3

FAST REROUTING MECHANISM

3.1 INTRODUCTION

Given that network topologies are never stable over time, rapid response to link failures and/or congestion by means of rerouting is critical. There are two basic methods for protected LSP recovery: (1) Dynamic rerouting and (2) Fast rerouting [SH02]. When the primary label switched path (LSP) encounters a problem due to link or node failure, the data that travels it needs to be rerouted over an alternative LSP. This is equivalent to using a new LSP to carry the data. The alternative LSP can be established after a protected LSP failure is detected, or it can be established beforehand in order to reduce the LSP switchover time [SH02]. The former option has a slow response in the rerouting function. The latter has a much better response time. In our proposal we use the fast rerouting technique with pre-established alternative LSPs to protect the packets travelling in the protected LSP.

Fast rerouting uses pre-established alternative LSPs. We focus on improving current mechanisms for fast rerouting proposed by Haskin. The objective is to improve packet delay during the restoration period and minimize packet disorder.

In [HK00], when a failure is detected in the protected LSP, the traffic is sent backwards to the ingress LSR using a pre-established LSP (*backward LSP*). When the ingress LSR receives the first packet from the backward LSP, the traffic flow for the protected LSP is redirected to the alternative LSP that was established previously between ingress and egress LSRs following a global repair strategy (Figure 3.1).

In Figure 3.1, the ingress and egress nodes are LSR0 and LSR4 respectively. 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 backward LSP will include the nodes 3, 2, 1 and 0. The *alternative LSP* will be formed by the LSR nodes 0, 5, 6, 7, 8 and 4.

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 backward LSP (Figure 3.1). The first packet that is sent back is used as a fault detect notification. Until the fault notification arrives at the ingress LSR, packets are sent via the already broken

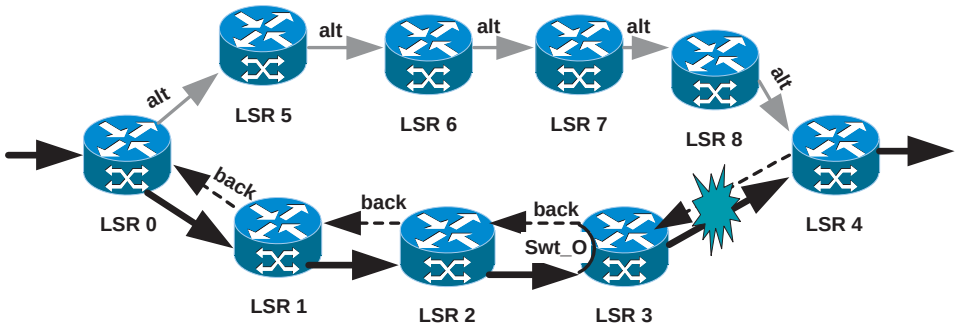

Figure 3.1 Scheme for alternative LSP to handle fast rerouting during the restoration period (back: backward LSP; alt: alternative LSP)

protected LSP. These packets will experience a two-way delay while traversing the backwards loop from the ingress LSR to the alert LSR and back to ingress LSR.

The restoration process ends when the last packet the ingress LSR sent through the already broken protected LSP comes back through the backward LSP. Then the protected and backward LSP are released.

As we explain in Section 2.3.4, an important drawback in this scheme is the delay involved in detecting the first packet that is sent back from the alert LSR to the ingress LSR, plus the delay for the subsequent packets sent along the broken LSP to return to the ingress LSR. Further, this approach also introduces data packet disordering during the LSP rerouting process. This is because once a fault is detected, the ingress node merges the newly incoming traffic and the packets coming back from the point of failure when sending them along the alternative LSP. The problem of this scheme concerning packet loss is left to be addressed in Chapter 4.

3.2 PROPOSED MECHANISM

Our proposal follows the principle described in [HK00] for setting both an alternative LSP and a backward LSP. In this section we address the drawbacks of Haskin's [HK00] proposal with respect to round-trip delay and packet disorder during restoration. In the description, upstream and downstream refer to the direction of traffic in the protected LSP.

