

2. Visualization of Segment Switch Router switching the traffic on to the appropriate backup path in case of a failure on the primary path.

9.2 Features of Visualization

- It shows the actual graph, which is being used by the simulator and then shows the steps taken by the algorithm on that particular graph.
- The animation is not static in nature and it is completely configurable by the user. The user can specify all the required parameters like the choice of the ingress node and the egress node, the bound on switch over delay, the node which is to fail for rerouting visualization and so on. Therefore the user can try out various test cases and study them separately using the visualization system.
- It uses many visual cues using color codes, blinking of nodes and links to show a specific path found, zooming in to specific nodes of interest (for example when the Segment Switch Router is established by the algorithm), showing failures using node explosion effects.
- Shows actual flow of packets and switching on to the backup path in case of failure on the primary path using packet flow animation.
- A running commentary is displayed in the side of the animation, describing each step as it is happening.
- The tools used by the visualization are available on multi-platform, so visualization is executable over various platforms like Linux, Windows.

9.3 Implementation Details

The visualization was implemented using the POLKA/SAMBA software visualization system developed by the Software Visualization group at Computer Science Department, Georgia Institute of Technology, Atlanta. We used the SAMBA animation system for developing our visualizations.

SAMBA is a front-end to the POLKA algorithm animation system. Samba is, in essence, an interactive animation interpreter that reads animation commands and performs the corresponding animation actions. The chief advantage of Samba is that a program in any language can be annotated to generate these ASCII commands, thus driving a visualization of the program. This facilitates its interfacing to any language of choice, by outputting the appropriate animation commands at each step in the code and then feeding the commands to SAMBA. We developed our simulator in C++ and later on, after getting familiar with SAMBA, later on modified our simulator to output each step of the simulator in form of SAMBA animation commands.

After the simulator has finished execution, a huge file of SAMBA commands is generated (in our case, it exceeded 16,000 lines of code) which is then fed into the SAMBA animation system, and it runs the commands in form of a visualization.

The following topology was used to generate the visualization.

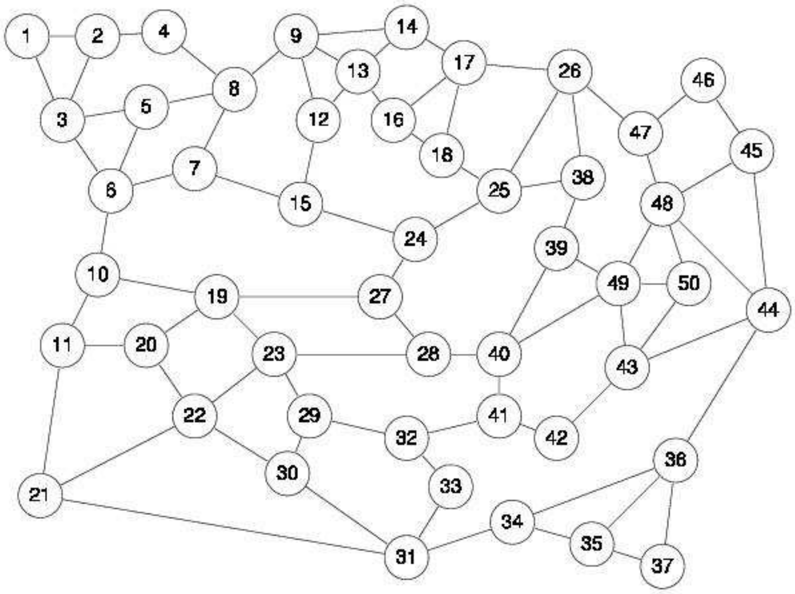


Figure 9.1: Topology used for driving the visualization

User can specify the following parameters for running the simulation and then its visualization:

1. The Ingress node and the egress node

2. Bound on the Switch over time

3. Permissible End-to-End delay

Then the user can also specify the details of the failure scenario by specifying which node should fail after the complete protection resources have been setup. Then the visualization system generates those faults one by one and shows how the network responds to those faults by appropriately switching over the traffic to the protection paths.

9.4 Examples of Visualization

Visualization of Primary Path and Backup Paths [1]

Figure 1 shows the network topology modeled as a graph. A primary path (shown in thick red edges) has been established from the ingress router to the egress router and a backup path has also been established by the adaptive segment based algorithm (shown by the thick white edges). The primary path consisting of the nodes and the edges is displayed in red, while the rest of the nodes not participating in any traffic flow from the ingress to the egress are shown in blue and the edges in green. Comments are also displayed on the right side, which describes each step as it is happening.

Searching for a Backup Path

Figure 2 shows the algorithm in process of searching for a new backup path (the second one in the current scenario) after an appropriate Segment Switch Router has been established. The grayed out edges indicate the edges, which cannot be a part of the backup path. This is necessary since these edges and nodes must be disjoint from the backup, otherwise , a loop can be formed. Right now, the algorithm is looking for the shortest path from the segment switch router to any of the nodes which lie downstream of the current segment. This is achieved by having a artificial node (colored purple here) , and connecting it to all such nodes with edge length 0. Then a shortest path is found from the Segment Switch Router to the artificial node, thus finding a backup path for this particular segment.

