Influence of GMPLS Recovery Mechanisms on TCP Performance

Chris Develder, Didier Colle, Sophie De Maesschalck,

Mario Pickavet, Piet Demeester

Ghent University – IMEC, Department of Information Technology Sint-Pietersnieuwstraat 41, 9000 Gent (Belgium)

Email: {chris.develder, didier.colle, sophie.demaesschalck, mario.pickavet, piet.demeester}@intec.rug.ac.be

Abstract. Optical networks based on Wavelength Division Multiplexing (WDM) techniques are very likely to be omnipresent in future telecommunication networks. Those networks are deployed in order to face the steady growth of traffic, which is for a large part Internet related. In the resulting IP-over-WDM scenario, TCP/IP constitutes an important fraction of the traffic transported over these networks. As IP networks are becoming increasingly mission-critical, it is of the utmost importance that these networks (and hence the supporting transport networks) be able to recover quickly from failures such as cable breaks or equipment outages. To that end, several IP-over-WDM network scenarios and corresponding protection and restoration strategies have been devised. It is clear that some trade-offs will have to be made in order to choose an appropriate strategy. In this paper, we investigate the effects of such recovery actions on the behavior of TCP, being the ubiquitous protocol used by today's network users. We examine the influence of different parameters such as the speed of recovery actions, changing length of the routes followed by the client data (TCP flows), changes in available bandwidth, etc. Thereby, we focus on what the TCP end-users care about, i.e. the number of bytes transported end-to-end within a certain time interval.

Keywords: IP-over-WDM, TCP, (G-)MPLS, recovery, protection

1. Introduction

Telecommunication networks in recent years have faced an explosive traffic growth, mainly due to the popularity of the Internet [1]. For a couple of years already, the data traffic volume dominates (classical) voice traffic, and recent forecasts do not seem to predict a rapid slowdown of this greediness for bandwidth [2]. Communication networks will be more and more optimized for the dominant IP traffic, as TCP/IP is functioning as the convergence layer for practically all forms of end-user communication in today's data communication networks. It is foreseeable that it will continue to do this in tomorrow's multi-service networks, where IP-based applications such as voice, video and other multimedia applications will generate the necessary revenues (foreseen to outpace classical voice revenues). While some of these new applications tend to use the unreliable User Datagram Protocol (UDP), the reliable Transmission Control Protocol (TCP) today still is responsible for a major portion of the IP traffic [3, 4] and is used extensively by many of the so-called peer-to-peer (P2P) applications [5] that arose in the wake of the popular Napster, and are used for various purposes, ranging from file and knowledge sharing to distributed computing.

To cope with the large traffic volumes that such a multi-service network necessitates, wavelength-division multiplexing (WDM) technology has been devised. Fiber exhaust is currently solved by multiplying the capacity of a fiber by means of point-to-point WDM systems; a multiplexing technique that has proven to be very cost-efficient due to the economy of scale [6]. Current optical component technologies enable to introduce networking functionality in the WDM layer (by means of optical add-drop multiplexers and cross-connects), that laid the foundations for an Optical Transport Network (OTN). Automating (the configuration of) these networking functions will result in an Automatic Switched (Optical) Transport Network, currently under development in e.g. the ITU [7].

Due to the growing importance of IP traffic and the opportunities offered by WDM technology, many research activities are dedicated to bringing the two closer together, under the flag of IP-over-(D)WDM [8]. A trend observed today is to eliminate or reduce intermediate layers between IP and WDM. Generalized Multi-Protocol Label Switching (GMPLS) [9–11] promises to offer the necessary control plane "glue" to join them quasi directly.

As IP networks grow more and more mission critical, more stringent requirements are imposed on them: the support of service differentiation, introduction of quality of service (QoS) [12], but also the ability to survive network failures [13], all are important research topics. The survivability of a network is usually guaranteed by a set of restoration and/or protection schemes. A question that arises when comparing those schemes is what the effect of such protection actions is on the dominant client layer: TCP (according to recent measurements reported by [4], it accounts for 85% of the packets, and more than 95% of the bytes transmitted on some transatlantic links). This is the question we will focus on in this paper. Evidently, the extent to which TCP withstands the actions taken by a protection mechanism is only one of the touchstones to judge it on. Other criteria include the amount of control traffic and/or state information it demands, and bandwidth requirements [14, 15]. This however lies out of the scope of this paper: the only facet of the protection mechanisms we study here is their effect on TCP behavior.

The remainder of this paper is structured as follows: in Section 2, we will give an overview of GMPLS protection mechanisms. In the following Section 3, we will discuss the reactive nature of TCP. The effects of the GMPLS recovery actions on TCP that we will study in this paper, and the adopted approach, will be outlined in Section 4.1. In the subsequent sections, we will present the simulations used to seek an answer to the questions raised in our discussion of TCP: in Section 5, we will investigate the effect of the speed of protection switching, while in Section 6 we will focus on the impact of the changing path length. The switch-back operation to the original path after the failure has been repaired will be dealt with in Secion 7. The joint impact of speed and changing RTTs will be discussed for the presented GMPLS recovery schemes is the subject of the case study in Section 8. In Section 9 we will summarise the conclusions of the paper.

2. GMPLS Protection Mechanisms

Protection in MPLS is based on pre-established Label Switched Paths (LSPs), spanning a single link or node from an associated working LSP, or the whole working LSP from ingress to egress. The former case is generally denoted as Local Protection, and the latter as Path Protection [16, 17]. These protection mechanisms are illustrated in Fig. 1. Path Protection always (e.g. during failure 1 and failure 2) switches the traffic in the ingress (node A in the figure) to the single backup LSP. Local Protection needs a backup LSP per protected link or node: in case of failure 1, traffic will be rerouted along the middle backup LSP —indicated by the dotted arrows— that is pre-established between the end-points of the affected link B–C. In a similar

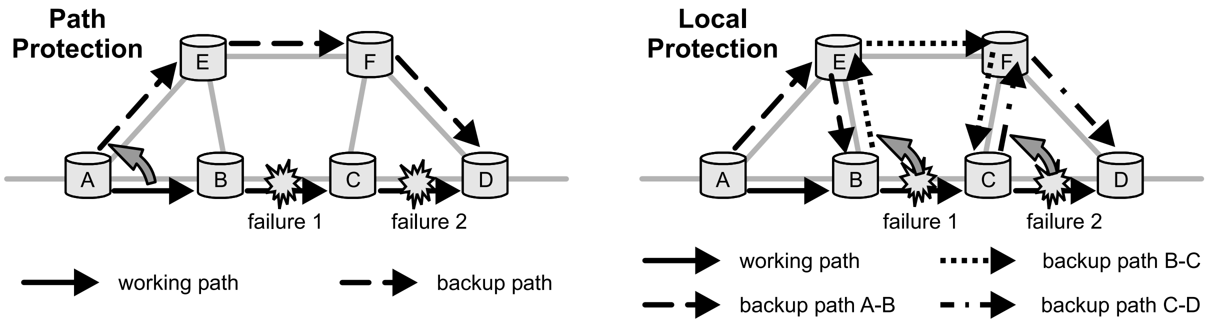


Fig. 1. Illustration of Path and Local Protection, under two different failure scenarios.

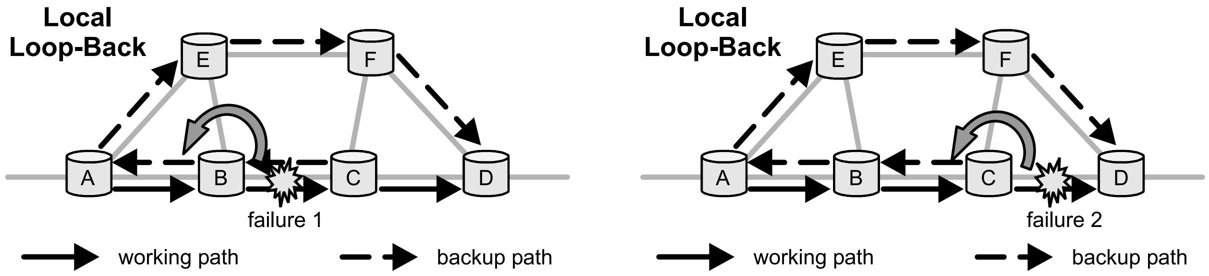


Fig. 2. The Local Loop-Back technique combines the advantage of Path Protection (single backup LSP) and Local Protection (protection switch performed locally in an LSR adjacent to the failure).

way, the backup LSP indicated by the dash-dotted arrows is used in response to failure 2. The Label Switch Router (LSR) where the backup LSP is originating and the switch-over operation from primary to backup path is performed, is called the Path Switch LSR (PSL), whereas the Path Merge LSR (PML) is the one where backup and primary LSPs are merged into a single outgoing LSP.