In our proposal, when a fault is detected by an LSR, a switchover procedure is initiated and the packets are sent back via the backward LSP. As soon as each upstream node on the backward LSP detects these packets, they start storing the incoming packets (on the primary or protected path) in a local buffer. This avoids the unnecessary forwarding of packets along the broken LSP. Furthermore, the last packet forwarded before initiating storage is tagged in order to be identified on its way back. By doing this, we are able to preserve the ordering of packets when it is time for each

intermediate node to send back its stored packets. We use one of the *Exp* field bits of the MPLS label stack [RTF+01] (see Figure 1.3) for the purpose of tagging and thereby avoid any overheads.

Each LSR on the backward LSP successively sends back its stored packets when it receives the tagged packet. When all packets are returned 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 are now sent via the alternative LSP. Note that global ordering of packets is preserved during the whole process.

The detailed algorithm along with the state machine diagram is presented in the next section.

3.3 ALGORITHM DESCRIPTION

Before getting into the details of our algorithm, it is important to take into account the modification we made in the label information base-forwarding table (LIB). We introduce a new field called "*STATUS*" in the LIB, and manage five states in this field. They are: NORMAL, FAULT_DETECT, ALTERNATIVE_DETECT, STORE_BUFFER and SEND_BUFFER.

NORMAL: This state corresponds to the normal operation condition. It means no fault is detected on the protected LSP: the LSR continues working in the normal condition.

FAULT_DETECT: As the name implies, this is the state to indicate the condition of a faulty link. The node becomes an alert LSR. When a tagged packet is forwarded through the backward LSP, or after a certain time depending on the implementation, the LSR removes the LIB entry.

ALTERNATE_DETECT: This state is in charge of notifying the incoming protected LSP packets of the failure after receiving a packet from the backward LSP. It waits for the first packet coming through the protected LSP, which will be tagged and transmitted.

STORE_BUFFER: This state is in charge of indicating the need to store packets travelling in the protected LSP after detecting the presence of packets through the backward LSP and sending the tagged packet to downstream LSRs. This avoids the unnecessary trip of packets downstream and back again.

SEND_BUFFER: The state in which the stored packets (i.e., packets stored during the STORE_BUFFER time) will be drained from the buffer to the alternative or backward LSP (if it is the ingress LSR or an intermediate LSR, respectively). This state is activated when the tagged packet is received at the ingress or at each intermediate LSR through the backward LSP.

Figure 3.2 presents the state machine diagram of the proposed algorithm. The ingress LSR forwards packets to the alternative LSP while the intermediate LSR forwards packets through the backward LSP. Though the state machine diagram by itself is a formal description, a detailed explanation of the process follows. Given this brief functional explanation of each state, we proceed to describe the whole algorithm of our proposal.

Note that traffic belonging to other LSPs going through the broken link is lost. Only protected LSP traffic is switched-over.

Once a failure along the protected LSP is detected, the protected LSR that detects the fault (alert LSR) performs the switchover procedure (LSR3 in Figure 3.3). 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 for the protected LSP is changed from NORMAL to FAULT_DETECT (Figure 3.2).

