4

RELIABLE AND FAST REROUTING (RFR)

4.1 INTRODUCTION

Fast rerouting has been recognized as a key component of providing service continuity to end users. We focus on improving current mechanisms for *Reliable and Fast Rerouting* (RFR). Given the function of fast rerouting mechanisms in the previous chapter, it is straightforward to introduce modifications to yield the reliable and fast rerouting mechanism.

In the proposal of Chapter 3, we were able to significantly reduce average delay due to path restoration while eliminating packet disorder for traffic in MPLS networks for a protected LSP. However, critical services (e.g. important traffic from premium customers) will be affected by packet losses and, for TCP traffic, lost packets trigger retransmission requests; hence the gains due to the decrease in restoration time may become negligible. As a consequence, poor performance and degraded service

delivery will be experienced and QoS parameters will be seriously affected during the restoration period.

The main factors that affect the performance of fast rerouting mechanisms are packet loss, traffic recovery delay (Full Restoration Time) and packet disorder. Our previous work has addressed the last two factors. Up to now, packet loss due to node or link failure was considered "inevitable" [SH02][BR02]. It has always been assumed that the transport layer would somehow take care of the retransmission of lost packets - eventually. It is for this reason, we believe, that there has not been any previous work aimed at eliminating packet loss. We have observed that the retransmission process due to packet loss significantly affects the throughput of TCP traffic due to the startup behavior (slow-start) of TCP. This point is briefly addressed in Chapter 5.

In this chapter we propose RFR, a novel recovery algorithm with small local buffers in each LSR node within the protected path in order to eliminate both *packet loss* due to link/node failure and *packet disorder* during the restoration period. This results in a significant throughput improvement for premium traffic.

It is important to note that the objective of this study is to provide and guarantee QoS for critical traffic carried by protected LSPs in MPLS networks and that not all LSPs are protected.

4.2 PROPOSED MECHANISM

The proposed mechanism is based on the mechanism already proposed in Chapter 3. We use the same figure to describe this proposal. In Figure 4.1, the ingress and egress nodes respectively are LSR0 and LSR4. The protected LSP is formed by the LSR nodes 0,1,2,3 and 4. If a link failure is detected by LSR3 - as shown in the figure, the path back to the ingress LSR will consist of the nodes 3,2,1 and 0 (*backward LSP*). The *alternative LSP* will be formed by the LSR nodes 0,5,6,7,8 and 4. We assume

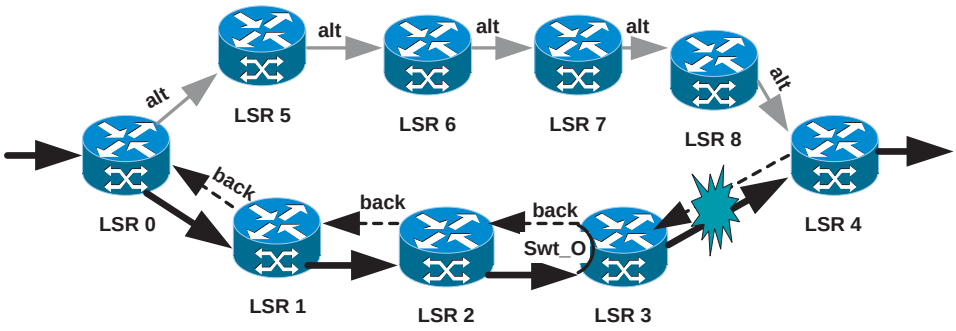

Figure 4.1 Simulation scenario

that the backward and alternative LSPs have already been set-up. As soon as an LSR node belonging to the protected LSP detects a fault, a switchover is established and packets are sent back through the newly activated *backward LSP*. The first packet that is sent back is used as a fault-detect notification.

In our proposal, each LSR in the protected path has a local buffer into which a copy of the incoming packet is saved while it is being forwarded to the next LSR along the protected path. The maximum size this buffer needs to be is about twice the number of packets that can circulate in a given link of the protected LSP. This is because the failure can occur either on a link or at a node. If the link fails, only the packets occupying the link from LSR3 to LSR4 during the failure would potentially be lost (Figure 4.1). If node LSR3 fails, packets on two links will have to be recovered.

There are two possible modes to store the incoming protected packets to the local buffer during the NORMAL condition. The first, called the non-swapped mode, is to store the protected packets before the swapping procedure to the backward/alternative LSP is done. This consists of a simple copy of packets to the local buffer as the packets are received by an LSR. The second is the swapped mode, in which the LSR stores the protected packets to the local buffer after executing the swapping procedure to the backward/alternative LSP. Both modes work well. The main differences between these approaches are the delay and the additional process overhead.

In the non-swapped mode once the fault is detected on the protected LSP, the LSR takes the packet from the local buffer and looks at the packet header ("shim" header or MPLS header) and then proceeds to swap it for the corresponding output label and changes the output interface. Note that this is the second time that the LSR looks at this header. The first was when the packet arrived at the LSR for the first time to be copied to the local buffer. This method introduces delays, processing overhead and additional CPU requirements. On the other hand, in the swapped mode the LSR sends the packets from local buffer directly to the output interface, as it did the swapping process before while in the normal condition, providing better performance than the non-swapped mode. For this reason our proposal uses the swapped mode.

4.2.1 Behavior of the Node that detects the failure

When a fault is detected by an LSR, a switchover procedure is initiated immediately (assuming that the fault-detection-time is zero) and all the packets in its buffer are drained and sent back via the backward LSP. Any subsequent packet coming in on the protected LSP is also sent back. The switchover consists of a simple label swapping operation from the protected LSP to the backward LSP. Note that this node has *copies of packets* that were dropped from the faulty link/node and hence there is *no packet loss*.