Packet flow over primary path and visualization of a node failure

Figure 3 shows the packets flowing on the primary path (packets are represented as blue filled dots). Just now a node on the primary path has failed, shown by the blowing up node in gray. After this a notification will reach the Segment Switch Router responsible for protecting this particular segment and will reroute the traffic to the alternate backup path. Note that the animation and the failure is completely configurable i.e. the user can specify his choice regarding which node or link is to fail before the animation and the appropriate scenario will be visualized.

In Figure 4, the packets have been rerouted over the backup path and are reaching the egress via the backup path. Later on , the recovery of the original primary path is also shown, and then the packets are switched back on to the primary path by the Segment Switch Router. As can be seen, after the recovery some packets are still being transmitted over the backup path and the traffic has now been switched over to the primary LSP.

[1] Please refer to the images shown at the end of the chapter

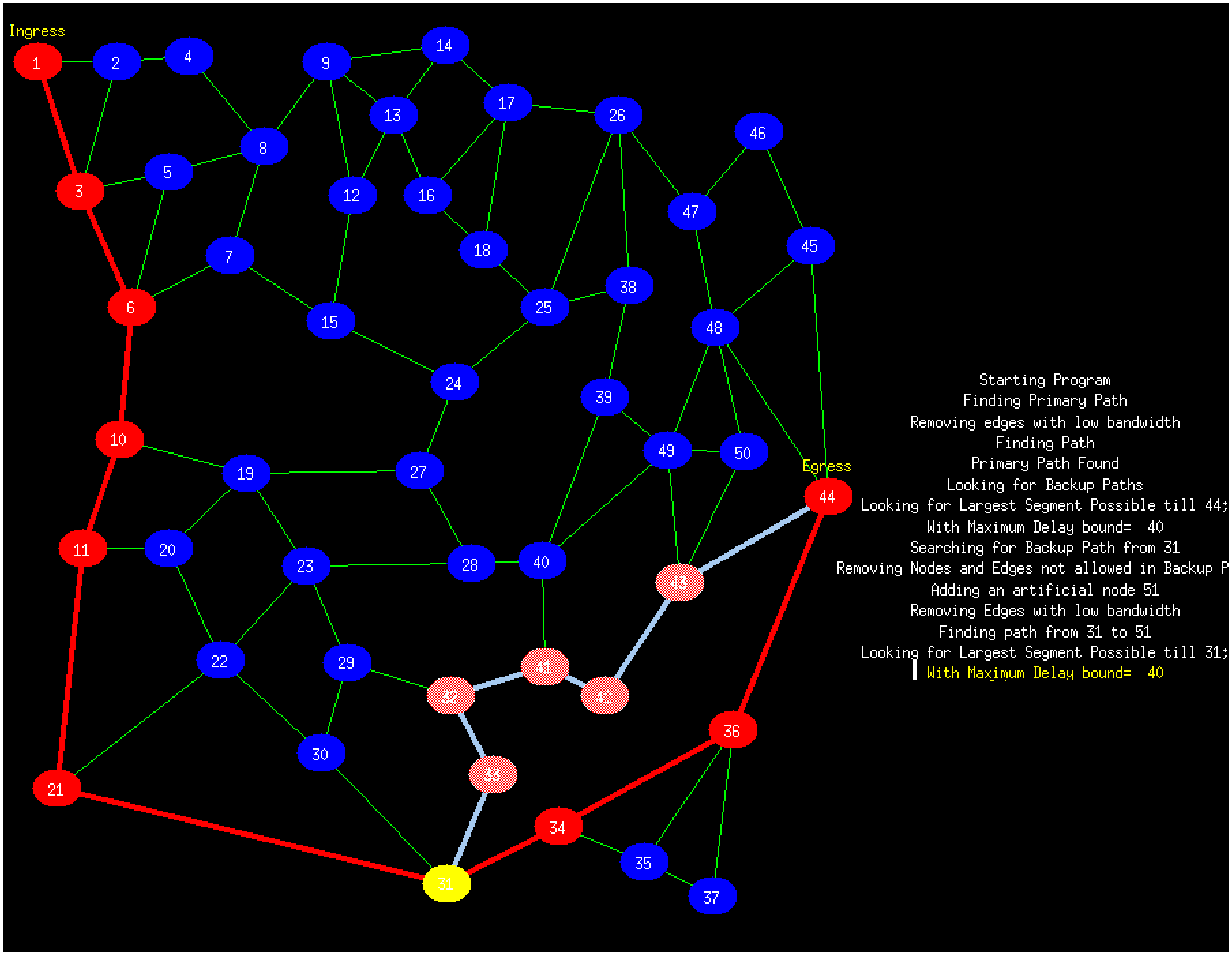


Figure 9.2: Visualization of Primary Path and Backup Paths