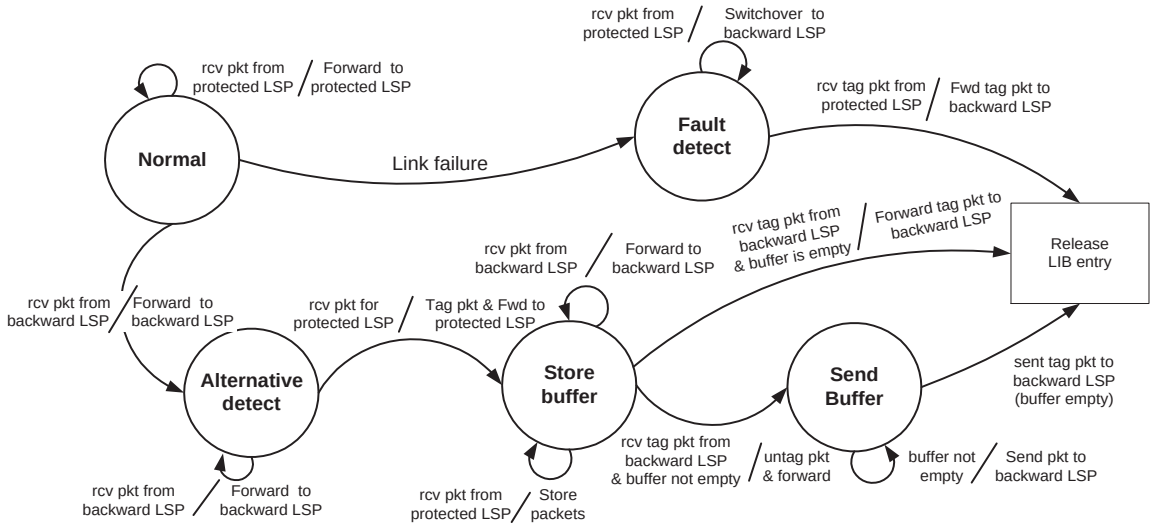

Figure 3.2 State machine diagram for intermediate LSRs

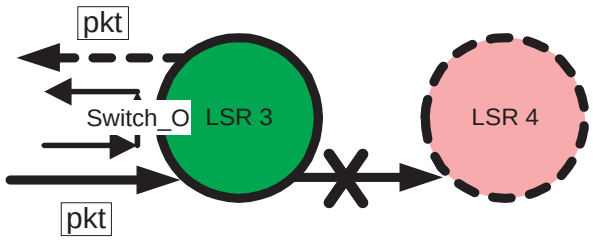

Figure 3.3 FAULT_DETECT and Switchover

The intermediate upstream LSR, in this case LSR2 (Figure 3.4 and Figure 3.1) receives these reversed packets from LSR3 through the backward LSP. When it receives the first packet through 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 3.2). Then, the first packet received from the protected LSP sees this entry as ALTERNATIVE_DETECT. This indicates that there is a link problem somewhere in the protected LSP. This packet must then be tagged as the last packet from this LSR (LSR2) and forwarded normally downstream and the LIB entry status must be changed from ALTERNATIVE_DETECT to STORE_BUFFER

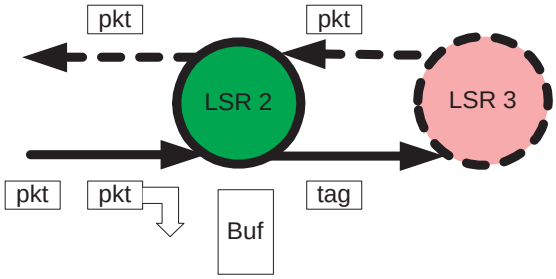

Figure 3.4 Intermediate LSR ALTERNATIVE_DETECT, Tag and STORE_BUFFER

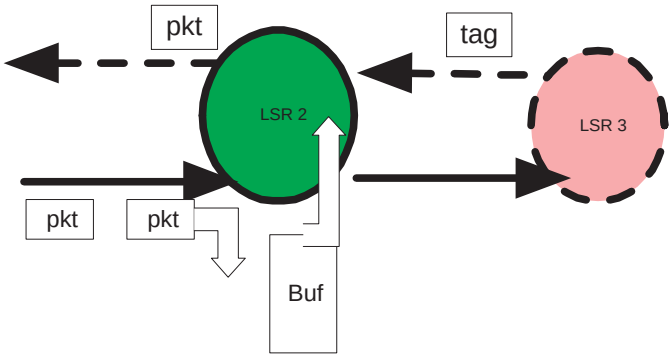

Figure 3.5 In intermediate LSR Tagged packet received and SEND_BUFFR

(Figure 3.2 and Figure 3.4). The next packets in the protected LSP will be stored in the buffer because they will find the link status in LIB as STORE_BUFFER. This continues until the tagged packet is received back through backward LSP.