From the figure, some weaknesses of each of the approaches are immediately apparent. Local Protection typically needs to foresee a backup LSP for each link (or node) of the primary LSP (in Fig. 1, there is a backup LSP for each link) —yet, some workarounds exist where label stacking is available [18]. Path protection on the other hand, while requiring only a single backup LSP per primary LSP, necessitates additional signaling functionality (the PSL needs to be notified of the upstream failure, in order to decide when to switch to the backup LSP) which takes more time and thus results in more data loss.

A third protection scheme, illustrated in Fig. 2, was suggested in [19] and combines the "best characteristics" of both schemes: Local Loop-back. The key idea of this approach is to have only a single backup LSP per primary LSP, while permitting local protection switches (i.e. at the node detecting the failure, without requiring a signal to be sent to a distant PSL). The figure

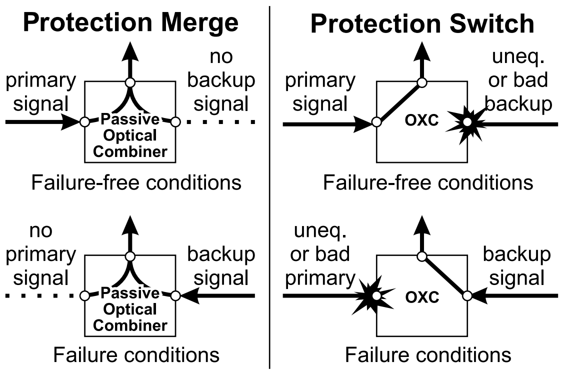


Fig. 3. A protection merge (left) can be realized by a passive optical combiner, if and only if backup and primary signals never are received simultaneously. If this condition cannot be met, a protection switch (right) is needed instead of a protection merge [13].

shows that the loop-back protection switch is performed in different LSRs (although a single backup LSP is required), under distinct failure conditions. The backup LSP consists of two parts: first from the penultimate node back to the source node of the primary path (using the same links in the reverse direction), and then via a disjoint path to the destination node. As shown in the figure, at what LSR exactly the loop-back protection switch is performed, depends on the failure conditions.

These protection mechanisms were originally conceived for electrical MPLS, which is a packet-switch-capable (PSC) technology. The WDM layer however is an optical, lambda-switch-capable (LSC) layer in GMPLS. When porting the mechanisms to a circuit-based, non-merge-capable layer, such as the LSC layer (further denoted as MPλS), two main issues arise [15]. The first is that merging primary and backup paths at the PML may not be possible, as illustrated in Fig. 3. At all times, only one of the optical signals may be forwarded along the outgoing interface —even when the available bandwidth would suffice, when looking at the bandwidth effectively used by IP. Thus, when along both primary and backup incoming interfaces a signal carrying data would come in, at least one of them will be discarded.

A second and related issue, which has its implications on the dimensioning of the network [14], is that in MPλS a “label” corresponds with a wavelength, and therefore implies the occupation of a “circuit”. Where PSC technologies allow statistical multiplexing of several (backup) LSPs over the same link, MPλS and other circuit-switched approaches, such as TDM or fiber-switching, do not.

3. The reactive nature of TCP

The Transmission Control Protocol (TCP) is a connection-oriented data communication protocol, which is reliable in the sense that the sender keeps trying to send a data segment until the receiver acknowledges its proper receipt and the acknowledgement packet (ACK) does not get lost in the network. The amount of data the source may send out before it stops and waits for ACKs to come in from the receiver, is limited by the minimum of the congestion window (*cwnd*) and the receiver's advertised window (*rwnd*).

While TCP is still being studied and continuously under development (see e.g. [20] for a recent overview), the TCP basics have been described a relatively long time ago in RFC 793 [21]. The basis of TCP congestion control can be summarized in four components [20]: (i) the additive increase multiplicative decrease (AIMD), halving the congestion window in response to a packet drop, and if not, increasing it with one segment per round trip time (RTT); (ii) the use of a retransmit timer to start resending packets if they are not acknowledged within a certain period of time, denoted as the Retransmission TimeOut interval (RTO); (iii) the slow-start mechanism for initial probing of available bandwidth, where *cwnd* is increased with one Sender Maximum Segment Size (SMSS) per received ACK until it reaches the slow start threshold (*sstresh*); and (iv) the principle of ACK clocking, sending packets in response to the reception of ACKs.

When considering GMPLS, and in particular protection switching, two effects have an important impact on TCP behavior. The first is a sudden change in RTT experienced by the TCP flows. Indeed, the time needed to deliver a packet from source to destination (and vice versa for the ACKs) will change suddenly when the packets are sent along another path. The time to elapse before TCP triggers the retransmission of a packet, i.e. the RTO, is based on an estimate of this RTT that is called the Smoothed Round-Trip Time (SRTT), and is obtained by low-pass filtering the measured RTTs, which also takes into account the variance on the RTT [22]. A (protection) switch may cause the retransmission timer to expire, resulting in the unnecessary retransmission of some data segments (when packets are not lost, but simply are underway for a longer time). In order to respond reasonably fast to changing network situations, the low-pass filtering of the average and deviation of the measured RTT should not react too slow.

A second effect is that a burst of consecutive packets may be lost when switching a flow to another path. This will be the case when considering a protection switch in response to a link failure: packets in transit on the failing link will be lost, and so will the subsequent ones, until the failure has been detected and appropriate action (in casu the protection switch) has been taken. If we consider e.g. the NewReno version of TCP [23], it will fall back to the Fast Recovery/Fast Retransmit algorithm. This procedure is triggered by the reception of three duplicate ACKs, upon which *sstresh* is adjusted to half the flight size (i.e. the amount of unacknowledged data sent out by the TCP source) and the congestion window is reset to *sstresh* plus three times the SMSS (to account for the three segments that have left the network and caused the duplicate ACKs). The first lost segment is then retransmitted. To keep track of the recovery process, the highest unacknowledged sequence number is stored in the variable *recover*. The Fast Retransmit/Fast Recovery process terminates when this sequence number has been acknowledged. Until then, the source classifies the receipt of an acknowledgement in three categories [23]: (i) a Duplicate ACK (with the same sequence number as a previously received one), (ii) a Partial ACK (with a new sequence number, but not acknowledging all data sent out before the Fast Retransmit/Fast Recovery procedure was triggered), or (iii) a Complete ACK (acknowledging all data, which had been sent out at the time the Fast Recovery/Fast Retransmit procedure was triggered, i.e. with sequence number *recover*).

In case a large burst of consecutive packets is lost —as may happen in case of a link failure— the Fast Retransmit/Fast Recovery rules cause a performance lack. Indeed, we can interpret these rules as follows. The number of duplicate acknowledgements —denoted as *ndup*— received one RTT after initiating the Fast Recovery/Fast Retransmit procedure is calculated as the flight size minus the size of the lost burst. This *ndup* is minimally three and maximally the flight size (at the moment of the initiation of the procedure, or thus twice the new value of *sstresh*) minus once the SMSS. This number *ndup* times SMSS gives the amount by which the congestion window size is incremented above *sstresh* during the considered RTT. However, only the part of the congestion window above the flight size allows sending new data (thus with a sequence number larger than the value stored in the *recover* variable). The amount of new data sent during this first RTT of the Fast Retransmit/Fast Recovery period can be estimated using Eq. 1:

$$\begin{aligned} new &= \max(0, cwnd - (lastsent - lastack)) \\ &= \max(0, (sstresh + ndup) - flightsize) \\ &= \max(0, (sstresh + flightsize - burstsize) - flightsize) \\ &= \max(0, (sstresh - burstsize)) \end{aligned} \tag{1}$$

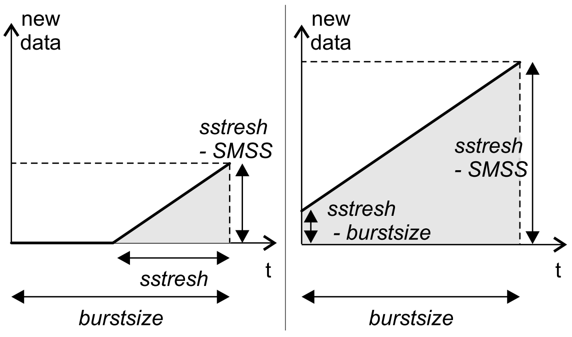


Fig. 4. Evolution of new data sent during successive RTTs in the Fast Retransmit/Fast Recovery phase of TCP NewReno. The left part illustrates the case where *burstsize>sstresh*, whereas the right part illustrates *burstsize<sstresh*. Note that the scales of the axes are not the same for both figures; the sloping part of the graphs increases with one SMSS per RTT.