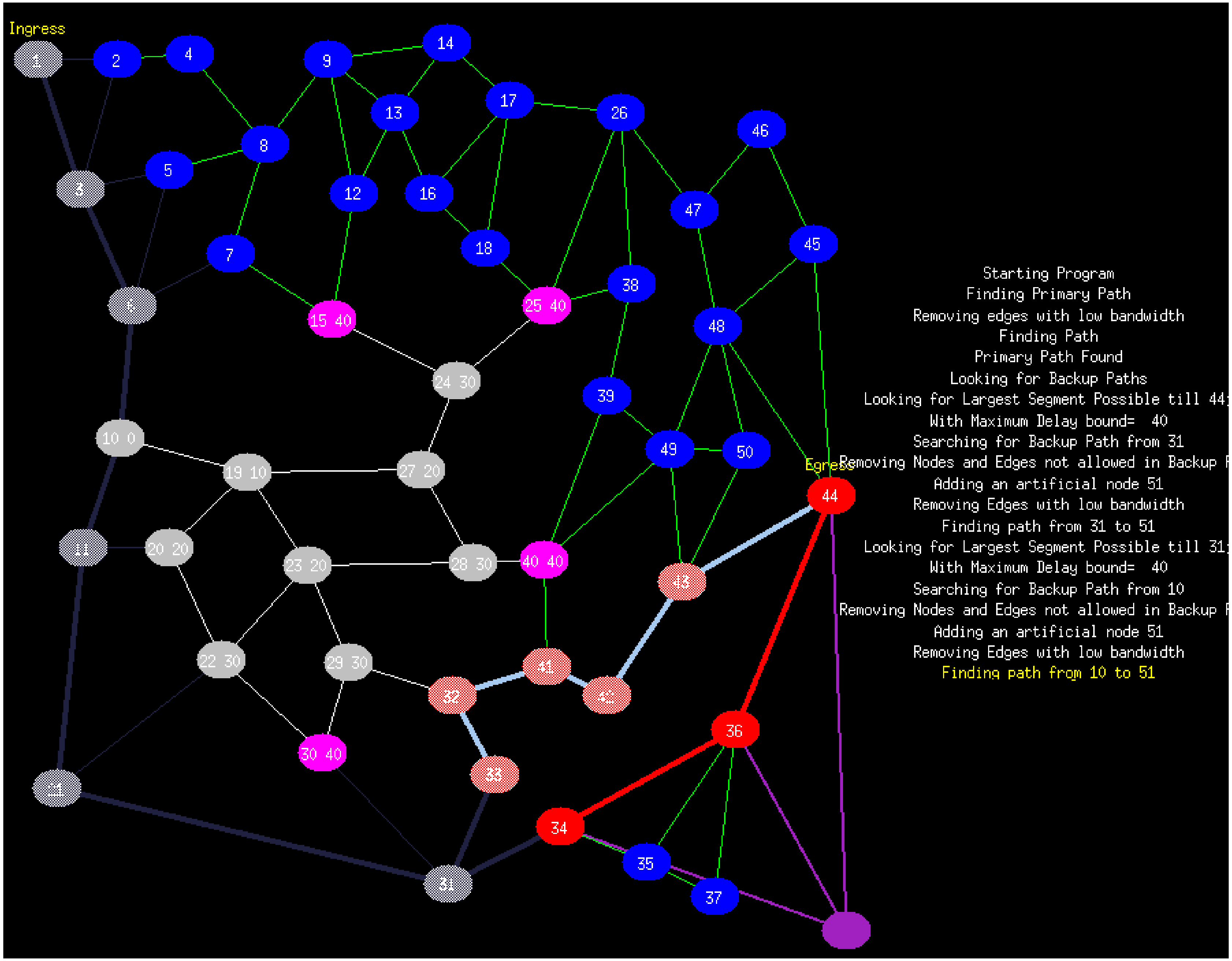


Figure 9.3: Searching for a Backup Path

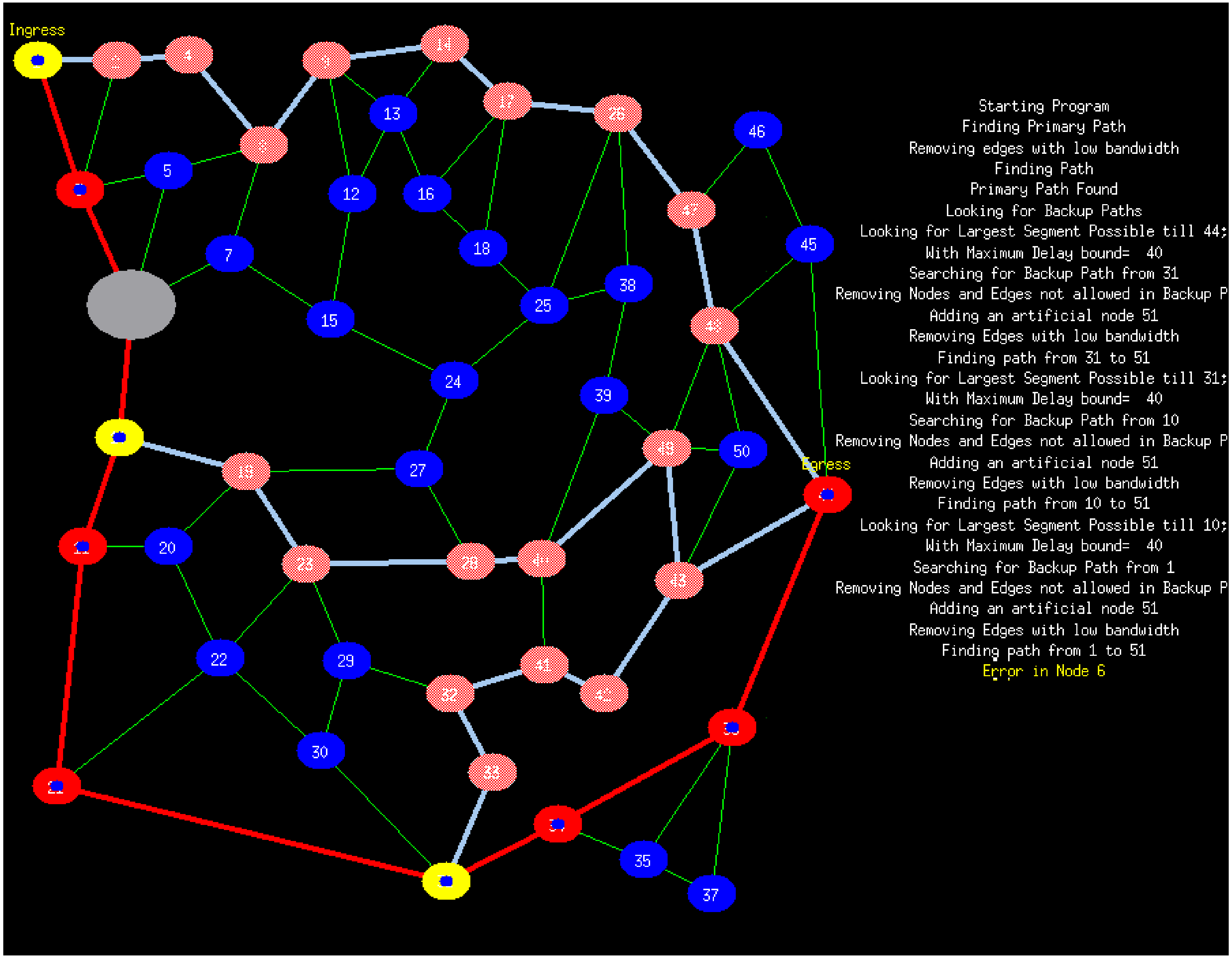


Figure 9.4: Packet flow over primary path and visualization of a node failure

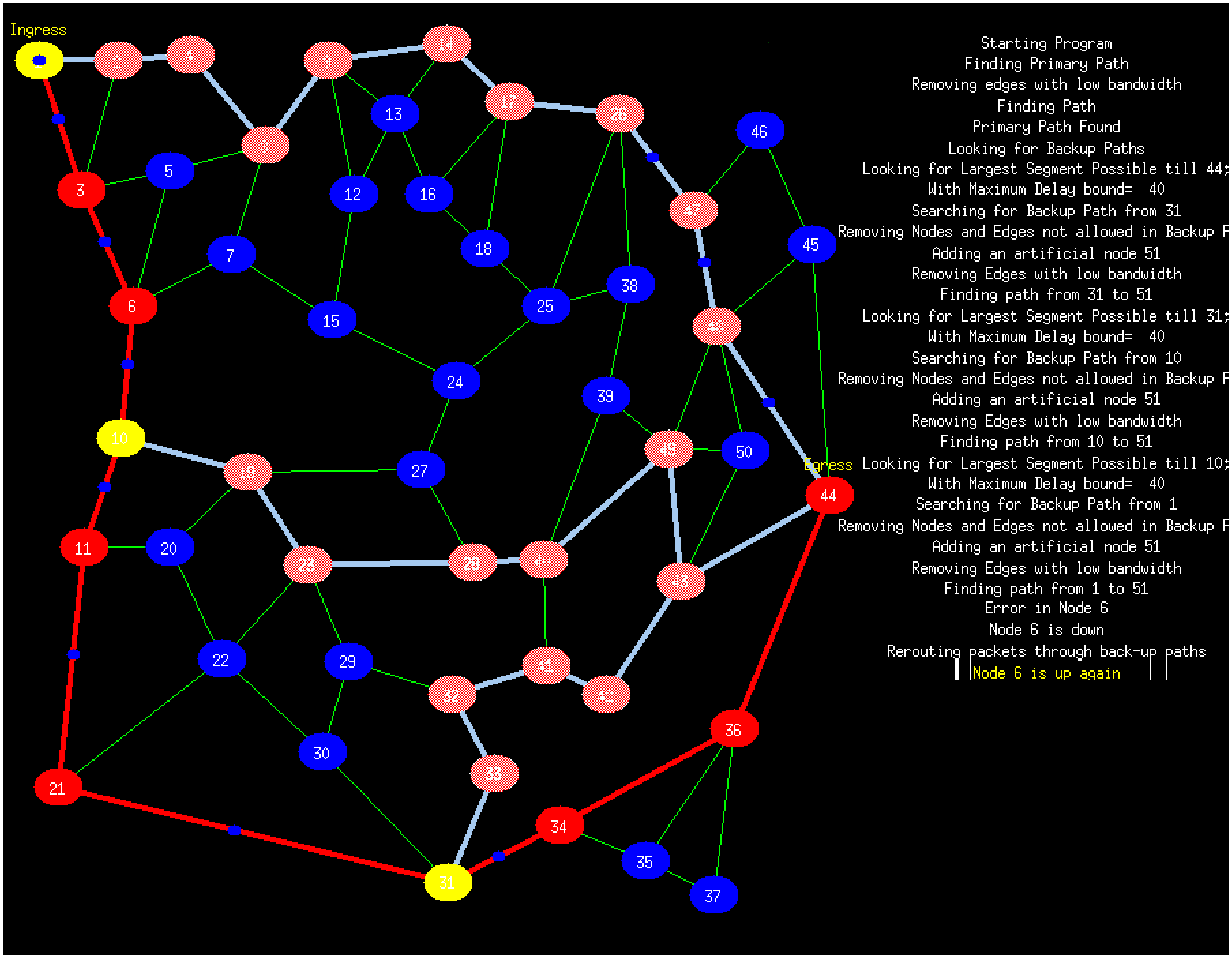


Figure 9.5: Packet flow over primary path and visualization of a node failure

Chapter 10

Results and Conclusion

In this project, we investigate a new scheme for path protection using a segment-based approach. The aim is to provide reliable and efficient transmission of data in case of failures and to satisfy various QOS constraints like switch over time, reliability without using more resources than necessary.

The segment-based approach provides lot of flexibility to the network administrator in providing path protection in MPLS networks as it tries to meet the constraints in a "tight" manner, reserving as much backup paths as necessary. Experimental results show the advantage of this approach that using fewer backup paths result in more conservative use of resources and in fact the improvement is significant (simulation results show only 50 percent protection resources as compared to Local Path Protection).