When an LSR receives a packet from the backward LSP it checks if the received packet is a normal backward packet or a *Tagged* packet. The tagged packet received through the backward LSP is used as a trigger to perform the drain of the stored packets from the local buffer. The LSR has to check if the tag bit of the packet received from the backward LSP is set (1) or not (0). If the comparison result is false (i.e., the tag bit of the packet is set to 0 -normal backward packet-) the packet will

be forwarded using the normal swapping operation. Otherwise, the LSR knows that no more packets are expected from the backward LSP.

At this time, depending on the local buffer condition, the LSR takes one of the following two actions:

i) if the buffer is empty the LSR forwards the tagged packet on the backward LSP without any change in the tag bit. Note that the upstream LSR sends at least one tagged packet to the downstream LSR through the protected LSP after receiving the first packet from the backward LSP, and waits for this tagged packet to return through the backward LSP. Buffer empty means the unique packet sent by the upstream LSR is this tagged packet. For this reason it must be sent directly to the upstream node. Then the LIB entry from the LIB table is released.

ii) if the buffer is not empty 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 (Figure 3.5), and then when the buffer is empty, it releases the LIB entry from the LIB table Figure 3.2. Note that SEND_BUFFER finishes its process when it sends the tagged packet through the backward LSP. With this condition, no more packets on the protected LSP can use the output interface corresponding to this LSP before the fault was detected. Finally, the label associated with the protected LSP is removed. This process is repeated at every LSR until reaching the ingress LSR. In the case of the backward LSP, as it can carry other traffic on it (from egress to ingress) the LIB entry release process is done by the normal release procedure. The only thing needed is to release its resources reserved for the protected LSP.

The ingress LSR, unlike the intermediate LSRs, has the responsibility to divert or detour the incoming traffic (i.e., traffic entering the MPLS domain) from the failed primary or protected LSP to the previously established alternative LSP from ingress LSR to egress LSR (end-to-end in the MPLS domain) when it receives the tagged packet. While the intermediate LSR returns the traffic to the backward LSP, when

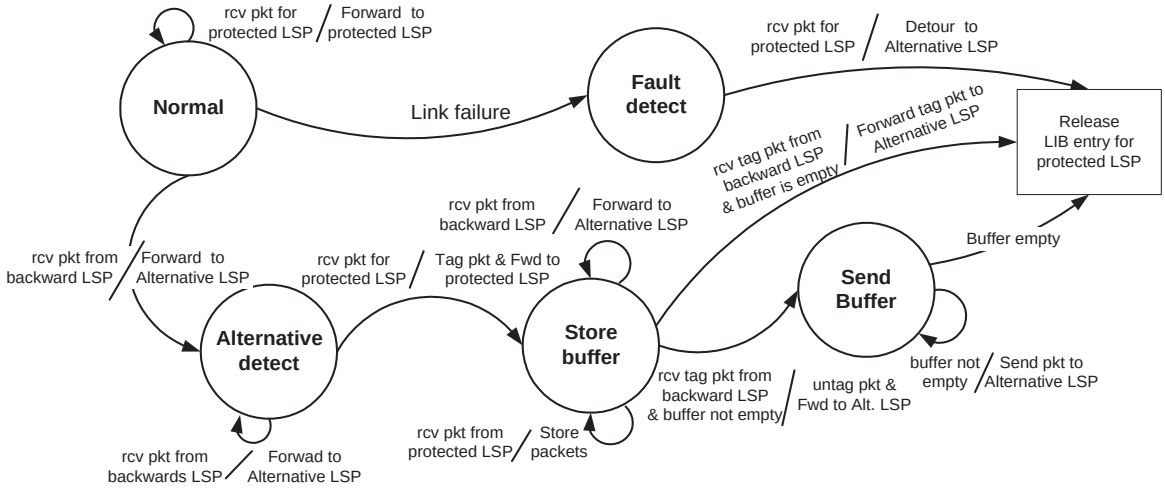

Figure 3.6 State machine diagram for ingress LSR

the ingress LSR receives the tagged packet it drains all its stored packets like any intermediate LSR and when it finishes, starts redirecting the incoming traffic directly to the alternative LSP. Figures 3.7 and 3.8 show the operation at the ingress LSR. When the incoming traffic is transited to the alternative LSP without passing through the buffer, the full restoration process terminates Figure 3.8.