4.2.2 Behavior of all other nodes on the backward LSP

As soon as each node of the *backward LSP* detects the first packet coming back (sign of fault or problem downstream), it forwards this packet along the *backward LSP* and invalidates all data that are stored in its buffer for recovery of data from a possible link/node failure associated with this output interface. The next packet coming in from the upstream LSR of the protected LSP will be tagged and forwarded to the downstream LSR via the protected LSP. All subsequent packets that arrive at this node or LSR along the protected path are stored in its buffer without being forwarded (Chapter 3). This contributes significantly to the reduction of the average

packet delay because it avoids the circulation of packets along the loop formed by the already broken protected LSP and the backward LSP.

4.2.3 Role of tagging in eliminating disorder of packets

When a node detects the packet it tagged (the last packet sent downstream before starting to store incoming packets) coming along the *backward LSP*, it knows that all downstream packets have been drained and that it must now send back all its buffered packets. By doing this, it is able to preserve the ordering of packets. Using one of the *Exp* field bits of the MPLS label stack [RTF+01] for the purpose of tagging avoids any overheads.

Each LSR along the *backward LSP* successively sends back its stored packets when it receives its tagged packet. Note that the node responsible for removing the tag is the same node (LSR) which tagged it. When all packets return to the ingress LSR (i.e., the ingress LSR receives its tagged packet) and have been rerouted to the *alternative LSP*, the restoration period terminates. The packets stored during this time in the ingress LSR, along with all new incoming packets (from the source) are now sent via the alternative LSP. Note that at the end of the whole process, global ordering of packets is preserved, packet loss has been eliminated, and the proposal has a shorter restoration period than Haskin's.

4.3 ALGORITHM DESCRIPTION

Figure 4.2 presents the state diagram of the proposed algorithm (RFR). Though the state diagram by itself is a formal description, a detailed explanation of the process follows. We introduce a new field in the label information based-forwarding table (LIB) called status (link state). Five link state identifiers are defined for a protected LSP: NORMAL, FAULT DETECT, ALTERNATE DETECT, STORE BUFFER and SEND BUFFER.

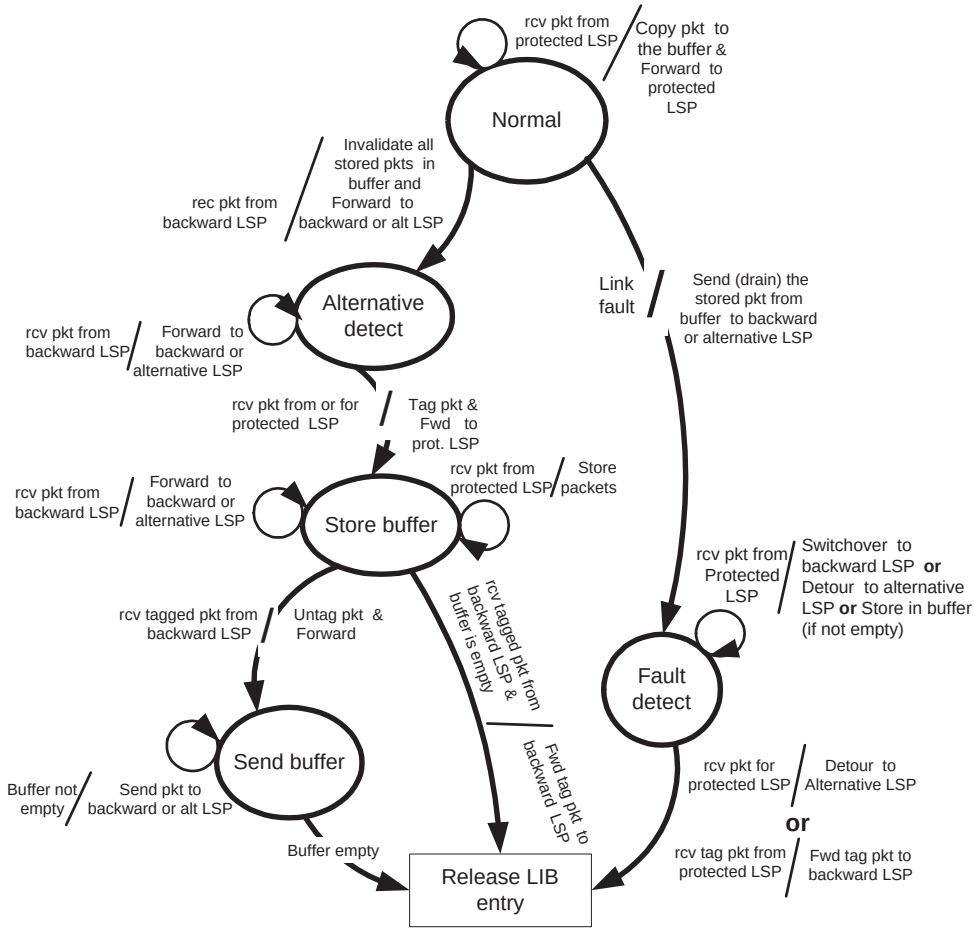

Figure 4.2 RFR state machine diagram

Since this proposal is an extension of the mechanism proposed in Chapter 3, the change introduced in this proposal affects only the NORMAL, ALTERNATE_DETECT and FAULT_DETECT state functions. The functions of STORE_BUFFER and SEND_BUFFER remain unchanged.