Essential mechanisms for providing Path Protection are discussed, especially in context of the new segment based approach and what changes need to be introduced in mechanisms for detecting and notifying the fault to the appropriate LSR.

Most importantly, we develop various algorithms to provide path protection with many QOS constraints namely:

1. Switch over time

2. End-to-end delay

3. Jitter

4. Reliability

5. Combination of the above parameters.

The aim of these algorithms is to provide a protection configuration consisting of a primary path and the minimum number of backup paths, while also satisfying the constraints.

Simulation results show the advantage of these algorithms over the standard protection techniques like Global Path Protection and Local Path Protection. Another important technique for conserving resources is the idea of sharing of backup bandwidth. Consideration of Sharing in our algorithms shows further improvement in the utilization of resources.

Chapter 11

Future work

In our work, we have developed algorithms for various QOS constraints using the segment-based approach and also implemented some algorithms and evaluated their performance.

There is further scope of work in the area of optimal resource usage by the protection resources. Currently we use the shortest path algorithm to find out the backup paths, but there may exist more efficient schemes for establishing backup paths, which allow for admission of more LSP requests. Much work has been done in the area of Optimal Spare Capacity Allocation (see references) and the proposed algorithms can be further extended to ensure optimal resource allocation.

The algorithms can also be implemented in the popular NS framework, for actual packet simulation. Use packet simulation, one can time the simulations and see the QOS constraints being satisfied using the proposed algorithms.

Also, label management and distribution issues also need to be addressed in detail, which are needed for implementing the algorithms on a real MPLS network. Changes will need to be made to the Next Hop Forwarding Tables of the Segment Switch Routers, and how these routers fill up those tables, the mechanism for this information exchange in the network, all this needs to be worked out.

These algorithms can also be incorporated into the MPLS Emulator, which has been developed at the Advanced Networking Lab here, which currently lacks any fault tolerance mechanisms.