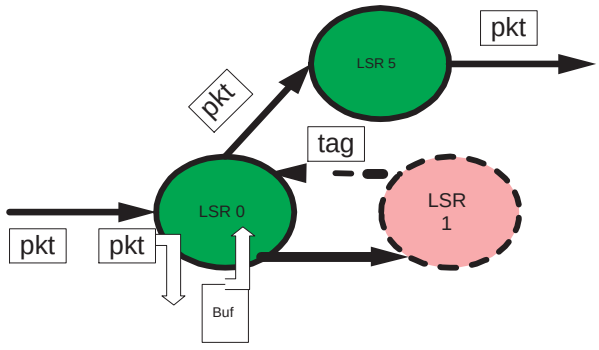

Figure 3.7 In ingress LSR Tagged packet received and SEND_BUFFER

Our proposal avoids sending packets downstream once any intermediate LSR between the ingress LSR and the point of failure detects packets on the backward LSP. This

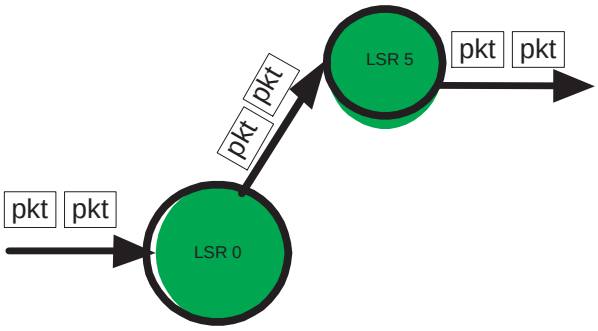

Figure 3.8 Restoration period terminates

reduces considerably the average delay of packets travelling in the protected LSP during the detection of the fault in a distant LSR.

3.3.1 Description of LIB table management

In Figure 3.9 we give a graphical description of the sequence of changes in the label information based forwarding table (LIB) during the recovery period including the field added by our proposal, link status. As you can see, for the purpose of simplicity after receiving the tagged packet we remove the entry corresponding to the backward LSP. This is because we assume that the protected and the backward LSPs belong to the same physical link, or that the backward LSP does not carry other traffic using this node as a merging point. When this is used for other traffic from the egress LSR to the ingress LSR using another physical link or a merging point for other LDP peers, or simply for traffic reverting purposes, it is left for the normal LSP release procedure and the mechanism can only decide whether or not to release the reserved bandwidth for the protected LSP depending on the conditions. Note the link status for this entry remains unchanged (normal).

Although the case when the ingress LSR (LSR0) as alert LSR detecting the failure is not present explicitly, it is easy to infer from Figure 3.9. If the ingress LSR detects the failure, it changes the link state for the protected LSP (LIB_entry 2) to fault_detect,

drains the buffer, and detours the traffic from interface 1 to interface 4 by pushing label 9 on to the packet (LIB_entry 1).