At the end of this RTT, a partial acknowledgement should arrive as result of the retransmitted segment. The effect of the partial acknowledgement is that the flight size is decremented by the amount of acknowledged data (equaling the size of the acknowledged data segment); this is SMSS bytes more than the decrement of congestion window size (which is typically zero). In other words, the receipt of a partial acknowledgement causes the flight size to decrement faster than the congestion window size. Therefore, after the receipt of one or more partial ACKs, the amount of new data that can be sent during the following RTT will evaluate to a strictly positive value. Subsequently, it keeps growing by one SMSS per retransmitted data segment (i.e. per RTT). Indeed, while the second lost segment is retransmitted, the source will receive a duplicate acknowledgement for each data segment potentially sent out during the first RTT. Each of those duplicate ACKs will result in the increment of the congestion window with one SMSS, allowing the same amount of new data to be sent out, in addition to the SMSS gain of the previous partial acknowledgement: the number of newly sent packets increases with one per RTT. This scenario per RTT will keep repeating until the last lost segment has been retransmitted. The amount of new data transmitted each RTT during the Fast Retransmit/Fast Recovery period is sketched in Fig. 4.

In conclusion, the fast recovery/fast retransmit procedure would last for the number of lost data segments (*burstsize*) times the RTT. As long as the size of the unacknowledged burst remains higher than *sstresh* (minus SMSS), no progress will be made. The overall progress during this procedure is estimated as follows:

$$\mathit{progress} = \frac{\max(0, \mathit{sstresh} - \mathit{burstsize}) + (\mathit{sstresh} - \mathit{SMSS}))}{2}$$

$$\cdot \frac{\min(\mathit{sstresh}, \mathit{burstsize})}{\mathit{SMSS}} \tag{2}$$

This may become significantly less efficient than e.g. returning to slow start, when the size of the burst of lost data becomes too large. The drawback of slow start is that it needs to retransmit everything outstanding at the time the source detects a loss (i.e. twice the *sstresh*). Therefore, the maximal burst size (*mbz*) for which the Fast Recovery/Fast Retransmit procedure would effectively turn out to be the fastest is given by the inequality Eq. 3. The right hand side gives the amount of data sent during mbz round-trip times, having sequence numbers starting with the first packet of the burst. The left hand side gives the number of packets that would have been transmitted using the Fast Retransmit/Fast Recovery algorithm of NewReno.

$$\begin{aligned} \mathit{progress} + 2 \cdot \mathit{sstresh} &\geq (1 + 2 + \cdots + 2^{mbz}) \cdot \mathit{SMSS} \\ &\geq (2^{mbz+1}) \cdot \mathit{SMSS} \end{aligned} \tag{3}$$

The conclusion is that, when developing a fast protection scheme, one should try to minimize the amount of lost packets: when this is not the case, one would intuitively expect from the above that it is better to allow the RTO timer to expire, bringing the TCP connection back into the slow start mode.

The discussion held so far focussed on a single TCP flow. However, a link will usually carry a multiplex of many TCP flows, and therefore the number of losses inflicted on a single TCP flow (i.e. the *burstsize*) and/or the number of affected flows may be relatively low. What exactly the effects of switching are on such an aggregate of flows, hence is not that straightforward to predict and far more difficult to capture in formulae. This is why we resorted to simulations, as described in the following section.

4. The effects of recovery actions on TCP

From our discourse on the reactive nature of TCP, we have learnt that for TCP basically two effects take place when considering network failures and GMPLS recovery actions: a burst of packet losses, and the changing RTT because of the switch to a (usually longer) backup path. Now, how well are recovery actions, and the aforementioned effects they cause, digested by the principal client layer protocol TCP? In Section 4.1 we will clarify what exactly we will focus on. For our study, we have resorted to simulations, whose set-up will be discussed in Section 4.2.

The criterion used to quantify the behavior of TCP is the so-called goodput, which is briefly explained in Section 4.3.

4.1. Topics

We will address four topics, being (i) the effect of protection speed, (ii) the effect of changing RTTs at a switch-over, (iii) the effects of switch-back operations for both optical and electrical GMPLS layers, and (iv) how the previously discussed effects translate to the major GMPLS protection strategies.

The first issue we will investigate is what the influence is of the speed of protection switching on TCP flows. Indeed, using a (G)MPLS protection mechanism, the network can respond quite fast to network failures. The question arises how advantageous this is from a TCP point of view. Clearly, the number of losses inflicted on the TCP flows will be directly related to how fast the connectivity is restored after a failure. Also, if the time needed to perform the switch-over to an alternative path is relatively large compared to the RTT for the affected flows, then TCP's retransmission timer may expire, causing it to fall back to slow start. As indicated before, this implies that the retransmission of the lost packets may be realised at a quicker pace than with the Fast Recovery/Fast Retransmit procedure. Still, the fact remains that due to the longer outage of the connectivity more packets will be lost. What the net effect is, and how much worse off we are when the protection switch times get larger, will be addressed in Section 5. The issue of protection switching speed gets even more interesting when those actions are performed by an (electrical) PSC layer. In that case, the "switched flows" redirected via a backup path may have to share bandwidth with "fixed flows" already present on (parts of) the backup path and whose routing is not changed in response to the failure. Thus, not only the flows crossing the failing network part, but also other flows will be affected by a failure. The speed of protection switching will have a major impact on how this interaction between "fixed" and "switched" flows evolves.

The speed of protection switching mainly affects the number of losses inflicted on the suffering TCP flows. A second important effect of protection switching we highlighted before was the change in RTT. Usually the backup path, to which TCP flows will be switched by a GMPLS recovery mechanism, will be longer than the originally followed working path. Therefore, when switching TCP flows to the backup path, they will experience a sudden increase in RTT, possibly causing their RTO timer to expire. Indeed, this timer uses an estimate of the RTT (see Section 3) that is based on the shorter original working path. This results in unnecessary retransmissions of segments that simply had to make a longer journey than their predecessors. Exactly how detrimental this effect is, is the subject of a second series of simulations presented in Section 6.

A third set of experiments, focusses on the switch-back operation that may follow a repair of the network failure. When flows are switched back to the original paths, this results in a decrease of the RTT. Moreover, depending on the GMPLS layer at which the recovery actions are performed, out-of-order-delivery and/or packet losses may be inflicted on the TCP flows. If the GMPLS layer is a merge-capable one —such as electrical MPLS— the PML will merge packets redirected to the original working path with those still traveling on the longer backup path. Obviously, out-of-order delivery will result, causing some innecessary retransmissions. When both incoming links (along working path and backup path) at the merge point are highly loaded, also buffer overflows (implying packet loss) may occur. For non-merge-capable GMPLS technologies —such as any optical layer, recall Fig. 3— after the switch-back operation the packets still underway along the backup path will be discarded at the PML, thus requiring retransmission. The difference in TCP behavior for electrical and optical cases following the switch-back operation are treated in Section 7.

Using the insights gained by adressing the outlined issues, we will try to asses the main differences between the prevailing GMPLS protection strategies. A case study will compare them from a TCP point of view in Section 8.

4.2. Simulation approach

The answer to the questions raised by the four topics is sought by means of simulations, using the wide-spread tool Network Simulator (a.k.a. ns–2) [24]. This is a discrete event simulator targeted at networking research, providing substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks.