Chapter 12

References

- J. Lawrence, "Designing Multiprotocol Label Switching Networks", IEEE Communications Magazine, July 2001
- E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001, Available at http://www.ietf.org/rfc/rfc3031.txt
- A. Banerjee, J. Drake, J. Lang, B. Turner, D. Awduche, L. Berger, K. Kompella, Y. Rekhter, "Generalized Multiprotocol Label Switching: An Overview of Signaling Enhancements and Recovery Techniques", IEEE Communications Magazine, July 2001
- A. Iwata, N. Fujita, T. Nishida, "MPLS Signaling Extensions for Shared Fast Rerouting", Internet Draft draft-iwata-mpls-shared-fast-reroute-00.txt, July 2001
- B. Cain, B. Jamoussi, "Requirement Framework for Fast Re-route with MPLS", Internet Draft draft-andersson-reroute-frmwrk-00.txt, October 1999
- D. Haskin, R. Krishnan, "A Method for Setting an Alternative Label Switched Paths to Handle Fast Reroute", Internet Draft draft-haskin-mpls-fast-reroute-01.txt, March 2000
- K. Owens, V. Sharma, S. Makam, C. Huang, "A Path Protection/Restoration Mechanism for MPLS Networks", Internet Draft draft-chang-mpls-path-protection-03.txt, July 2001
- M. Kodialam and T.V.Lakshman, "Dynamic Routing of Locally Restorable Bandwidth Guaranteed Tunnels using Aggregated Link Usage Information", Proceedings of IEEE Infocom 2001, March 2001
- M. Kodialam and T.V.Lakshman, "Dynamic Routing of Bandwidth Guaranteed Paths with Restoration", Proceedings of IEEE Infocom 2000, March 2000
- Nortel Networks, "An Introduction to Multiprotocol Label Switching", White Paper, April 2001

- Y. Liu, D. Tipper, D. Medhi, A. Srikitja, "Self configuring Survivable Techniques for Quality of Service Enabled Internet", Technical Report, Proceedings Information Survivability Workshops ISW-2000, Boston, October 2000, Available at www.cert.org/research/isw/isw2000/papers/35.pdf

- W. Cui, "Efficient Bandwidth Allocation of Backup Paths", Class Project Report, University of California-Berkeley, Spring 2001, Available at www.eecs.berkeley.edu/ wdc/classes/cs268-report.ps

- W. Grover et al, "New Options and Insights for Survivable Transport Networks", IEEE Communications Magazine (Special Feature on DRCN Workshop topics), Nov 2001

Chapter 13

Appendix

13.1 Introduction to MPLS Networks

With the exponential growth of the Internet over the last few years, technology has had to adapt constantly to new demands for increased bandwidth. In addition, the Internet will continue to see dramatic growth due to the ever-increasing demand for more bandwidth to the home, with projections of every household having broadband access in the future.

With over 250 million new users projected in the next decade and with the implementation of Internet protocol version 6 (IPv6), carriers and service providers struggle to scale their current infrastructures for the inevitable demand on their networks.

In order to meet the growing demand for bandwidth, Internet service providers (ISPs) need higher performance switching and routing products. Although most carrier and service provider core networks run on impressive asynchronous transfer mode (ATM) backbones, most connections to these providers continue to be slow frame relay and point-to-point connections, introducing latency and sometimes bottlenecks at the edge access points. Core network routers also contribute to latencies, as each must make its own individual decision on the best way to forward each incoming packet. Traditionally, IP has been routed over ATM using IP over ATM via virtual circuits (VCs) or multi-protocol over ATM (MPOA). These forwarding methods proved to be cumbersome and complicated. The need for a simpler forwarding method one with the traffic management features and performance of traditional switches combined with the forwarding intelligence of a router is definitely felt.

All of these needs can be met with multi-protocol label switching (MPLS), because it integrates the key features of both Layer 2 and Layer 3. Most importantly, it is not limited to any Layer 2 or Layer 3 protocol. In particular, MPLS has several applications and can be extended across multiple product segments (such as an MPLS router, an IP services switch/router, a multi-service switch, an Optical Ethernet switch, as well as optical switches).

13.1.1 Traditional routing

In traditional routing environments, a packet is forwarded through a network on a hop-by-hop basis using interior gateway protocols (IGPs), such as routing information protocol (RIP) and open shortest path first (OSPF), or exterior gateway protocol (EGPs) such as border gateway protocol (BGP). This is done by referencing the destination Layer 3 addresses against a routing table for a next hop entry. To clarify, each router that a packet traverses must do a route look-up, based on that destination Layer 3 address in the IP header. This must be performed to determine the packet s next hop in its path to get it to its final destination. The Layer 2 destination address is then replaced with the address of the next hop s Layer 2 address, and the source Layer 2 address is then replaced with the Layer 2 address of the current router, leaving the source and destina-tion Layer 3 addresses in place for the next hop to perform its own route look-up on the packet. This process must be repeated at each hop to deliver the packet to its final destination. In Figure 1, to forward packets to Router F, Router C will reference only the destination address of Router F. Router C will then determine the best route based on the attributes that are defined for that particular IGP. If the router is using RIP, the lowest sum of all hops to the destination is preferred as the best path, as long as the total number of hops does not exceed 15. If the IGP is OSPF, the total cumulative cost (i.e., metric, usually based on bandwidth) to a destination is referenced, and the lowest total cost of all links is preferred.