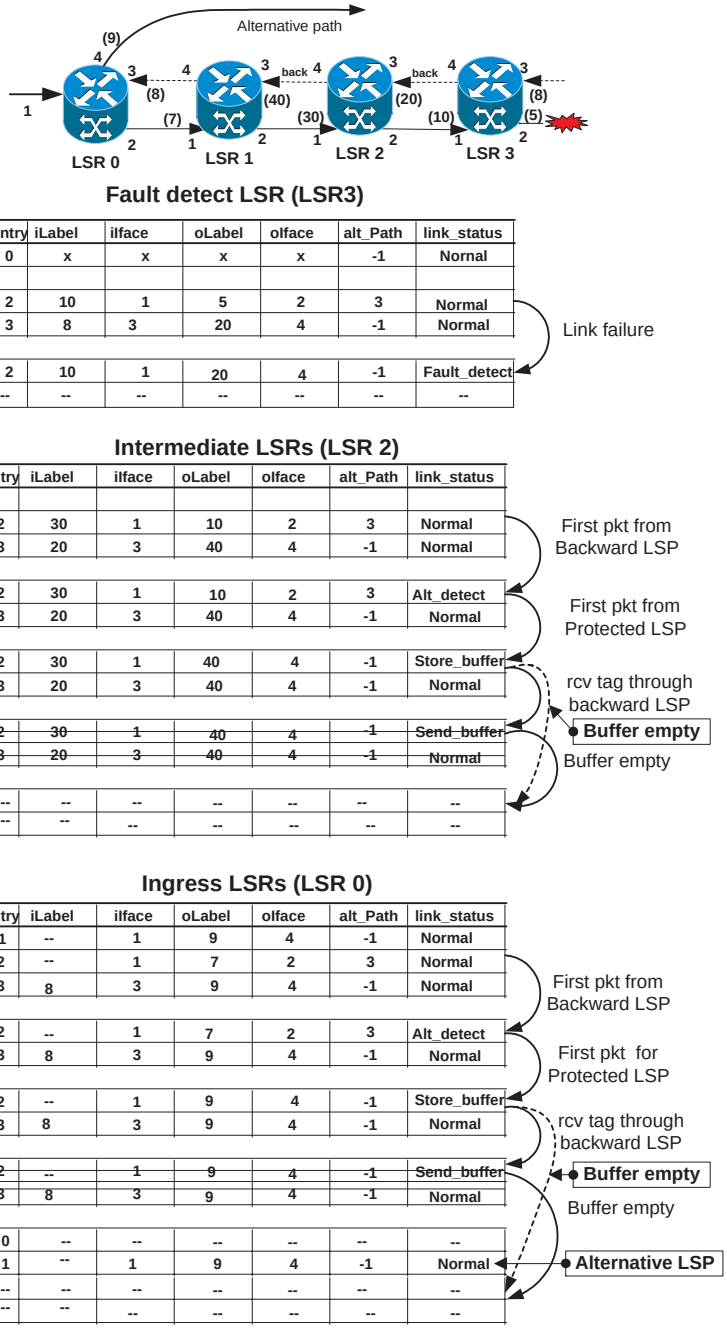

Figure 3.9 LIB entry (label forwarding table), we assume the backward LSP is not carrying other traffic

3.4 RESULTS

The simulated scenario is the one shown in Figure 3.1. The simple network topology with a protected LSP and a pre-established end-to-end alternative LSP is used. We extend the simple network topology for different numbers of intermediate LSRs in the protected LSP, yielding different sizes of LSPs (i.e., LSPs with 5 (e.g., Figure 4.1), 10 and 15 LSRs and with alert LSR at 3rd (Figure 4.1), 8th and 13th respectively) to analyze and compare the behavior of both proposals.

We modified part of the MNS source code to satisfy our particular requirement for the simulation of both mechanisms (ours and Haskin's [HK00]). Note that the simulation platform for these proposals was the same in order to be able to compare the simulation results. We compared our results for the disordered and dropped packets during path restoration with the ones published in [GW01b], thus validating our modified simulator.

In order to compare the results we use CBR traffic flow and a UDP agent with the following characteristics: packet size = 1600 bits, source rate= 400Kbps, burst time=0 and idle time =0, $T_{prop} = 10msec$, and $B_{W\_lsp} = B_{W\_back} = B_{W\_atl} = 1Mbps$ as defaults.