The proposed algorithm functions as follows. In the normal condition all LSRs store a copy of received packets in the local buffer. The buffer is dimensioned with sufficient capacity to protect against packet losses during a link/node failure in the protected LSP.

Once a failure along the protected LSP is detected, the protected LSR that detects the fault performs the switchover procedure (LSR3 in Figure 4.1). This procedure consists of a simple label swapping operation from the protected LSP to the backward LSP for all packets with a label corresponding to the protected LSP. The link status of the label information base forwarding table (LIB) of this LSR is changed from NORMAL to FAULT_DETECT (Figure 4.2). It then begins to drain all the packets stored in its buffer - i.e., send them back along the backward LSP. Any incoming packets on the protected LSP are also sent back.

The immediate upstream LSR, in this case LSR2 (Figure 4.1) receives these reversed packets from LSR3 through the backward LSP. When it detects the first packet coming on the backward LSP, it changes the link status of the LIB entry of the protected LSP corresponding to this backward LSP to ALTERNATIVE_DETECT (Figure 4.2). Additionally, it invalidates all data in its buffer. Recall that these packets are stored to be used in case the output link fails. When the LSR enters the ALTERNATE_DETECT state the buffer is emptied and will be used to store packets coming in from the protected LSP until they may be forwarded through the backward LSP.

The next, immediate packet received from the protected LSP sees the LIB entry as ALTERNATIVE_DETECT. This indicates that there is a link problem somewhere in the protected LSP. This packet is then tagged as the last packet from this LSR (LSR2) and forwarded normally downstream and the LIB entry status is changed from ALTERNATIVE_DETECT to STORE_BUFFER (Figure 4.2). The subsequent packets coming in on the protected LSP will be stored in the buffer because they will find the link status is STORE_BUFFER. This continues until the tagged packet is received through the backward LSP.

In order to detect the tagged packet coming back on the backward LSP, the LSR has to check if the tag bit of the received packet is *set* or *not*. If the comparison result is false the packet will be forwarded using the normal swapping operation. Otherwise, the LSR knows that no more packets are expected from the backward LSP. Note that

at this point there are two possible actions depending the buffer condition. Here we assume the buffer is "not empty" to describe the complete algorithm. In this case, the tag bit in the label must be disabled (set to 0) and the packet is sent according to the label swapping result as a normal packet. Moreover, it changes the status from STORE_BUFFER to SEND_BUFFER, and then when the buffer is empty, the label is removed from the LIB.

This process is repeated at every LSR up to the ingress LSR. Although in this description we presented the example of link failure, our algorithm can also be used without requiring any additional algorithm for node failure restoration.

4.4 DERIVATION OF THE MODEL

The mathematical formulation of our model is an important step to validate the simulation results. Once we do this, we can study the trade-offs between the cost of using buffers in each LSR within the protected path and the benefits that accrue in terms of performance for high-priority QoS traffic. The size of the buffers required both at the ingress node and at the intermediate nodes between the ingress and the point of failure can be estimated from the derived model and validated by our simulation. The following are the terms used in our derivation with a brief explanation of each:

T_{tran} : − Transmission delay time or packet transmission time,

T_{prop} : − Propagation delay time in a link,

V_{T_lsp} : − Source rate (reference traffic),

B_{W_lsp} : − LSP bandwidth that is the peak rate admitted,

P : − Packet size,

d : − Distance between two adjacent LSRs,

$T_{recovery}$: − Full restoration time,

T_{fault_detect} : − Fault detect time,

$B_{ingress}$: − Buffer size in ingress LSR,

N : − Number of LSR that detects the fault (i.e., number of nodes of the backward LSP excluding the ingress node).

According to our generalized network simulation model (Figure 4.3), after the detection of a failure the total time required by the node detecting the failure to switchover all packets (including buffered packets) and the time for the tagged packet to return to the immediate upstream node must be calculated. This is equal to the link delay for the first packet switched over to reach the next upstream LSR along the backward LSP, plus the round trip link delay for the tagged packet to return to its node (the node which tagged it).

$$T_{switch\_over} = 3 * T_{link} \tag{4.1}$$

Where, T_{link} (link delay) is calculated in our case as the sum of the transmission $\left(\frac{P}{B_{W\_lsp}}\right)$ and propagation $\left(\frac{d}{c}\right)$ delays, assuming that both queuing and processing delays are zero. The reason is that the NS-2 simulator does not account for queueing and processing delays.

$$T_{link} = T_{tran} + T_{prop} \tag{4.2}$$

The rest of the delays up to the point of restoration of traffic along the *alternative LSP* are the sum of the delays for each intermediate LSR to pass back all of its packets to the immediate upstream node. This time can be broken down into two components: (1) time taken to drain all packets from its buffer, and (2) time taken for the last packet (the one that was *tagged*) to reach the next upstream node (T_{link}).

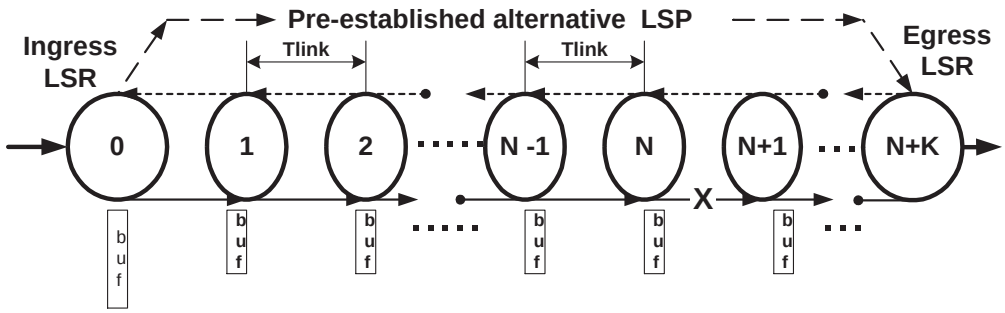

Figure 4.3 Model for equation. Solid line: Protected LSP; Dashed line: Backward LSP

