A Reliable QoS Provision and Fast Recovery Method for Protected LSP in MPLS-based Networks

Lemma Hundessa and Jordi Domingo Pascual
Departament d'Arquitectura de Computadors
Universitat Politècnica de Catalunya (UPC)
C/ Jordi Girona 1-3, 08034. Barcelona, Spain
E-mail:{hundessa,jordid}@ac.upc.es

We present a reliable QoS provision and fast recovery method for protected traffic when there is a link/node failure or congestion problem in Multiprotocol Label Switching (MPLS) networks. This proposal is able to guarantee rigorous QoS criteria for high-priority data traffic by *eliminating packet loss and disorder, and minimizing packet delay.* We show that this directly translates into an improvement of the important quality measure attributes such as performance, reliability and fault tolerance in the MPLS-based networks. We use a predefined, alternative Label Switched Path (LSP) in order to restore traffic (Protection Switching or fast rerouting). A theoretical model is formulated for the failure scenario in a protected LSP segment and we validate it through simulations using the MPLS Network Simulator (MNS). The results of further simulations show that our mechanism is able to completely eliminate packet loss and disorder while reducing the Full Restoration Time. The potential cost in terms of buffer requirements - an important issue of our proposal - was also studied. We show that even for the worst case, buffer requirements are well within justifiable limits for guaranteeing QoS for high-priority data traffic in protected LSPs. The combination of these improvements helps to minimize the effects of link failure. This facilitates satisfying rigorous QoS requirements, increasing throughput of critical traffic, rapid release of network resources and enhancement of the end-to-end performance of MPLS networks.

Keywords: MPLS, Label Switched Path (LSP), QoS, Switchover, Backward LSP, Alternative LSP, Label Switching Router (LSR).

1 Introduction

The introduction of Multi Protocol Label Switching (MPLS) as part of the Internet forwarding architecture will contribute significantly to traffic engineering[1],[2],[3]. Some components of the MPLS traffic engineering solution are: Label Switched Paths (LSPs), appropriate path discovery, traffic assignment to paths and fast response to topology changes.

Given that network topologies are never stable over time, rapid response to link failures and/or congestion by means of rerouting is critical. This is even more important for high-priority data traffic that has rigorous Quality of Service (QoS) requirements. To provide correct QoS, it is not sufficient to only establish the protected LSP, it is also necessary to guarantee it for the duration of the session. There are two basic methods for LSP recovery: (1) Rerouting and (2) Fast rerouting [4]. The former works by establishing an alternative Label Switched Path or LSP segment on-demand, after the occurrence of a fault. Fast rerouting uses pre-established alternative LSPs or LSP segments. Fast rerouting from a network failure has been recognized as a key component part to provide service continuity to end users. We focus on improving current mechanisms for *Reliable QoS and Fast Recovery* (RFR).

In our previous work [5], 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, We found that for critical services (important traffic from premium customers) will be affected by packet loss. As a consequence bad performance and degradated service delivery will be experienced. Our present scheme (RFR) porpose a novel recovery algorithm with small amounts of 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 the premium traffic.

2 Fast rerouting performance constraints

Fast rerouting or Protection Switching [4] uses pre-established LSPs or LSP segments. When a fault is detected, the protected traffic is switched over to the alternative LSP(s). Setting pre-established alternative paths, results in a faster switchover compared to establishing new, alternative paths on-demand [4], [6], [7], [8].

Fast rerouting can be accomplished by protection mechanisms that are activated locally or that are global in scope. Local repair uses an alternative Label Switched Path (LSP) that serves as a bypass from the point of protection to the next LSR node or to the destination. The techniques proposed for local repairs in MPLS networks are splicing and stacking[8]. Global repair is activated on an end-to-end basis. That is, an alternative LSP is pre-established from ingress to egress nodes of the path to be protected. Our proposal combines both these techniques.

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[5] has addressed the last two mentioned factors. Up to now, packet loss due to node or link failure was considered to be ‘inevitable’ [4]. 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 significantly affects the throughput of TCP traffic due to the startup behavior (slow-start) of TCP.