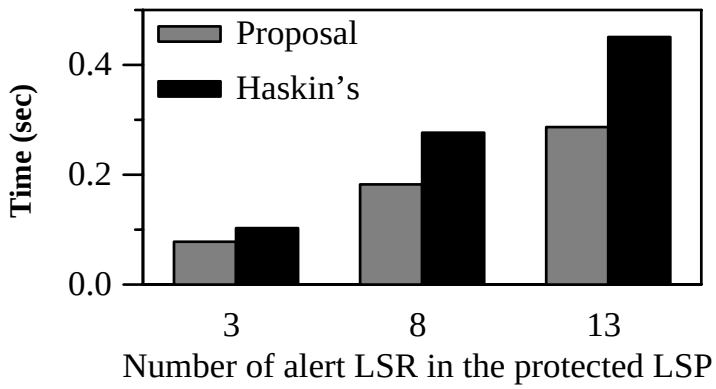

Figure 3.10 Restoration time to alternative LSP

Figure 3.10 shows the overall restoration period for both proposals for different positions of the alert LSR (number of the LSR) within the protected LSP. Note that the position of the alert LSR coincides with the number assigned to the LSR on the protected LSP. We assume the worst case in the sense that the failure occurs in the last link. Time is computed from the detection of the fault until the protected LSP is completely eliminated. The proposed mechanism provides a significant improvement of the path restoration period. Time reductions of 24.12%, 34.05% and 36.37% for the 3rd, 8th, and 13th alert LSRs on the protected LSP respectively are achieved.

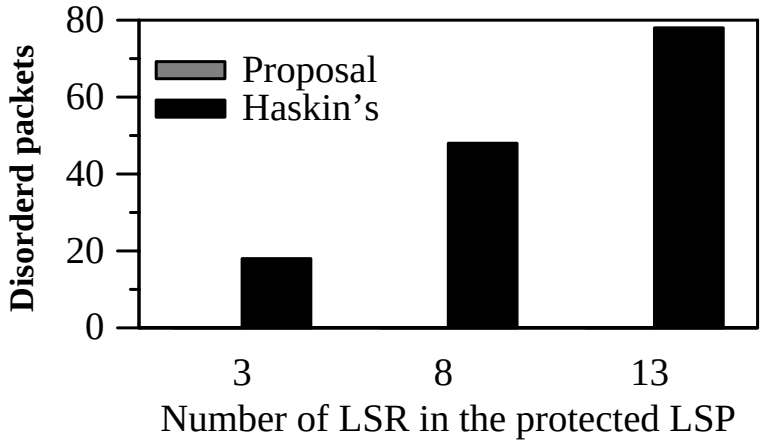

Figure 3.11 Number of disordered packets

Figure 3.11 confirms that the proposed mechanism avoids packet disordering while the restoration is in process.

In Figure 3.12 we can observe the simulation results for the restoration time and different bandwidths given the same traffic. The bigger the BW is, the faster the transmission, and smaller the slope of the line. That means shorter recovery periods are obtained with a faster LSP.

Figure 3.13 varies the packet size for a given bandwidth (5Mbps). In both cases, the number of intermediate LSRs were varied from 1 to 7. As can be seen from both figures, the reduction in restoration time is significantly better for the proposed mechanism for longer protected LSPs (i.e., LSPs with a greater number of nodes).