To investigate a particular protection scheme, or one of the parameters playing an important role in its effects on TCP behavior, we will always refer to the same generic topology (or part thereof). The network we consider is made up of an (G)MPLS backbone in which we simulate a link failure, as sketched in Fig. 5. The parameters used for the simulation are listed in Table 1. We have chosen to keep the bottleneck —causing the losses that will limit the bandwidth troughput generated by the TCP flows— to be located in the access part of the network (at least under failure free conditions) by giving the access links a lower bandwidth. The propagation delay of

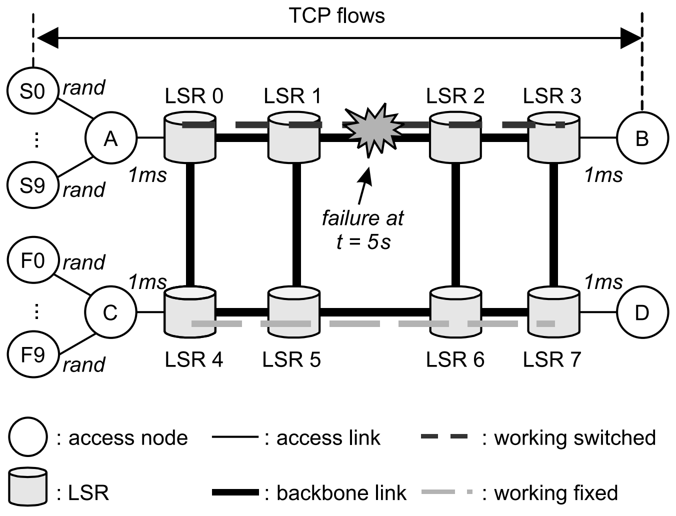


Fig. 5. Illustration of general set-up of our simulations. Each LSP carries an aggregate of 100 TCP flows, originating at nodes Sx that are connected to the backbone via access links, of which the first (to node A) has a randomly generated length (in terms of propagation delay). The TCP sources are started at different random times.

the links was set in the range of 1–60 ms depending on the scenario at hand, resulting in RTTs of the order of one to a few hundred milliseconds (see resp. sections for exact values).

The packets flowing through this network follow predefined routes (paths), for which we exploit the MPLS provisions of the simulation tool —evidently, the exact routing paths depend on the protection mechanism under study. Along these paths, an aggregate of many TCP flows is sent. In order to reflect that each of these flows usually will be starting and/or continuing outside the backbone, and therefore may experience different delays, we have added different access nodes. Each of these access nodes will act as the source of a set of TCP connections. The access nodes will be connected to the rest of the network through access links having randomly generated propagation times uniformly distributed in the range [10 ms,100 ms]. In addition, to avoid unnatural synchronization between flows originating at the same access node in our simulated topology, we also use the random generator to determine the starting times of the TCP flows. Consequently, the network will be loaded with a mixture of TCP flows with uncorrelated starting times. Furthermore, by experiencing different RTTs, the flows will have different reaction times to network changes affecting their RTTs; indeed, both the RTO timeout mechanism and the ACK-clocking mechanism are largely dependent on this RTT.

The two main parameters that will be varied for the scenarios used to address the topics outlined before are:

Table 1. Common simulation parameters.

Parameter	Value
TCP	TCP NewReno [23]
nr. of source nodes	10
nr. of TCP flows	10 per source node (thus per LSP 100 in total)
access bw.	80% of backbone bandwidth; (90% for the timing effects in Section 5)
rand delay	randomly generated propagation delay using uniform distribution in [10 ms,100 ms]

- **Protection speed:** This is the time that elapses between the occurence of the link failure, and the switch of the flows to the alternative path. It will be denoted by δ.

- **Change in RTT:** This is related to the extra propagation time the TCP flows will experience when they are switched from the primary to the backup path. For propagation times, we will use roman letters, usually d. The resulting difference in RTT between primary and backup paths will be denoted by ΔRTT.

The scenario used to investigate the effect of changeing these parameters will consist of three periods of five seconds each, as summarised in Table 2. During the first period, the TCP sources will start and gradually fill the network with traffic. At $t = 5\,\mathrm{s}$, the link LSR1–LSR2 will go down. Subsequently, at $t = 5\,\mathrm{s} + \delta$, the protection switch will be carried out, rerouting the affected flows along the backup path. In order to be able to precisely influence the timing, we perform the switch "manually" exactly δ (delta) seconds after the link went down. That means that all packets in transit on link LSR1–LSR2 at the time of the failure ($t = 5\,\mathrm{s}$) and arriving at LSR1 (or LSR2 for the ACKs) between $t = 5\,\mathrm{s}$ and $5\,\mathrm{s} + \delta$ will be lost. The last $5\,\mathrm{s}$ period of our simulation, the link LSR1–LSR2 will be up again.

Each scenario was simulated 150 times different random seeds to generate the start times of the TCP sources, and the *rand* propagation delays of the access links.

Table 2. Common simulation scenario.

Time	Event
$t = 0\,\mathrm{s}$	Start of simulation
$0.5\,\mathrm{s} < t < 1\,\mathrm{s}$	TCP sources start to generate traffic
$t = 5\,\mathrm{s}$	Link LSR1–LSR2 goes down
$t = 5\,\mathrm{s} + \delta$	Protection switch is carried out
$t = 10\,\mathrm{s}$	Link LSR1–LSR2 is up again
$t = 15\,\mathrm{s}$	End of simulation

4.3. TCP goodput

The criterion we will use to evaluate the protection mechanisms is TCP goodput. This is the number of unique bytes successfully transmitted end-to-end (i.e. from TCP source to TCP destination) within a certain time interval, expressed in e.g. bytes per second. Clearly, this is what end-users of the network employing TCP will care about. In the graphs we present, we usually will express goodput relatively to the link bandwidth available to the (aggregate of) TCP flow(s). This means that we take the total of successfully transmitted bytes, as recorded in a variable *bytes* (see further), at fixed times $t_i = i \cdot T$, with an interval of duration T, and divide it by the maximum number of bytes that could have been transmitted (i.e. T multiplied by the bandwidth): the plotted values are given by Eq. 4.

$$Good(t_i) = \frac{bytes(t_i) - bytes(t_{i-1})}{T \cdot bandwidth}, \quad \forall\, t_i = i \cdot T \tag{4}$$

Using ns–2, we record the goodput at the TCP receiver, which is called the Sink. We have slightly extended the ns–2 program to make a Sink keep track of the number of bytes that it has received in order; for this we use a variable named *bytes*. This variable, associated with a Sink, is updated upon the receipt of a packet, sent by the corresponding TCP source. Based on the sequence number contained within the TCP header, the Sink is able to determine whether the packet contains data that (i) hasn't been sent before, and (ii) is the data packet it expected to receive next (i.e. causing a new ACK to be sent out, which may be a partial ACK or a complete ACK). Only when both conditions are fulfilled, the variable *bytes* tracing goodput is incremented. Another way of describing this, is that upon each receipt of a packet, *bytes* is incremented with the newly ACKed data: the byte sequence number sent in the ACK minus the one sent in the previous ACK.

Consequently, when a packet loss has occurred, there will be no advance in goodput (as traced by *bytes*) for the TCP flow it is part of, until the packet has been successfully retransmitted. With the Fast Retransmit/Fast Recovery, this will occur only when a partial or complete ACK would be sent out. However, in the meantime new packets may have been sent out and successfully received by the Sink, causing duplicate ACKs, before the lost packet arrives. Hence, when finally the lost packet arrives, the *bytes* variable will suddenly increase with the difference between the highest received sequence number and that of the predecessor of the lost packet (assuming that since the losses triggering the Fast Retransmit/Fast Recovery, no other packets have been lost). When plotting the goodput evolution relatively to the available link bandwidth, this increase can obviously cause the goodput value for the corresponding time interval (given by Eq. 4) to be bigger than 100%.

5. The effect of timing: the influence of the speed of protection switching

5.1. Purpose and simulation scenario

The first issue we address is the impact of the speed at which the protection switch is carried out. As outlined in Section 4.1, in a packet-switch-capable (PSC) GMPLS domain, the impact of the protection switch will be the biggest. In a PSC domain, the flows switched to a backup LSP will go into competition with other flows already present on (parts of) the backup LSP. The observed behavior will depend for a large part on the timing of this protection switch. Indeed, when the switch is performed very fast, the TCP flows being switched to an alternative path will still be sending at a relatively high rate (as they will not have detected any packet losses yet) when joining other flows on the links that are part of the backup path. This may cause severe buffer overflows, resulting in excessive segment loss. Therefore, it is not obvious that making a protection mechanism act as fast as possible is the best thing to do.