The store period is $2 * T_{link}$ (two-way delay for the tagged packet). Given that the packets that are stored in the buffer arrive at the rate of reference traffic (V_{T_lsp}) during $2 * T_{link}$ and the rate at which the packets are drained from the buffer is equal to the bandwidth (B_{W_lsp}), we have:

$$T_{int\_buffer\_drain\_pkt} = \frac{2 * T_{link} * V_{T\_lsp}}{B_{W\_lsp}} \tag{4.3}$$

and the intermediate LSR delay time (T_{int}),

$$T_{int} = T_{int\_buffer\_drain\_pkt} + T_{link} \tag{4.4}$$

Once we know T_{fault_detect}, T_{switch_over}, T_{int} and N we can calculate the total restoration time starting from the time that the fault was detected. Note that we assume T_{link} over all links is the same (i.e., all links operate at the same rate (B_{W_lsp}) and have the same propagation delay (d)). The sum of delays in the intermediate LSRs is equal to $\sum_{i=1}^{N-1}(T_{int})_i = (N-1) * T_{int}$.

$$T_{recovery} = T_{fault\_detect} + T_{switch\_over} + \sum_{i=1}^{N-1} (T_{int})_i \tag{4.5}$$

applying our previous condition:

$$T_{recovery} = T_{fault\_detect} + T_{switch\_over} + (N-1) * T_{int} \tag{4.6}$$

We assume that the time to detect the fault by an LSR - $T_{fault\_detect} = 0$ since this affects all recovery schemes equally. Then the above equation becomes:

$$\mathbf{T_{recovery}} = \mathbf{T_{link}} \left(\mathbf{N + 2 + 2\,(N - 1)} \frac{\mathbf{V_{T\_lsp}}}{\mathbf{B_{W\_lsp}}} \right) \tag{4.7}$$

Finally, for the **worst case** (i.e., when the $V_{T\_lsp} = B_{W\_lsp}$)

$$\mathbf{T_{recovery}} = \mathbf{3 * N * T_{link}} \tag{4.8}$$

4.4.1 Buffer size requirement calculation for the ingress LSR during the restoration period.

The required buffer size in the ingress LSR is an important factor for the implementation of the proposed mechanism. This node has to store packets from the time it receives the first packet on the backward LSP switched over from the alert node (point-of-failure) until it receives its own tagged packet. The time taken by the former is:

$$T_{ing\_rcv\_first\_pkt} = N * T_{link} \tag{4.9}$$

and the time at which the latter takes place is $T_{recovery}$ (Figure 4.4). Therefore,

$$T_{ing\_store\_pkt} = T_{recovery} - T_{ing\_rcv\_first\_pkt} \tag{4.10}$$

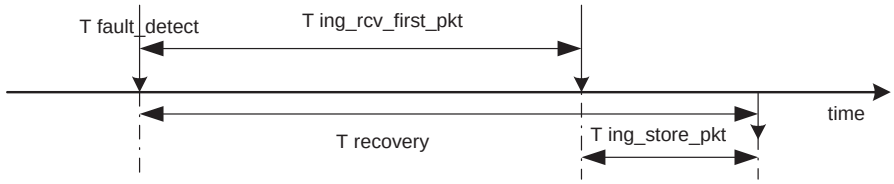

Figure 4.4 Graphical representation of times for ingress buffer calculation

The required buffer size in the ingress LSR during the restoration period is:

$$B_{ingress} = T_{ing\_store\_pkt} * V_{T\_lsp} \tag{4.11}$$

and hence,

$$\mathbf{B_{ingress} = 2 * T_{link} * V_{T\_lsp} * \left(\frac{(N-1)\, V_{T\_lsp}}{B_{W\_lsp}} + 1 \right)} \tag{4.12}$$

For the worst case, when V_{T_lsp}=B_{W_lsp},

$$\mathbf{B_{ingress} = 2 * N * T_{link} * V_{T\_lsp}} \tag{4.13}$$

The required buffer size in each intermediate LSR during the restoration period is:

$$\mathbf{B_{intermediate} = 2 * T_{link} * V_{T\_lsp}} \tag{4.14}$$

4.5 SIMULATIONS AND RESULTS

The objective of the simulation is to validate the formula and to compare the behavior of the proposed mechanism with previous MPLS protection mechanisms [HK00].

The simulated scenario is the one shown in Figure 4.1. The simple network topology with a protected and alternative LSP is used. We extend the simple network topology for different numbers of intermediate LSRs in the protected LSP. We vary the location of the node that detects a fault (alert LSR).

Parts of the MNS source code were modified to simulate both mechanisms (ours and Haskin's [HK00]) and the modified simulator was validated with previously published results for Haskin's method [GW01b] [HD01].

We present the results for CBR traffic flow with the following characteristics: packet size = 200 bytes, source rate= 400Kbps, burst time=0 and idle time =0.

The results based on the derived formula for the proposed model are plotted with the corresponding simulation results, for Full Restoration Time (Figure 4.5) and for the buffer size needed at the ingress LSR (Figure 4.6). These figures show that the analytical results are almost identical to the simulation results, validating our analytical expression of the proposed mechanism (RFR).