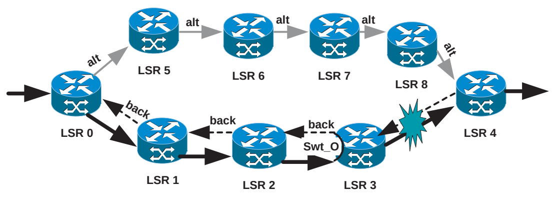


Figure 1: Simulation scenario

The main motivation of this work is to show that improving and/or eliminating the above-mentioned constraints will enhance significantly the throughput of all types of data traffic and the overall performance of the network.

It is important to note that the objective of the present paper is focused to provide and guarantee QoS for critical tarffic carried by protected LSP in MPLS network. Note that, not all LSPs are protected.

3 Proposed mechanism

The proposed mechanism is based on our previous work [5]. In Fig. 1, the ingress and egress nodes respectively are LSR0 and LSR4. The protected LSP is formed by the LSR nodes 0-1-2-3-4. 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 3-2-1-0 (we call it the *backwards LSP*) What we name as the *alternative LSP* will be formed by the LSR nodes 0-5-6-7-8-4. 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 newly activated *backward LSP* (Fig. 1). The first packet that is sent back is used as a fault-detect notification.

We assume that the backwards and alternative LSPs have already been set-up [5]. Changing to an alternative path is the responsibility of the routing control process (e.g. Label Distribution Protocol (LDP)[9] , extension of RSVP protocol [10]) and hence, is not within the scope of our study.

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 along the protected path. The maximum size of this buffer needs to be about twice the number of packets that can circulate in a given link of the protected LSP. This is so because the failure can occur either on a link or at a node. If the link fail , we would potentially lose only the packets occupying the link from LSR3 to LSR4 (Fig. 1). If node LSR3 fails, packets on both links to the node will have to be recovered.

3.1 Behavior of the Node that detects the failure

When a fault is detected by a 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 protected LSP to backwards LSP. Note that this node has *copies of packets* that were dropped from the faulty link/node and hence there is *no packet loss.*

3.2 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 is stored in its buffer. The next packet coming in from the protected LSP will be tagged and forwarded via the protected LSP. All subsequent packets that arrive at this node along the protected path are stored in its buffer without being forwarded [5]. 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 backwards LSP.

3.3 Role of tagging in eliminating disorder of packets

When a node detects the packet it tagged (the last packet sent onward 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 the stored packets. By doing this, we are able to preserve the ordering of packets. We use one of the *Exp* field bits of the MPLS label stack [11] for the purpose of tagging and thereby avoid 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 of 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, has a shorter restoration period than Haskin's proposal [5] and packet loss has been eliminated.

4 Algorithm description

Fig. 2 presents the state machine diagram of the proposed algorithm (RFR). Though the state machine 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 protected LSP: normal, fault detect, alternate detect, store buffer and send buffer.

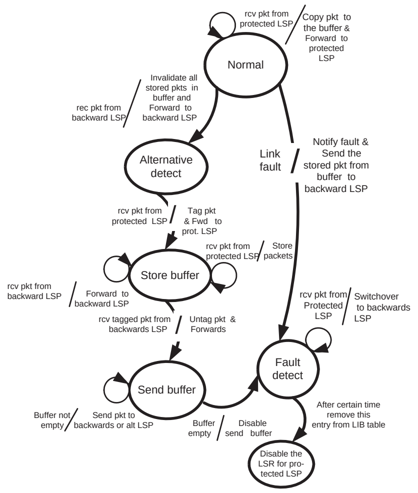

Normal
rcv pkt from protected LSP / Copy pkt to the buffer & Forward to protected LSP
rec pkt from backward LSP / Invalidate all stored pkts in buffer and Forward to backward LSP
Link fault / Notify fault & Send the stored pkt from buffer to backward LSP
Alternative detect
rcv pkt from protected LSP / Tag pkt & Fwd to prot. LSP
rcv pkt from protected LSP / Store packets
Store buffer
rcv pkt from backward LSP / Forward to backward LSP
rcv tagged pkt from backwards LSP / Untag pkt & Forwards
rcv pkt from Protected LSP / Switchover to backwards LSP
Fault detect
Send buffer
Buffer not empty / Send pkt to backwards or alt LSP
Buffer empty / Disable send buffer
After certain time remove this entry from LIB table
Disable the LSR for protected LSP

Figure 2: RFR state machine diagram.