The set-up used to investigate these matters is depicted in Fig. 6. We set up two sets of TCP flows: the "switched flows" originating at nodes S0–S9 following an LSP that will experience a failure and subsequent protection switch, and the "fixed flows" originating at nodes F0–F9 that keep following the same primary LSP (unaffected by the failure); the ACKs will follow the

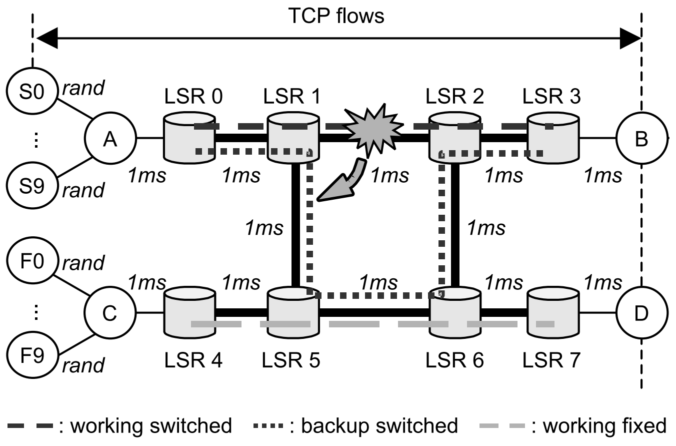


Fig. 6. Simulation set-up used to study the impact of protection switch speed on TCP.

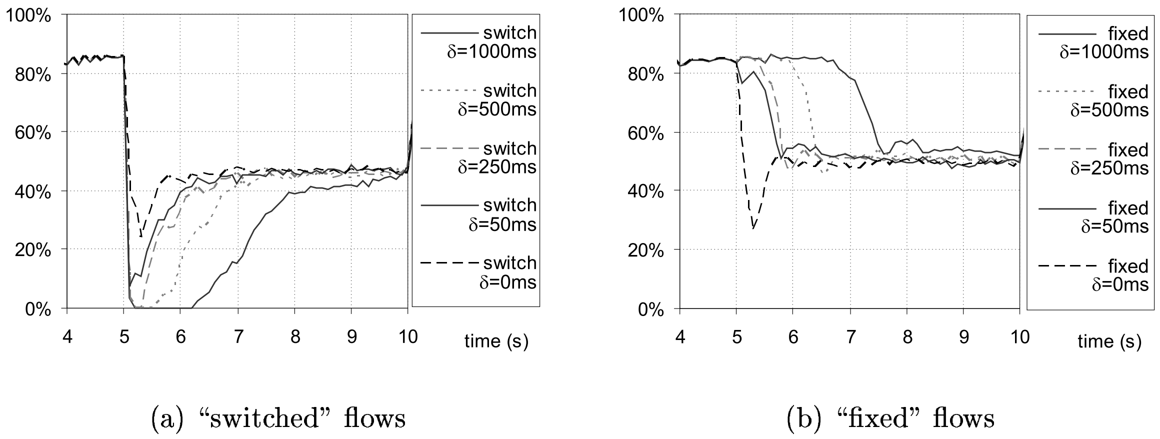


Fig. 7. TCP goodput evolution over time for different values of the switching time (delay) δ for the whole of the "switched" resp. the "fixed" flows. The goodput is expressed in % of backbone link bandwidth and was measured with a resolution of 10 ms (i.e. T=10 ms in Eq. 4).

reverse paths. The simulation period we focus on is when the link LSR1–LSR2 goes down, thus the interval [5 s,10 s].

5.2. Results for the electrical case

In Fig. 7, the evolution of goodput over time is depicted for different values of δ in the scenario presented before. There the heavy impact of the immediate buffer overflow on the fixed flows for δ=0 is clearly visible. Also, note that the time it takes for the interacting TCP flows to stabilize is in the order of a second or more.

We compared the different values of δ by considering $f(\delta) = Good(\delta) \,/\, Good(0)$, where $Good(\delta)$ is the total goodput, attained by the whole of fixed and switched flows, during the first 1.5

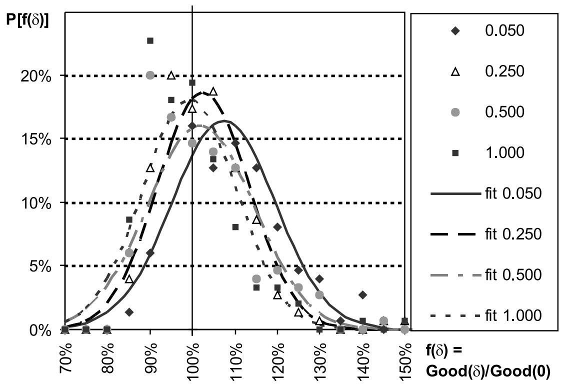


Fig. 8. Histograms (with a resolution of 5%) and normal fits for relative amount of goodput. A marker at (x, y) for a particular δ means that y% of the simulation results had $f(\delta)$ within $[x, x + 5\%)$.

Table 3. Comparison of different protection switch delays. The left column represents the x-value corresponding to the average of $f(\delta)$, i.e. the peak of the normal fit in Fig. 8, minus 1 (this is $Good(\delta)\,/\,Good(0\,\text{ms}) - 1$). The second column indicates the percentage of simulation results where $f(\delta) < 100\%$ (or, equivalently, $Good(\delta) < Good(0\,\text{ms})$), whereas the rightmost column gives the number of simulation results where $Good(\delta)$ was maximal (i.e. compared to other delays).

δ	relative difference in goodput compared to $\delta = 0$	% of random cases where δ is worse than $\delta = 0$	% of random cases where delay δ is best
0.000 s	0.00%	0.00%	20.00%
0.050 s	+9.85%	24.00%	64.67%
0.250 s	+4.99%	36.67%	9.33%
0.500 s	+4.39%	42.67%	5.33%
1.000 s	+1.75%	49.33%	0.67%

seconds after the link failure for switching time δ, as listed in Eq. 5. We chose 1.5 s as "integration interval", it being the relevant period for the differences in behavior for smaller values of δ. Using the data of our 150 runs, we constructed the histograms and corresponding normal fits as depicted in Fig. 8. That graph shows that, on average, all cases of δ>0 result in a better goodput than having an immediate protection switch (δ=0). Numerical results from the comparison are summarized in Table 3.

$$\begin{aligned} Good(\delta) &= \text{goodput for } \textit{all} \text{ flows in } (5\,\text{s},6.5\,\text{s}] \text{ for switching time } \delta \\ &= \sum_{\forall\ \text{flows}} \frac{bytes(6.5\,\text{ms}) - bytes(5\,\text{s})}{1.5\,\text{s} \cdot bandwidth} \end{aligned} \qquad (5)$$

These results seem to lead to the conclusion that pushing fast protection to the limit (i.e. extremely fast) may not be the wisest thing to do. From a qualitative point of view, the influence of changing δ can be explained as follows: if δ is set to zero (which corresponds to an immediate detection and subsequent triggering of the protection mechanism), the switched flows will join the fixed ones at LSR5 at a time when they are both sending at a quite high rate (limited only by the rate of the access links A–LSR4, resp. C–LSR8). This will result in an immediate buffer overflow at LSR5, causing a burst of losses affecting both flow categories. When introducing a certain delay (δ strictly positive), the switched flows will experience a higher number of losses (as packets cannot be forwarded along the primary path during that time), forcing the TCP sources to back off before they are switched to the backup path. The immediate buffer overflow at LSR5 will be avoided, and the fixed flows will be approached more "gently". We can indeed not avoid link LSR5–LSR6 becoming the bottleneck, but the buffer overflow at LSR5 will occur at a later time, and will cause fewer losses compared to the δ=0 case. Indeed, carrying out the protection switch as fast as possible in the considered case of electrical MPLS is not the most advantageous thing to do: it may be better to have a slightly slower protection action.

However, to decide what exactly is the "best" time to perform the protection switch, is not obvious. It at least depends on the link load (in the case presented above, when all links are up, backbone links are loaded for max. 90% due to the limits in the access part, but a protection switch results in a sudden load on link LSR5–LSR7 of almost 180%), the RTT experienced by the TCP sources (larger RTT means slower response to topology changes), and the number of concurrent TCP flows (larger number results in faster stabilization, up to a certain limit).