Observe that in both cases (Figures 4.5 and 4.6) for the $B_{W\_lsp} = 1Mbps$ the restoration time and the ingress LSR buffer requirement increase due to the fact that the transmission speed of the packets is low compared to 5Mbps, 10Mbps and above. The time required to reach the ingress LSR depends on the speed. The restoration time basically depends on the transmission speed and the same applies for the buffer requirements at the ingress LSR.

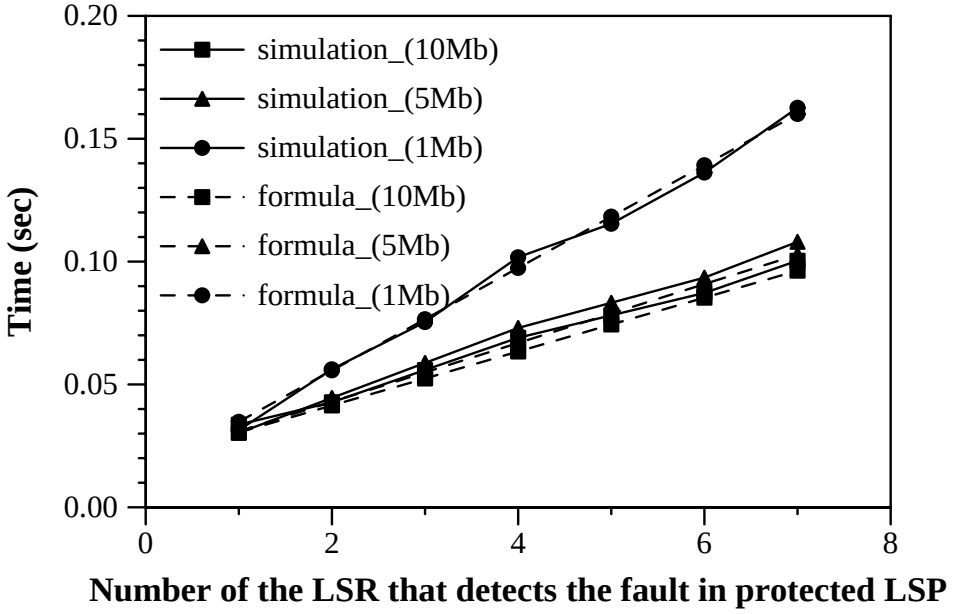

Figure 4.5 Recovery time for different LSP bandwidths

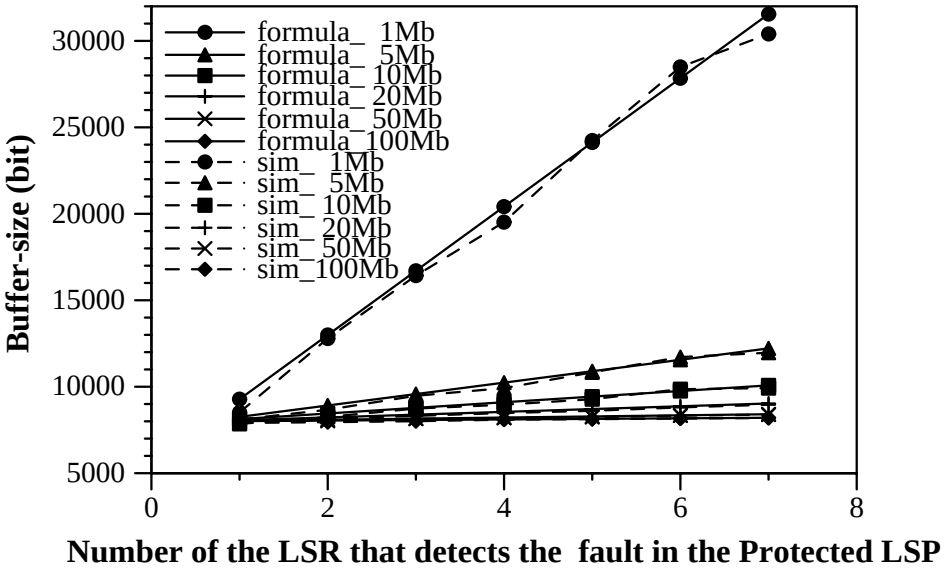

Figure 4.6 Ingress buffer size for Vt_lsp=400k and Pkt_size=200bytes for different LSP bandwidths and numbers of LSR (N)

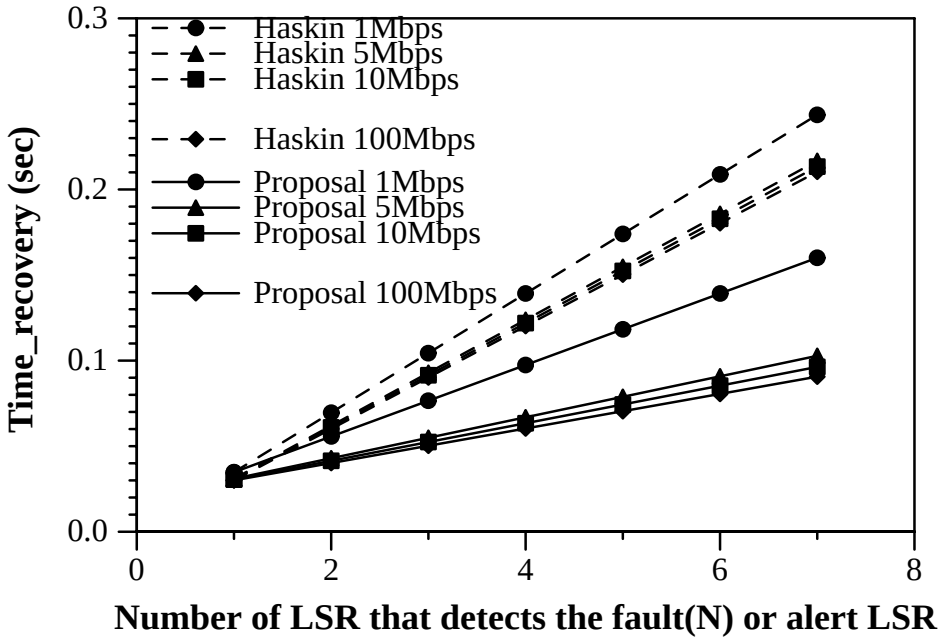

Figure 4.7 Restoration delay for 200 bytes packet size for different LSP bandwidth and number of alert LSR (N) using formula (derived model)