Once a failure along the protected LSP is detected, the protected LSR that detects the fault performs the switchover procedure (LSR3 in Fig. 1.) This procedure consists of a simple label swapping operation from protected LSP to backwards 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 (<input_label, output_interface> corresponding to this forwarding entry table) is changed from NORMAL to FAULT_DETECT (Fig. 2). It then begins to drain all the packets stored in its buffer - i.e send it back along the backward LSP. Any incoming packets on the protected LSP are also sent back

The immediate upstream LSR, in this case LSR2 (Fig. 1) receives these reversed packets from LSR3 through the backwards 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 (Fig. 2). Additionally, it invalidates all data in its buffer. The next, immediate 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. So, this packet is 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 (Fig. 2). The subsequent packets coming in on the protected LSP will be stored in the buffer because it will find the link status as 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, it knows that no more packets are expected from the backwards LSP. 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 status changes to FAULT_DETECT (Fig. 2). Finally, the label associated with the protected LSP is removed. 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 requiering any additional algotithm for node failure restoration.

5 Derivation of 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 to the benefits that accrue from enhancing throughput and performance for high-priority QoS traffic. The size of the buffers required both at the ingress node and 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 one:

V_{T_lsp} : − Source rate (reference traffic),B_{W_lsp} : − LSP bandwidth, 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, and N : − Number of LSR that detects the fault (i.e. number of nodes of the backwards LSP excluding the ingress node).

According to our generalized network simulation model (Fig. 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 would be 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).

$$T_{switch\_over} = 3 * T_{link} \tag{1}$$

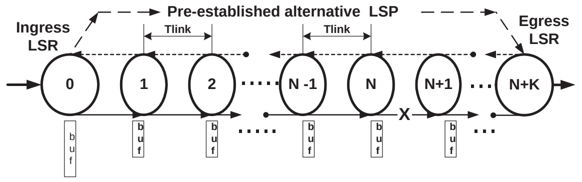


Figure 3: Model for equation. Solid line: Protected LSP; Dashed line: Backward LSP.

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.

$$T_{link} = T_{tran} + T_{prop} \tag{2}$$

The rest of the delays upto the point of restoration of traffic along the *alternate 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 (2) time taken for the last packet (the one that was *tagged*) to reach the next upstream node (T_{link}). 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:

$$T_{int\_buffer\_drain\_pkt} = \frac{2 * T_{link} * V_{T\_lsp}}{B_{W\_lsp}} \tag{3}$$

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

$$T_{int} = T_{int\_buffer\_drain\_pkt} + T_{link} \tag{4}$$

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. the all links operate at the same rate (B_{W_lsp}) and has the same propagation delay (d)), the sum delays in the intermediate LSRs is equal to $\sum_{i=1}^{N-1}(T_{int})_i = (N-1) * T_{int}$.

$$T_{recovery} = T_{fault\_detect} + T_{switch\_over} + (N-1) * T_{int} \tag{5}$$

We assume that the time to detect the fault by an LSR - $T_{fault\_detect} = 0$. Then the above equation becomes:

$$\mathbf{T_{recovery}} = \mathbf{T_{link}}\left(\mathbf{N} + \mathbf{2} + \mathbf{2}\,(\mathbf{N}-\mathbf{1})\,\frac{\mathbf{V_{T\_lsp}}}{\mathbf{B_{W\_lsp}}}\right) \tag{6}$$