From a practical point of view, the results seem to indicate that from a TCP goodput perspective, having fast protection (order of tens of milliseconds) is not that bad —despite the sudden overload. This conclusion is probably even more true in cases where backbone links carry a vast amount of concurrent TCP flows (cf. faster stabilization than small number of flows, and therefore optimal delay shifts towards δ=0) and/or are fairly underloaded. Indeed, when backbone links do not form the bottleneck for TCP flows, interaction between switched and fixed flows will be limited. Indeed, repetition of the simulation scenario discussed above showed that —all other parameters left apart— for an access link bandwidth being a smaller fraction of the backbone bandwidth (e.g. 60% instead of 90%, thus resulting in maximal load on link LSR5–LSR6 of 120%), the optimal protection switch delay clearly shifts to lower values (towards δ=0). The simulations carried out seem to indicate that only if the 1 of protection switching is well below 50 ms, TCP effects may call for a stop to the efforts to minimize it.

All this however does not imply that extremely fast protection switching is a must for TCP: the differences in goodput for switching times δ in the range 0–250 ms do not differ all that much, especially when the number of TCP flows is large.

5.3. Results for the optical case

The simulation discussed above considered fast protection at an electrical MPLS layer. However, if fast protection is offered by an optical MPλS layer (or any other "circuit" layer, e.g. using TDM channels as "labels", as in SDH networks), we are in an altogether different situation. Indeed, then we will have no interaction between competing TCP flows: in that case we assume that the capacity for protection is reserved, and is fully available from the very instant the protection switch is carried out. Clearly, dynamic behavior of TCP in response to packet losses will still occur.

In this optical case, the intuitively clear conclusion we have drawn from our simulations is: the faster the protection switch at the optical layer is performed, the better (from a TCP goodput point of view). The simulations performed for this case had a link going down for a certain amount of time δ, without any protection actions taken at the MPLS level. For 140 random cases (random RTTs, etc., as before) and δ in {0, 5, 10, 20, 30, 40, 50, 250, 500, 1000 ms} we saw that in 94% of the cases, δ=0 was the best (only packets in transit on failing link are lost); in the remaining 6% of the cases, δ=5 ms was the best (which is due to details in dynamic TCP behavior in some rather peculiar cases). Thus, the avoidance of TCP interactions is an advantage of protection at the optical layer with respect to protection at the merge-capable electrical MPLS layer. At the optical layer, even extremely fast protection switching does not seem to pose any problem (at least from a TCP point of view); clearly, the price paid for this is a higher cost in terms of network capacity to install (see e.g. [14]).

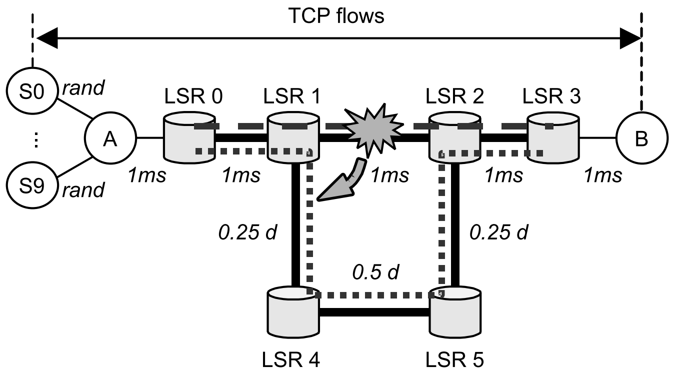


Fig. 9. Simulation topology used to investigate the influence of changes in RTT, caused by a protection switch, on TCP behavior. The access links had a bandwidth that was 80% of that of the backbone links.

6. The effect of changing RTTs

6.1. Purpose and simulation scenario

Now we will focus on the effect of the sudden increase in RTT experienced by TCP flows when switched to a longer backup path. The simulation topology is depicted in Fig. 9. The total propagation delay of the span LSR1–LSR4–LSR5–LSR2 was set to d, with d in {1 ms, 2 ms, 10 ms, 50 ms, 100 ms}. The original working path crossed the link LSR1–LSR2 with a propagation delay of 1 ms. Thus, the increase in RTT is given by $\Delta RTT = 2 \cdot (d - 1\,\mathrm{ms})$. The average RTT of the original paths between the sources Sx and destination B, was 120 ms, as can be derived from Fig. 9. So, when setting d to e.g. 100 ms, this results in almost tripling the RTT (increase with 2·99=198 ms). This is what could happen when considering e.g. local loop-back protection for a failure of a link close to the egress node of the LSP (esp. in a network with a low connectivity degree): in that case the backup path will indeed be considerably longer than the working path, as it will be made up of almost the entire original path, its reverse, plus the link-disjunct alternative route to the egress (recall Fig. 2).

As the comparison of the different increases in RTT will surely depend on the speed of protection, we have repeated the experiment with a protection switch delay δ in {5 ms, 10 ms, 20 ms, 100 ms}.

6.2. Results

When we consider TCP flows switched to a backup path that is longer than the original working path, the increase in RTT they thus experience reinforces the temporary drop in the goodput

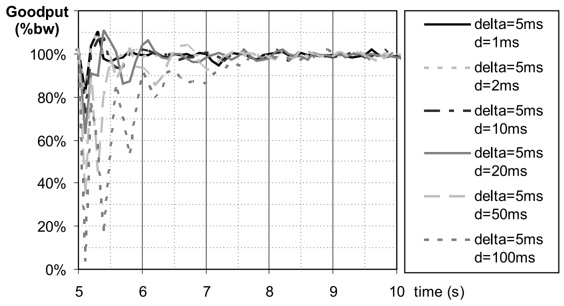


Fig. 10. Goodput evolution in interval (5s,10s] for δ=5ms, and d in {1 ms,2 ms,10 ms,20 ms,50 ms,100 ms}. It is plotted here with a resolution of 100 ms, which means that every 100 ms, the number of bytes successfully transported end-to-end in (t−100 ms,t] was measured.

evolution (due to lost packets and their retransmission). Indeed, the switch to a longer path may cause TCP's retransmission timer to expire even when the protection switch is carried out quickly enough (small δ). Moreover, since the speed of increase in TCP window size (which will be reduced due to detected packet losses) is related to the RTT (cf. ACK-clocking property of TCP), a larger RTT also means slower recovery: it will take longer for the flows to fully exploit the available bandwidth again. Intuitively, we expect that the larger the difference in RTT, the more severe the penalty will be.

In Fig. 10 we have plotted the goodput evolution for the case δ=5 ms, which shows the expected drop right after the link failure, and the subsequent gradual recovery. It confirms our qualitative discussion: the larger the difference in RTT, the more severe the drop in goodput is, and the slower the recovery. If we concentrate on the smaller differences (say d up to 50 ms), and compare the total goodput achieved within the first 1.5 seconds after the failure (denoted by $Good(\delta, d)$, with the same definition as in Eq. 5), we can construct a histogram for the relative goodput compared to d=1 ms (no change in RTT). This results in the graph presented in Fig. 11, and the accompanying numerical data in Table 4, with similar interpretations as those in Section 5.

From these numerical data, we may conclude that for the considered switching time δ=5 ms, the penalty of having a longer backup path is probably acceptable if the increase in RTT is limited to around 50% (resulting in a drop in goodput of less than 10%). One may wonder whether we can accept larger differences in RTTs when the reaction time of the protection mechanism is slower, i.e. when δ is larger. If it is sufficiently large, we expect to have the same drop in goodput for all cases of d, immediately after the failure: it will completely fall back to

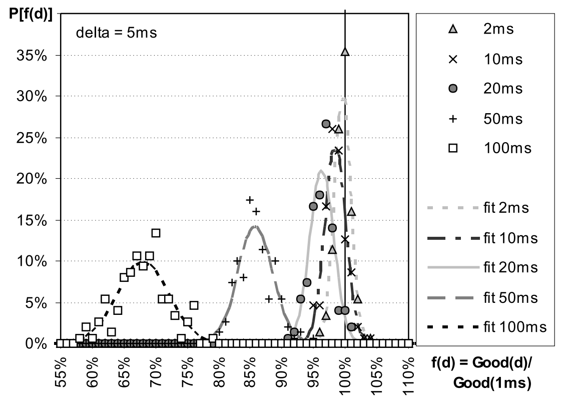


Fig. 11. Histograms (with a resolution of 1%) and normal fits for relative amount of goodput. A marker at (x,y) for a particular d means that y% of the simulation results had f(d) within [x,x+1%). Good(.) is the goodput attained by the total of all TCP flows in the period (5 s,6.5 s].