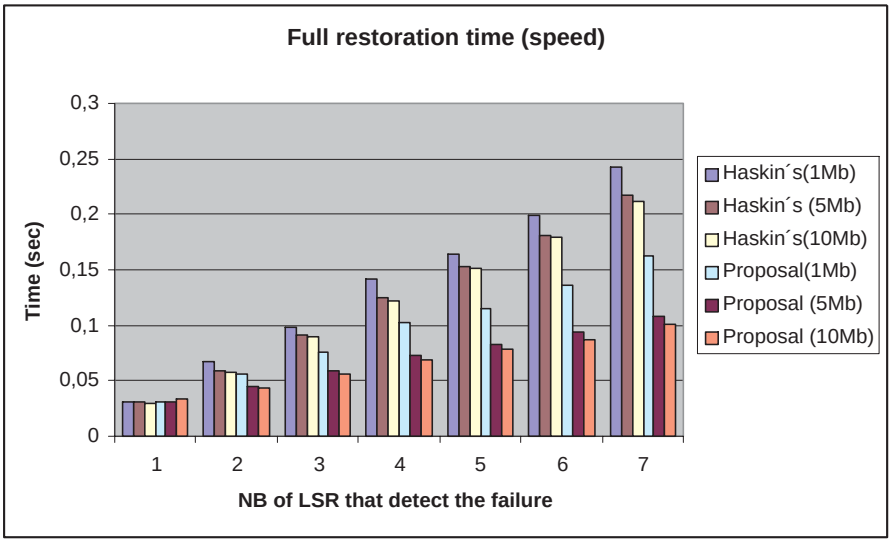

Figure 3.12 Restoration delay for 1600 bits packet size

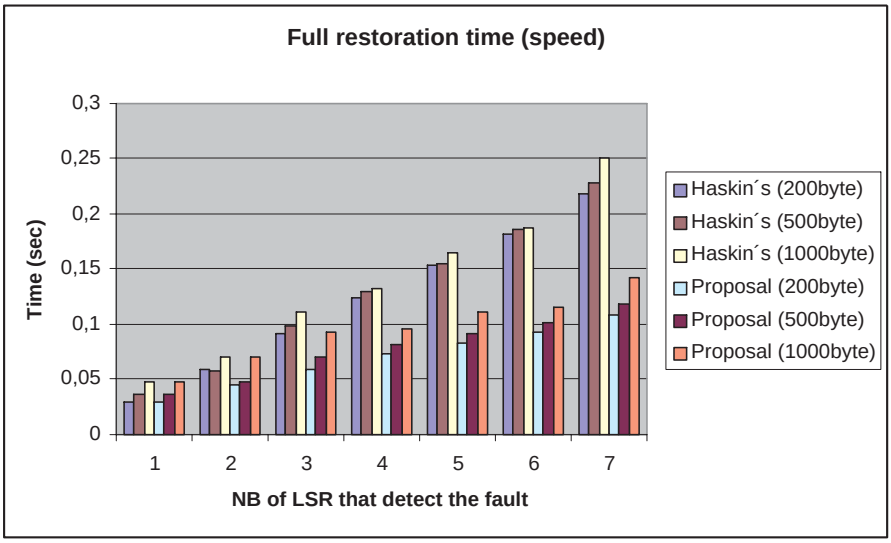

Figure 3.13 Restoration delay for 5Mbps LSP

Shortening the restoration period and the average packet delay during restoration, together with preserving packet sequence, minimizes the effect of a fault and leads to an improvement in the end-to-end performance.

3.5 SUMMARY

This chapter has presented a mechanism to perform fast rerouting of traffic in MPLS networks. We proposed a method to avoid packet disorder and improve the packet average delay time during the restoration period.

A decentralized mechanism involving all LSR from the ingress LSR to the alert LSR is described using a state diagram. Implementation details are also considered in the proposal.

In summary our proposal has the following advantages:

1. Improves the average latency (average packet delay).
2. Avoids packet disorder.
3. Improves end-to-end performance (overall performance).
4. Has a shorter restoration period than Haskin's proposal (i.e., faster network resources release).

In addition to recovery from failures, the proposed mechanism can be used for quality of service (QoS) provision. Once a given LSR detects congestion or a situation that leads to a Service Level Agreement (SLA) or QoS agreement being violated, it may start a fast reroute of a protected LSP that shares the link.

An LSP rerouted due to congestion may experience a slight increase in delay for a short period but no packets will be lost or disordered. Unlike the problem of failure in the link or node, the congestion problem gives administrators time to maneuver the rerouting of packets towards the alternative path. To extend the proposed mechanism to the congestion problem only a guarantee that the LSR is aware of the congestion is needed - just as in the case of a link fault. If this condition is satisfied, the flow can be diverted to the alternative path during a congestion situation.