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

$$\mathbf{T_{recovery}} = \mathbf{3} * \mathbf{N} * \mathbf{T_{link}} \tag{7}$$

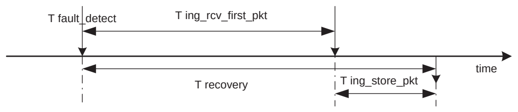


Figure 4: Graphical representation of times for ingress buffer calculation

5.1 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 since it receives the first packet switched over from the point-of-failure until it receives its own tagged packet. The time taken by the former is:

$$T_{ing\_rcv\_first\_pkt} = N * T_{link} \tag{8}$$

and the time at which the latter takes place is $T_{recovery}$ (Fig. 4). Therefore,

$$T_{ing\_store\_pkt} = T_{recovery} - T_{ing\_rcv\_first\_pkt} \tag{9}$$

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

$$B_{ingress} = T_{ing\_store\_pkt} * V_{T\_lsp} \tag{10}$$

and hence,

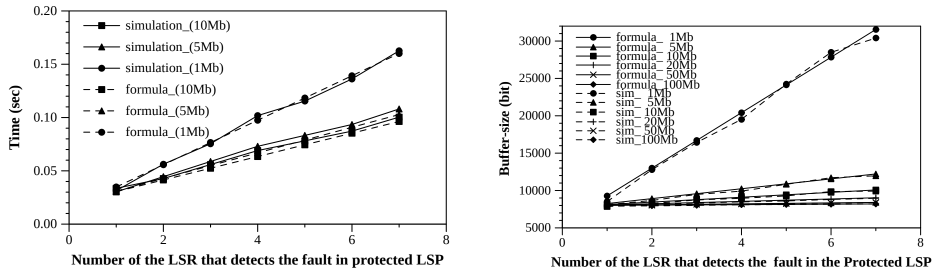
$$\mathbf{B_{ingress} = 2 * T_{link} * V_{T\_lsp} * \left(\frac{(N-1)\,V_{T\_lsp}}{B_{W\_lsp}} - 1\right)} \tag{11}$$

The required amount of buffer size in each intermediate LSRs during the restoration period is:

$$\mathbf{B_{intermediate} = 2 * T_{link} * V_{T\_lsp}} \tag{12}$$

6 Simulations and results

The simulation tool used to evaluate the proposal is an extension of the network simulator (NS) for MPLS networks called MPLS network simulator (MNS)[12]. The objective of the simulation is to validate the formula and to evaluate the behavior of TCP sources in MPLS-based networks comparing the proposed mechanism with the one proposed previously by Haskin[6].



a) b)

Figure 5: a) Recovery time for different LSP bandwidths. b)Ingress buffer size for Vt_lsp=400k and Pkt_size=200bytes for different LSP bandwidths and numbers of LSR (N).

We setup a FTP session over a TCP connection with packet size = 1000 bytes and the CBR traffic flow with the following characteristics: packet size = 200 bytes, source rate= 400K, burst time=0 and idle time =0. The simulated scenario is the one shown in Fig. 1.

The simple network topology with a protected and alternative LSP is used. We extend the simple network topology for different number of intermediate LSRs in the protected LSP. In this way, we vary the location of the node that detects a fault - thereby varying the size of the protected LSP within which rerouting takes place.

Parts of the MNS source code were modified to simulate both mechanisms (ours and Haskin's[6]) and the modified simulator was validated with previously published results for Haskin's method[13], [5].

The results based on the derived formula for the proposed model are plotted with the corresponding simulation results, for Full Restoration Time (Fig. 5a) and for the buffer size needed at the ingress LSR (Fig. 5b). These figures show that the analytical results are almost identical with the simulation results validating our analytical expression of the proposed mechanism (RFR). Observe that in both cases (Figs. 5a and 5b) for the B_{W_lsp} = *1Mb* the restoration time and the ingress LSR buffer requirement increases due to the fact that the transmission speed of the packets is low compared to 5Mb, 10Mb and above. The time required to reach the ingress LSR depends in the time speed. The restoration time basicaly depends on the transmission speed and the save applies for the buffer requirements at the ingress LSR. Fig. 6 compares the behavior of RFR and Haskin scheme. In Fig. 6a the difference in the sequence number of TCP segment received by the egress LSR is seen clearly for the same simulation time. Fig. 6b shows a more detailed view of the sequence number during the restoration period. Additionally, in Fig. 6b one can observe the perturbation caused by disorder of packets. Note that the time of link failure is 1.51 sec. Fig. 6 confirms that the proposed mechanism