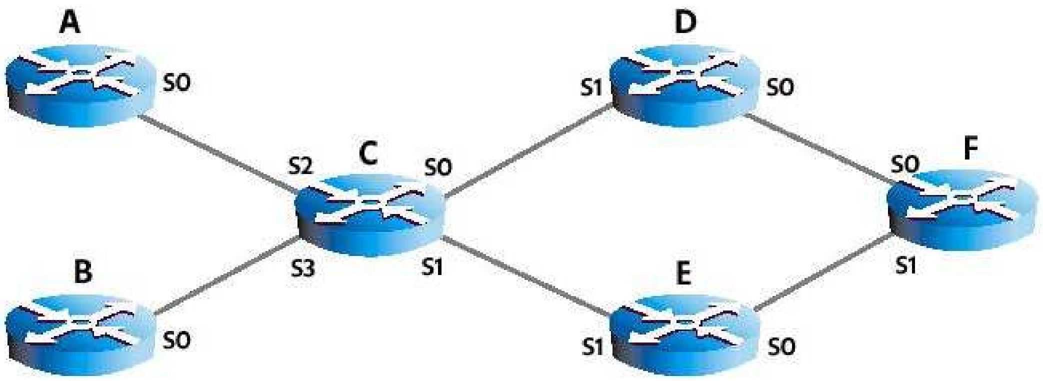


Figure 13.1: Traditional Routing Example

Running IGPs such as RIP and OSPF have provided scalable solutions, but fall short when introducing the need for inter-AS (autonomous system) routing, network management, traffic engineering, and scalable IP services. To reference Figure 1 again, in traditional routing environments, Router C must make its forwarding decisions for packets destined to Router F based on the metrics defined by the IGP being implemented. If OSPF, the metric can be based on various criteria, although bandwidth is usually the one used. RIP, on the other hand uses a metric based on hop count and drops packets after 15 hops. As seen in Figure 1, all packets coming from Router A or B destined for Router F will be forwarded in the same way, along the path with the preferred metric. Therefore, if the path to Router F via Router D is a higher bandwidth such as two DS-3s and the path via Router E was connected through T-1s, the path via Router D would be the only one used unless a network failure occurred.

13.1.2 Introduction to MPLS

In contrast, MPLS is a method of forwarding packets at a high rate of speed. It combines the speed and performance of Layer 2 with the scalability and IP intelligence of Layer 3. Routers on the edge of the network (label edge routers [LERs]) attach labels to packets based on a forwarding equivalence class (FEC). Packets would then be forwarded through the MPLS network, based on their associated FECs, through swapping the labels by routers or switches in the core of the network called label switch routers (LSRs) to their destination. The most important idea is that by adding a simple label, the LSR is able to switch a packet much more efficiently, due to a simple forwarding element, in contrast to the hop-by-by basis used in traditional routing. A label is usually a locally significant, condensed view of the IP header, which is bound to a FEC (such as a given IP prefix) that is then referenced against a table of incoming labels to outgoing labels and interface mappings called the label information base (LIB). The label itself represents the particular FEC to which it was mapped. By referencing an LIB, the true strengths of MPLS traffic engineering capabilities become quite apparent. When an LSR or LER constructs its LIB, explicit control can be used to direct a packets route through a network.

The true strength of the MPLS forwarding algorithm is that analysis of the IP packet header only needs to be done once, at the ingress of the MPLS domain by an LER. Once a packet has been assigned to a FEC, forwarding of the packet can be done solely on the labels used by the particular label switch path (LSP).

In Figure 2, we see that by using MPLS, we have granular control over a packet s path. Packets destined to Router F sourcing from Router A follow the solid green LSP. Packets originating from Router B will be forwarded along the dotted red LSP. This is accomplished by referencing incoming labels to the LIB in order to attain the value of the outgoing label and the outgoing inter-face. Figure 3 is an example of a router s LIB. Packets destined for a FEC off Router F will arrive at Router C on interface S2 with a label of 50. They will be referenced against an LIB to ascertain a forwarding decision. Based on the LIB, packets with a label value of 50 arriving on interface S2 will be swapped for a label with a value of 12 and be switched out interface S0.

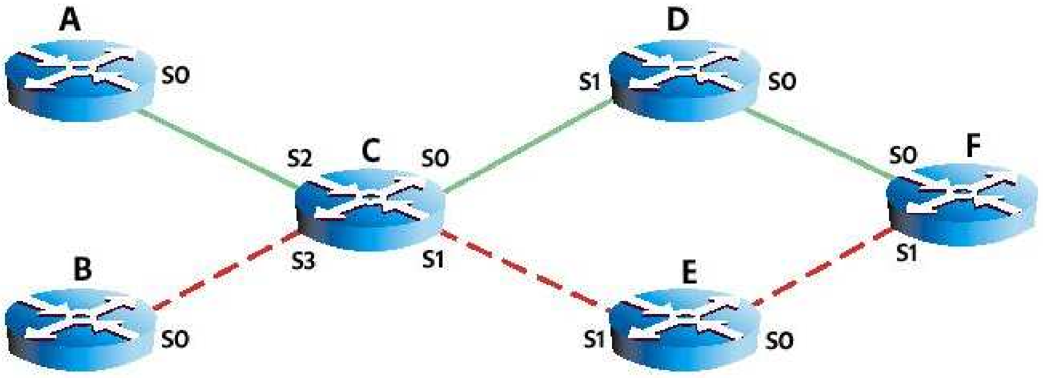


Figure 13.2: MPLS Traffic Engineering Example

Interface IN	Label In	Destination	Exit Interface	Label Out
S2	50	F	S0	12
S3	45	F	S1	98

Figure 13.3: Router C's routing table

The strength of controlling traffic patterns quickly becomes apparent when compared to traditional IGPs (such as OSPF), which forward packets based on the destination Layer 3 address only. Most OSPF network designs with multiple paths to the same destination use only the route with the lowest accumulative cost.