Table 4. Comparison of different changes in RTT. The second column represents the x-value corresponding to the average of f(d), i.e. the peak of the normal fit in Fig. 11, minus 1 (this is Good(d)/Good(1 ms) − 1). The middle column indicates the percentage of simulation results where f(d) < 100% (or, equivalently, Good(d) < Good(1 ms)), whereas the rightmost column gives the number of simulation results where Good(d) was maximal (i.e. compared to other delays).

d (δ=5 ms)	ΔRTT	relative difference in goodput compared to d=1 ms	% of random cases where d is worse than d=1 ms	% of random cases where delay d is best
1 ms	0.00%	0.00%	0.00%	38.67%
2 ms	+1.67%	+0.13%	42.00%	46.00%
10 ms	+15.00%	−1.03%	75.33%	15.33%
20 ms	+31.67%	−3.13%	94.00%	0.00%
50 ms	+81.67%	−13.66%	100.00%	0.00%
100 ms	+165.00%	−31.48%	100.00%	0.00%

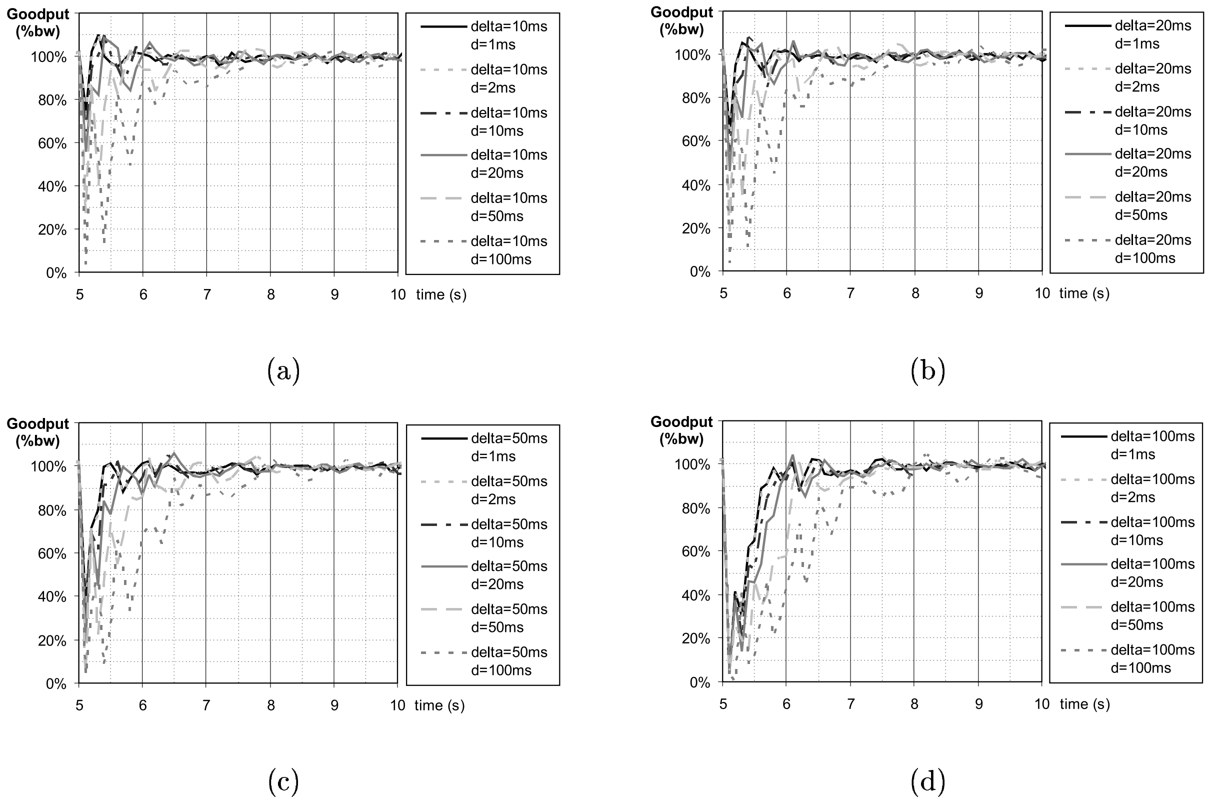


Fig. 12. Goodput evolution in interval [5 s,10 s] for all simulated combinations of (d,δ), grouped per δ.

zero, whatever the change in RTT is. Yet, the speed of recovery will be dependent on the value of the RTT change, as discussed before.

The experiments with δ in {10 ms, 20 ms, 50 ms, 100 ms} resulted in the goodput evolution graphs during [5 s,10 s] as plotted in Fig. 12. These graphs confirm our qualitative predictions: when increasing δ, the drop in goodput is the same for all cases of d. For small values of d, the drop in goodput is significantly bigger when δ is increased. For larger d values the effect of a slower protection switch (larger δ) is not that the initial drop in goodput is larger, but rather that TCP recovers more gradually from it: the time to stabilize is about the same for different protection switch times δ (see Fig. 12 for d=100 ms, where it takes about 3 s for every δ), but the total goodput during this stabilization period is lower for larger δ.

In order to make a more accurate comparison, we again computed the total goodput during the first 1.5 seconds after the failure (as in Eq. 5). Note that this is smaller than the stabilization period for some cases, but taking a larger integration interval does not change the relative positions, in terms of better goodput figures, of the different (d,δ)-cases. Fig. 13 shows that, when increasing δ, the "best" corresponding d value shifts towards lower values. This indicates

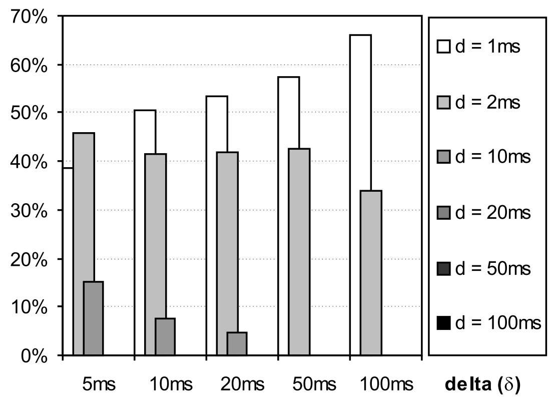


Fig. 13. Bar graph showing, for each δ, the percentage of simulation cases where a particular d was best. A bar with height y% for (d,δ) means that for y% of the simulations with δ, those for d resulted in the best value of Good(d,δ) (i.e. the goodput during (5 s,6.5 s])

that when we wait longer to switch to the protection path, we can be more certain that a lower change in RTT performs better than a large one.

This is confirmed by Fig. 14, showing the total goodput attained by the total of all flows in (5 s,6.5 s] for all the considered combinations of d and δ. From the same goodput data, we can calculate the penalty of increasing the change in RTT (i.e. increasing d) compared to having no change (i.e. d=1 ms), for different values of δ as listed in Table 5. This all shows that the larger δ, the larger the relative penalty is of increasing the RTT (compared to keeping the same RTT).

Note also that Fig. 14 indicates that an increase in path length (and hence RTT) seems to be as damaging as increasing the protection switch time δ with the same order of magnitude (but keeping the change in RTT minimal), e.g. compare points (d,δ)=(5 ms, 100 ms) and (100 ms, 5 ms). This suggests that it is not obvious what is better: performing a fast protection switch to a longer (thus non-optimal) path —as e.g. with the local loop-back scheme— or rather delaying the switch a bit in order to find a route of about the same length as the original path (supposing that it exists) —which could be the case for path protection.

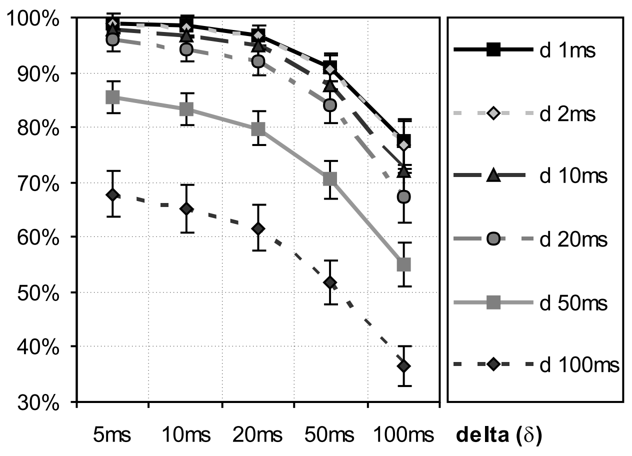


Fig. 14. Comparison of different values of d and δ. A marker with value y% means that averaged over 150 simulation runs, Good(d,δ) was y% of the maximal achievable goodput, i.e. that would be reached when every packet would be sent only once, and the access bandwidth would be fully used all the time. Good(d,δ) is goodput measured in (5 s,6.5 s].