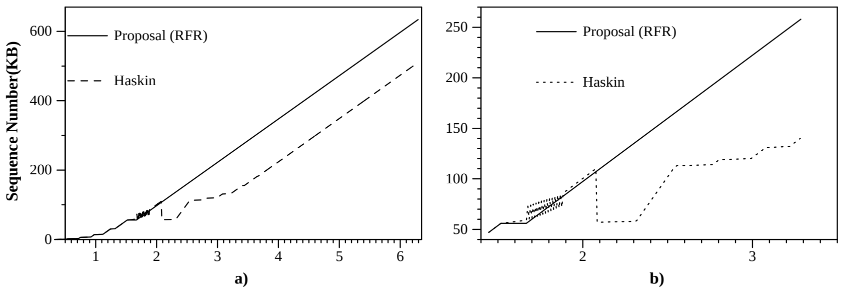


Figure 6: Behavior of TCP traffic for MSS of 1000byte, X-axis is Time(sec).

avoids packet loss and disordering. This benefit (advantage) is due to the use of the buffer, that avoids the loss of packets and therefore the penalty due to retransmission.

The plots in Fig. 7a and 7b correspond to the comparison between Haskin's scheme and RFR for the derived model and the simulation respectively for overall restoration period. We use Fig. 7b 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 when the fault is detected until the protected LSP is completely eliminated. Our proposed mechanism significantly improves the Full Restoration Time. A reduction of 24.6%, 27.9%, 29.8%, 31.7% and 33% for the 3rd, 4th, 5th, 6th and 7th node of the LSR that detects the fault on the protected LSP respectively are achieved. Note that the above percentage values correspond to a LSP bandwidth of 1Mbps. The improvements are greater as the bandwidth increasess.

In Fig. 8a we present the result concerning to the ingress node buffer requirement varying the B_{W_lsp}, distance (d) and N. Results show that for even a long-distance LSP the amount of buffer required at the ingress node is reasonable compared to the benefits provided by the RFR mechanism. In Fig. 8b we maintain the V_{T_lsp} and the distance(d) constant, and vary the

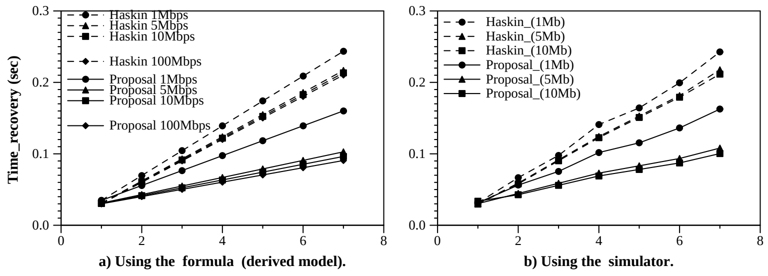


Figure 7: Restoration delay for 200bytes packet size for different LSP bandwidth, X-axis is number of LSR that detects the fault (N).

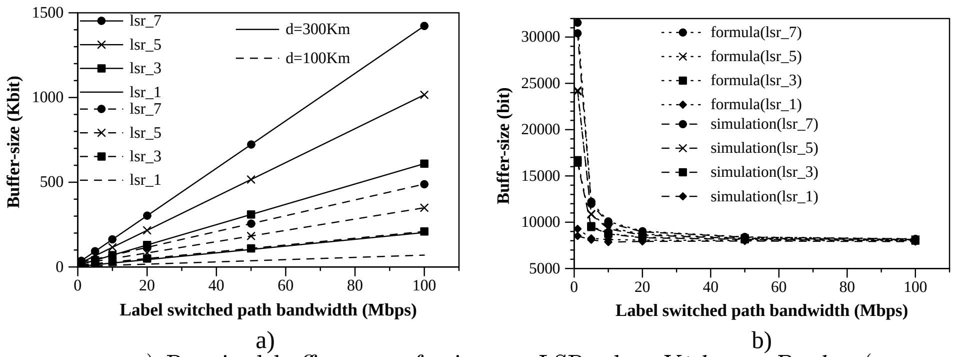


Figure 8: a) Required buffer space for ingress LSR when $Vt\_lsp = Bw\_lsp$ (worst case) and Pkt_size=200bytes for d=300Km and d=100Km varying the Bw_lsp and N. b) Comparison between formula and simulation results for ingress buffer with Vt_lsp=400k and Pkt_size=200bytes for different N and Bw_lsp