13.1.3 Advantages of MPLS

In the MPLS forwarding paradigm, once a packet is assigned to a FEC, no further header analysis is done by subsequent routers; all forwarding is driven by the labels. This has a number of advantages over conventional network layer forwarding.

- MPLS forwarding can be done by switches which are capable of doing label lookup and replacement, but are either not capable of analyzing the network layer headers, or are not capable of analyzing the network layer headers at adequate speed.
- Since a packet is assigned to a FEC when it enters the network, the ingress router may use, in determining the assignment, any information it has about

the packet, even if that information cannot be gleaned from the network layer header. For example, packets arriving on different ports may be assigned to different FECs. Conventional forwarding, on the other hand, can only consider information which travels with the packet in the packet header.

- A packet that enters the network at a particular router can be labeled differently than the same packet entering the network at a different router, and as a result forwarding decisions that depend on the ingress router can be easily made. This cannot be done with conventional forwarding, since the identity of a packet's ingress router does not travel with the packet.

- The considerations that determine how a packet is assigned to a FEC can become ever more and more complicated, without any impact at all on the routers that merely forward labeled packets.

- Sometimes it is desirable to force a packet to follow a particular route which is explicitly chosen at or before the time the packet enters the network, rather than being chosen by the normal dynamic routing algorithm as the packet travels through the network. In conventional forwarding, this requires the packet to carry an encoding of its route along with it ("source routing"). In MPLS, a label can be used to represent the route, so that the identity of the explicit route need not be carried with the packet.

13.2 Acronyms and Abbreviations

FEC	Forwarding Equivalence Class
FTN	FEC to NHLFE Map
ILM	Incoming Label Map
IP	Internet Protocol
LDP	Label Distribution Protocol
L2	Layer 2
L3	Layer 3
LSP	Label Switched Path
LSR	Label Switching Router
MPLS	Multi-Protocol Label Switching
NHLFE	Next Hop Label Forwarding Entry
SSR	Segment Switch Router
TTL	Time-To-Live

13.3 Terminology

Forwarding Equivalence Class	A group of IP packets which are forwarded in the same manner (e.g., over the same path, with the same forwarding treatment)
Label	A short fixed length physically contiguous identifier, which is used to identify a FEC, usually of local significance.
Label swap	The basic forwarding operation consisting of looking up an incoming label to determine the outgoing label, encapsulation, port, and other data handling information.
Label swapping	A forwarding paradigm allowing streamlined forwarding of data by using labels to identify classes of data packets, which are treated indistinguishably when forwarding.
Label switched hop	The hop between two MPLS nodes, on which forwarding is done using labels.
Label switched path	The path through one or more LSRs at one level of the hierarchy followed by a packets in a particular FEC.
Label switching router	An MPLS node which is capable of forwarding native L3 packets
Layer 2	The protocol layer under layer 3 (which therefore offers the services used by layer 3). Forwarding, when done by the swapping of short fixed length labels, occurs at layer 2 regardless of whether the label being examined is an ATM VPI/VCI, a frame relay DLCI, or an MPLS label.
Layer 3	The protocol layer at which IP and its associated routing protocols operate link layer synonymous with layer 2
Loop detection	a method of dealing with loops in which loops are allowed to be set up, and data may be transmitted over the loop, but the loop is later detected
Loop prevention	a method of dealing with loops in which data is never transmitted over a loop
Label stack	An ordered set of labels
MPLS domain	A contiguous set of nodes which operate MPLS routing and forwarding and which are also in one Routing or Administrative Domain

MPLS edge node	An MPLS node that connects an MPLS domain with a node, which is outside of the domain, either because it does not run MPLS, and/or because it is in a different domain. Note that if an LSR has a neighboring host, which is not running MPLS, that that LSR is an MPLS edge node.
MPLS egress node	An MPLS edge node in its role in handling traffic as it leaves an MPLS domain
MPLS ingress node	An MPLS edge node in its role in handling traffic as it enters an MPLS domain
MPLS label	A label which is carried in a packet header, and which represents the packet's FEC
MPLS node	A node, which is running MPLS. An MPLS node will be aware of MPLS control protocols, will operate one or more L3 routing protocols, and will be capable of forwarding packets based on labels. An MPLS node may optionally be also capable of forwarding native L3 packets.