The plots in Figures 4.7 and 4.8 correspond to the comparison of the overall restoration period between Haskin's scheme and RFR for the derived model and the simulation respectively.

We use Figure 4.8 results for comparison of the overall restoration period for both proposals for different points of failure and for different bandwidths. Time is computed from the instant the fault is detected until the protected LSP is completely eliminated. Our proposed mechanism significantly improves the Full Restoration Time. A time reduction of 24.6%, 27.9%, 29.8%, 31.7% and 33% for the 3rd, 4th, 5th, 6th and 7th node of the alert LSR on the protected LSP respectively are achieved. Note that the above percentage values correspond to an LSP bandwidth of 1Mbps. The improvements are greater as the bandwidth increases and the transmission rate of the source remains fixed.

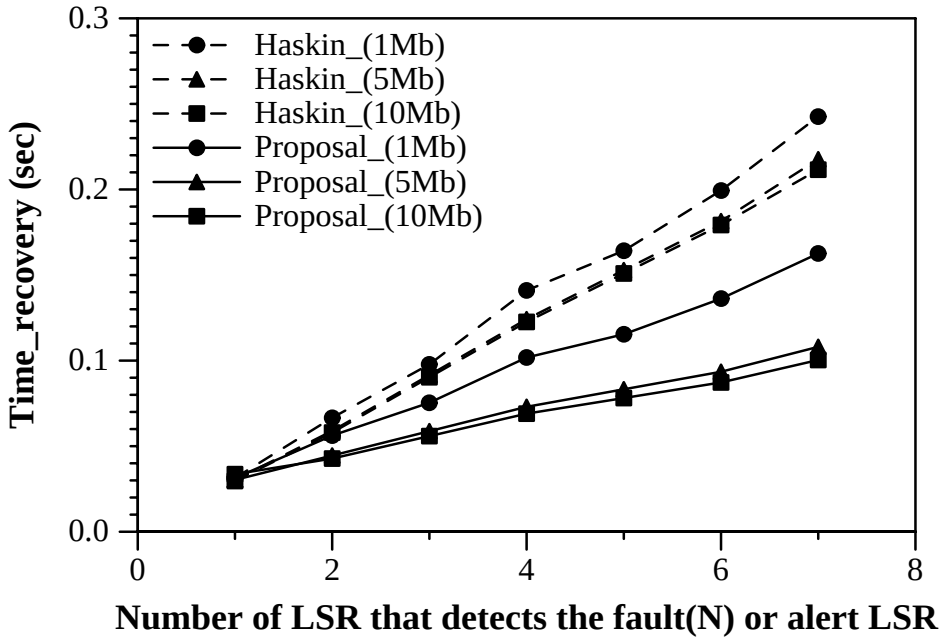

Figure 4.8 Restoration delay for 200 bytes packet size for different LSP bandwidth and number of alert LSR (N) using the simulator

In Figure 4.9 we present the results concerning the ingress node buffer requirement, varying the B_{W_lsp}, distance (d) and N. Results show that even for a long-distance LSP the buffer size required at the ingress node is reasonable compared to the benefits provided by the RFR mechanism.

In Figure 4.10 we maintain the V_{T_lsp} and the distance (d) constant, and vary the B_{W_lsp} and N. In this case, as we increase the B_{W_lsp} the effect of N on the required ingress buffer space becomes negligible. Note that in both cases we maintain the packet size (P) constant. The buffer space (memory) requirements for the implementation of our proposal under different conditions are clearly demonstrated. We have plotted the buffer needs for the ingress node for the longest period it could theoretically have to store packets (waiting for all downstream nodes to drain their packets).

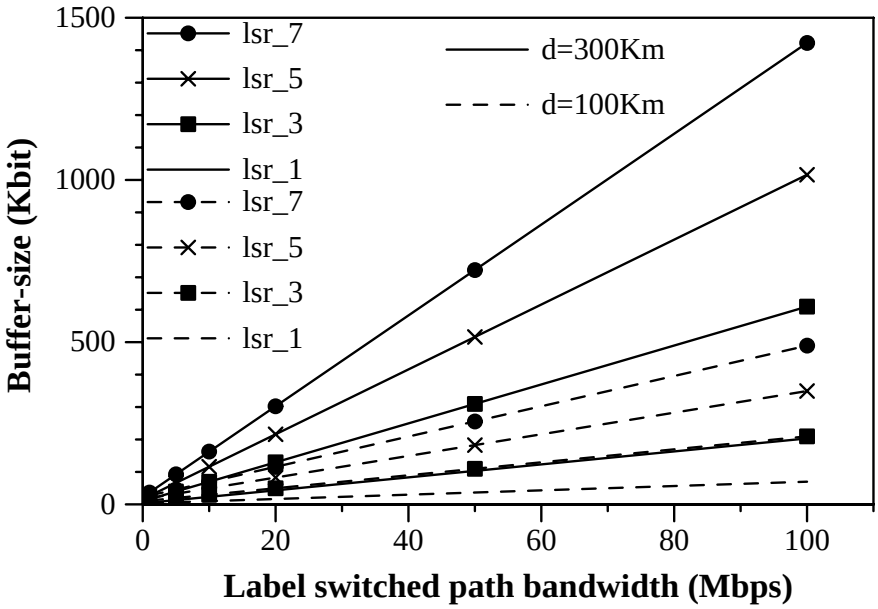

Figure 4.9 Required buffer space for ingress LSR when $Vt\_lsp = Bw\_lsp$ (worst case) and Pkt_size=1600 bits for d=300Km and d=100Km varying the Bw_lsp and N