B_{W_lsp} and N. In this case, as we increase the B_{W_lsp} the effect of N in the required ingress buffer space becomes negligible. Note that in both cases we maintain the packet size (P) constant. So, one can observe the buffer space (memory) requirements for the implementation of our proposal for different conditions. We have plotted the buffer needs for the ingress node since it has to store packets for the longest period (waiting for all downstream nodes to drain their packets). Apart from the 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 would allow to estimate easily predict the buffer requirements for given bandwidths and QoS constraints.

7 Conclusions and future work

This paper presents a mechanism to perform Reliable QoS and Fast Recovery (RFR) of traffic in Multiprotocol Label Switching (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.

Our proposal has the following advantages:

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

The proposed mechanism provides a method for reliable 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 reliable QoS and fast recovery (RFR) of a protected LSP that shares the link.

The RFR due to its advantages turns the link/node failure problem into an equivalent problem of avoiding congestion in the protected LSP with the difference that for link congestion we have more time to manoeuvre the rerouting of packets to alternative path. To extend our mechanism to the congestion avoidance problem one only needs to guarantee that the LSR be aware of it - just as in the case of a link fault. If this condition is satisfied, we can divert the flow to the alternative path avioding a congestion situation.

Finally, the criteria for selecting alternative LSPs for QoS provision and the potential intermediate node(s) capable of "shortcut rerouting" from the point of failure to the pre-established alternative path is left for further study.

References

1. R. Callon, P. Doolan, N. Feldman, A. Fredette, G. Swallow, and A. Viswanathan. A framework for multiprotocol label switching. *Internet draft<draft-ietf-mpls-framework-05.txt>*, September 1999.
2. E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. *RFC 3031*, January 2001.
3. D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for traffic engineering over mpls. *RFC 2702*, September 1999.
4. V. Sharma, Ben-Mack Crane, S. Makam, K. Owens, C. Huang, F. Hellstrand, J. Weil, L. Andersson, B. Jamoussi, B. Cain, S. Civanlar, and A. Chiu. Framework for mpls-based recovery. *Internet draft<draft-ietf-mpls-recovery-frmwrk-03.txt>*, July 2001.
5. L. Hundessa and J. Domingo-Pascual. Fast rerouting mechanism for a protected label switched path. *Proceeding of the IEEE International Conference on Computer Communications and Networks (ICCCN2001)*, October 2001.
6. D. Haskin and R. Krishnan. A method for setting an alternative label switched paths to handle fast reroute. *Internet draft<draft-haskin-mpls-fast-reroute-05.txt>*, November 2000.
7. K.Owens, V.Sharma, S.Makam, and C.Huang. A path protection/restoration mechanism for mpls networks. *Internet draft<draft-chang-mpls-protection-03.txt>*, July 2001.
8. G. Swallow. Mpls advantages for traffic engineering. *IEEE Comunication Magazine*, pages 54–57, December 1999.
9. L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thoma. Ldp specification. *RFC 3036*, January 2001.
10. Daniel O. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow. Rsvp-te: Extensions to rsvp for lsp tunnels. *RFC 3209*, December 2001.
11. E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, and A. Conta. Mpls label stack encoding. *RFC 3032*, January 2001.
12. A. Gaeil and C. Woojik. Design and implementation of mpls network simulator (mns) supporting ldp and cr-ldp. *proceedings of the IEEE International Conference on Networks (ICON'00),*, September 2000.
13. A. Gaeil and C. Woojik. Simulator for mpls path restoration and performance evaluation. *http://flower.ce.cnu.ac.kr/˜ fog1/mns/index.htm.*, April 2001.