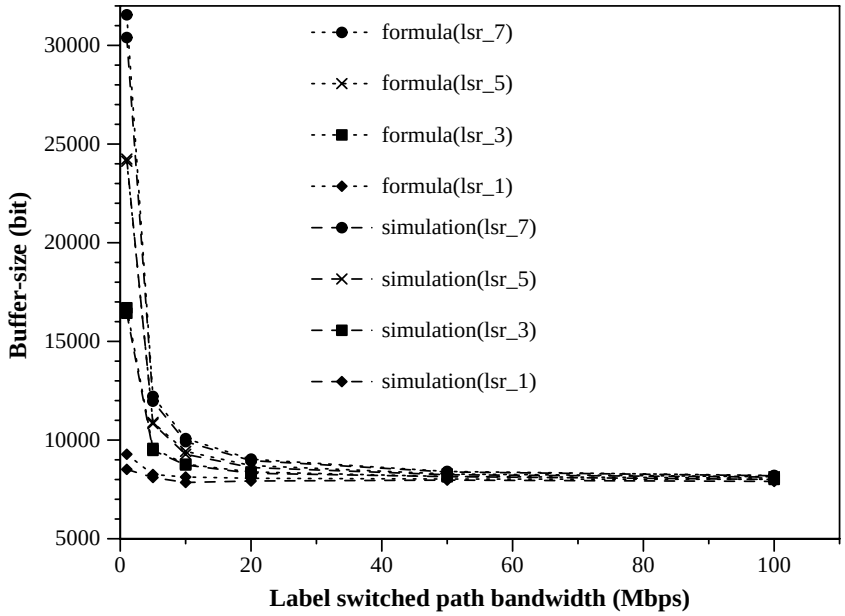

Figure 4.10 Comparison between formula and simulation results for ingress buffer with Vt_lsp=400k and Pkt_size=200bytes for different N and Bw_lsp

4.5.1 Validation of the results and qualitative analysis

The formula for buffer requirements for the ingress buffer (4.12) is:

$$B_{ingress} = 2 * T_{link} * V_{T\_lsp} * \left(\frac{(N-1)\, V_{T\_lsp}}{B_{W\_lsp}} + 1 \right)$$

we define α and β as:

$$\alpha = \frac{B_{W\_lsp}}{V_{T\_lsp}} \tag{4.15}$$

where $\alpha \geq 1$, and

$$\beta = \frac{T_{tran}}{T_{prop}} \tag{4.16}$$

where $\beta > 0$.

substituting the above definitions of α and β, and T_{link} from (4.2) in the formula for the ingress buffer requirements derived from our model, we get,

$$\mathbf{B_{ingress}} = \mathbf{2} \left(\frac{\mathbf{P}}{\alpha} \right) \left(\mathbf{1} + \frac{\mathbf{1}}{\beta} \right) \left(\frac{(\mathbf{N}-\mathbf{1})}{\alpha} + \mathbf{1} \right) \tag{4.17}$$

Considering the scenario used for simulation is: $V_{T\_lsp} = V_{W\_lsp}$, $P = 1600bits$, and T_{Prop} equals 1 msec and 0.1 msec, we get the α and β values to calculate the ingress buffer. The results agree with the simulation.

Note that in this condition Haskin's scheme also stores packets in the ingress LSR equivalent to the amount of packets circulating during the round trip time (i.e., $2 * T_{link} * V_{T\_lsp}$).

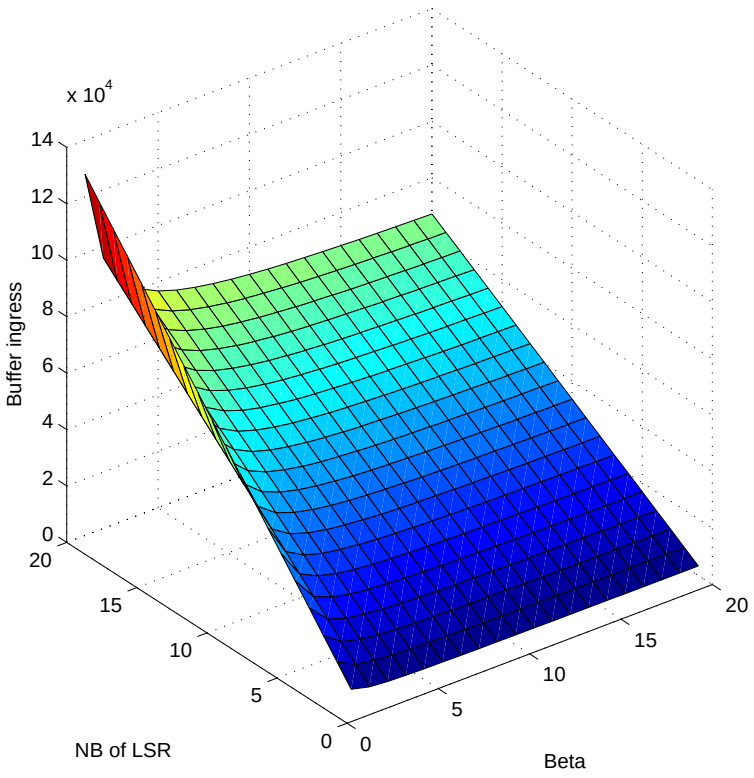

Figure 4.11 Behavior of ingress buffer

For $\alpha = 1$:

$$B_{ingress} = 2PN\left(1 + \frac{1}{\beta}\right) \tag{4.18}$$

When $\beta \to \infty$, it implies the propagation time tends to zero, so,

$$B_{ingress} = 2PN \tag{4.19}$$

The above result proves that with the propagation time zero the nodes are tight and the result depends only on the LSR number (N) and the packet size, which is proportional to T_{tran}.

When $\beta \rightarrow 0$, it implies the propagation time tends to ∞, so the $B_{ingress} \rightarrow \infty$. This is true because there are almost no packets circulating in the network.

The formula for the recovery period from (4.7) is:

$$T_{recovery} = T_{link}\left(N + 2 + 2\left(N-1\right)\frac{V_{T\_lsp}}{B_{W\_lsp}}\right) \tag{4.20}$$

substituting α and β, and T_{link} in $T_{recovery}$, we get,

$$\mathbf{T_{recovery}} = \mathbf{T_{tran}}\left(\mathbf{N + 2 + 2}\left(\frac{\mathbf{N-1}}{\alpha}\right)\right)\left(\mathbf{1} + \frac{\mathbf{1}}{\beta}\right) \tag{4.21}$$

For $\alpha = 1$:

$$T_{recovery} = 3N * T_{tran}\left(1 + \frac{1}{\beta}\right) \tag{4.22}$$

When $\beta \rightarrow \infty$, it implies the propagation time tends to zero, so the recovery time depends on the number of LSRs and the packet transmission time, T_{tran}.

$$T_{recovery} = 3N\frac{P}{B_{W\_lsp}} = 3N * T_{tran} \tag{4.23}$$

When $\beta \rightarrow 0$, it implies the propagation time tends to ∞, so the recovery time tends to ∞.

For Haskin's scheme the recovery period is the sum of the time taken by the first packet switched-over by the alert LSR to arrive at the ingress LSR (T_{link}) and the time taken by the last packet sent before the ingress LSR received the first packet through the backward LSP to travel from the ingress LSR and return back to this LSR, which is equal to $2 * T_{link}$ (i.e., time from ingress-alert-ingress).

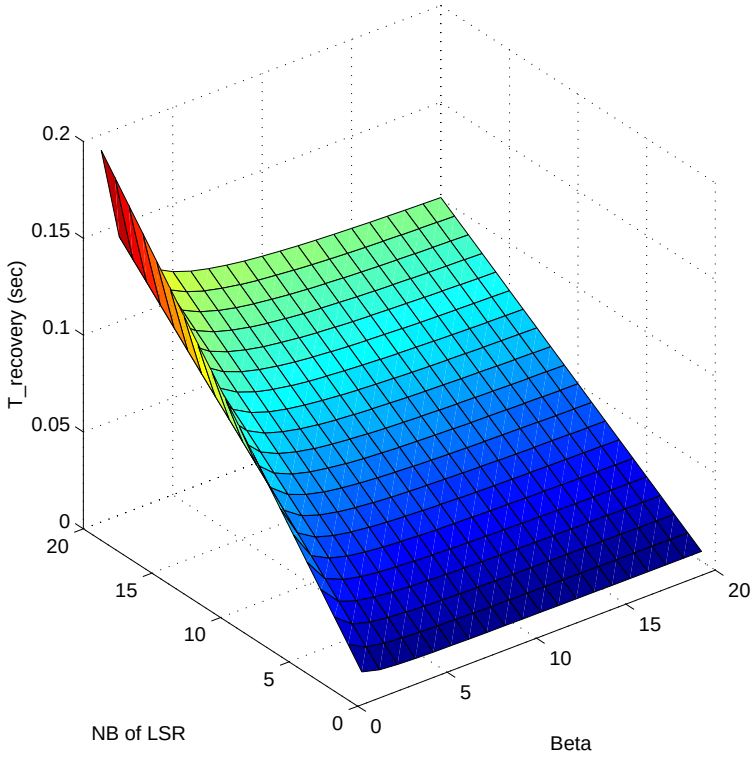

Figure 4.12 Behavior of recovery time

$$T_{recovery\_haskin} = 3 * N * T_{link} \tag{4.24}$$

With α and β,

$$T_{recovery\_haskin} = T_{tran} * 3N \left(1 + \frac{1}{\beta}\right) \tag{4.25}$$

Considering the simulation scenario: $V_{T\_lsp} = 400kbps$, $V_{W\_lsp} = 1Mbps$, $P = 1600bits$, and $T_{prop} = 10msec$ we get the α and β values to calculate the recovery time for Haskin's and our proposal. The results agree with the simulation.

4.6 SUMMARY

This chapter has presented a mechanism to perform Reliable and Fast Rerouting (RFR) of traffic in MPLS networks. Our method eliminates packet loss and packet disorder while improving the average delay time during the restoration period. This is achieved at a minimal cost for additional buffer space (memory) that is far outweighed by the benefits.

Apart from the buffer size, which is not very significant even for the worst case, the most interesting aspect is the linear behavior of our model relating B_{W_lsp}, V_{T_lsp}, P, d and N. This allows easy estimation of the buffer requirements for given bandwidths and QoS constraints.