Table 5. Comparison of different changes in RTT. An entry for (d,δ) gives the difference in goodput between d and d=1 ms for that particular δ, i.e. it is Good(d,δ)/Good(1 ms,δ)−1, where Good(.,.) is the total goodput in (5 s,6.5 s]. Each entry is the average of 150 simulation runs using different random seeds.

d	ΔRTT	δ=5 ms	δ=10 ms	δ=20 ms	δ=50 ms	δ=100 ms
1 ms	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
2 ms	+1.67%	+0.13%	−0.20%	−0.11%	−0.23%	−0.71%
10 ms	+15.00%	−1.03%	−1.65%	−2.01%	−3.43%	−6.65%
20 ms	+31.67%	−3.13%	−4.20%	−5.10%	−7.82%	−13.25%
50 ms	+81.67%	−13.66%	−15.43%	−17.55%	−22.56%	−29.04%
100 ms	+165.00%	−31.48%	−33.83%	−36.43%	−43.10%	−52.83%

7. Effects of a switch-back to the primary path

7.1. Purpose and simulation scenario

The previous sections concentrated on the effects of protection switching actions taken upon a link failure. However, when a revertive mode of operation is adopted, traffic is automatically switched back from the recovery path to the original working path upon the restoration of the working path to a fault-free condition. In this section, we focus on the effects of such a switch-back operation in two cases, denoted as "electrical" and "optical", and compare them with the non-revertive mode of operation where the flows keep following the backup paths (a strategy denoted as "none"). The "electrical" case corresponds with protection switching on a GMPLS level where merging of flows is possible, which tallies with protection actions in a PSC domain. The "optical" case reflects the behavior where merging is not possible. The results of the presented simulations assist in a comparative study of electrical and optical protection mechanisms.

The simulation topology is the same as the experiments for the switch-over (see Fig. 9). However, we now focus on the interval [10 s,15 s], where the link is up again. The actions taken in this interval will be one of the following:

- **Electrical:** the routes followed will be switched back to the original path, as indicated in the figure below, at t=10 s. In the electrical scenario, packets still underway on the backup path will be merged with the new ones forwarded along the original path. This merging operation takes place at LSR2 for the data packets, and LSR1 for the ACKs.

- **Optical:** merging is not possible. From the moment the first data packet arrives at LSR2 along the original path (coming from LSR1), the packets coming from LSR5 are discarded. The same goes, mutatis mutandis, for LSR1. As an aside, note that in our simulations using ns–2, we have used a kind of approximation: from t=10 s on, the paths are switched back to the original working path —as in the electrical case— but at t=10 s+x we have made the links LSR5–LSR1 and LSR4–LSR0 go down (fail), with x the propagation delay on link LSR1–LSR2 (1 ms in our example).

- **No action ("none"):** the last option we consider is leaving the routing as it was (the flows continue using the backup path).

We have again considered the effects for d in {1 ms, 2 ms, 10 ms, 20 ms, 50 ms, 100 ms}.

7.2. Results

Before looking at the simulation results in more detail, we are already able to predict the following differences between the scenarios:

- **Electrical:** after the link has come up again, both flows are merged at LSR2. When d gets large, this means we have a fairly long period where the link LSR2–LSR3 is overloaded. Indeed, both the incoming interfaces LSR5–LSR2 and LSR1–LSR2 offer a continuous stream of packets for some time (max. duration = $d - 1\,\text{ms}$; less if TCP reduces its sending rate before). Therefore, we expect some packet losses as long as this overlap exists, at least if d is big enough (big with respect to the buffer size for the interface to LSR2–LSR3). In any case, also when d is small (but larger than the delay of 1 ms on LSR1–LSR2), we will get out-of order delivery at the destination. This will result in duplicate ACKs, to which TCP will respond, as if it were the result of losses, by retransmitting some packets. So we certainly do expect a lower goodput than the "none" strategy, where we keep sending along the backup path.

- **Optical:** Here we effectively will have packet losses: as soon as LSR2 detects a signal again coming from LSR1, it will forward this and discard whatever is still coming from LSR5. The bigger d is, the more packets will get lost, but clearly no duplicate ACKs will be generated due to out-of-order delivery. Still, we again expect lower goodput values than with the non-revertive strategy ("none").

From this qualitative discussion alone, it is not clear yet which of the strategies (electrical or optical) will be the better. Obviously, for strategy "none" the value of d has no impact on the goodput evolution.

When the RTT difference is negligible (d=1 ms, 2 ms), there is hardly any noticeable difference between the three cases: for the "electrical" case, we will have no losses at the merging point LSR and the number of out-of-order deliveries will be very limited; for the "optical" case, the number of losses at the merging point will be very small. However, for d in $\{10\,\text{ms}, 20\,\text{ms}\}$, we see already a different behavior for the "electrical" and "optical" cases. The electrical case shows a tiny drop in TCP goodput (around 5% of the link bandwidth in the evolution graphs), due to some TCP flows going temporarily to the fast retransmit/fast recovery phase. However, since it's due to out-of-order delivery rather than packet losses, TCP recovers rather quickly. For the "optical" case, the drop is more pronounced, as it is the result of packet losses rather

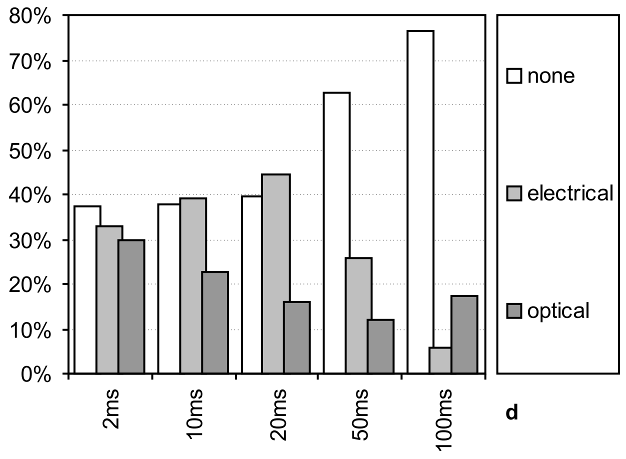


Fig. 15. Bar graph showing, for each d, the percentage of simulation cases where a particular strategy was best. E.g. a bar with height y% for "none" at d=1 ms means that for y% of the simulations with d=1 ms, "none" resulted in the best value of Good(.,d) (i.e. the goodput during (10 s, 11.5 s]).

than out-of-order delivery. When increasing the difference in RTT to bigger values (d in $\{50\,\text{ms}, 100\,\text{ms}\}$), drops also occur in the electrical case (due to buffer overflow). Additionally, there are some out-of-order deliveries of packets. At least for 100 ms, the joint effect of these phenomena apparently results in a worse goodput behavior than in the optical case.

To allow a more concise comparison, we again have calculated the goodput during the first 1.5 seconds after the switch-back operation (which covers the entire period where "optical" and "electrical" cases differ significantly). In Fig. 15, we show the fractions of the simulation cases where a particular strategy proved to be the best one. When the difference in round trip time is small (small d), it is not easy to distinguish what strategy is best. However, for big differences, the trend is clear: the strategy "none" is best. This means that only when the difference in RTT becomes large, the penalty of having a switch-back operation seems to be large enough. To distinguish between the "optical" and "electrical" cases is not evident from this figure, yet it seems to confirm that for big RTT changes (d=100 ms), the optical scenario is slightly better, but for smaller RTT increases the electrical merging protection switch seems to be preferable (and not much worse than adopting a non-revertive strategy).

A more detailed comparison between the optical and electrical scenarios is possible by comparing the ratio of the goodput attained by the two cases: $Good(\text{opt}, d) / Good(\text{el}, d)$, where $Good(s, d)$ is the goodput attained by all TCP flows in (10 s,11.5 s]. This comparison is presented in Fig. 16 and the accompanying Table 6. For small RTT changes (d=1 ms, 2 ms) the electrical

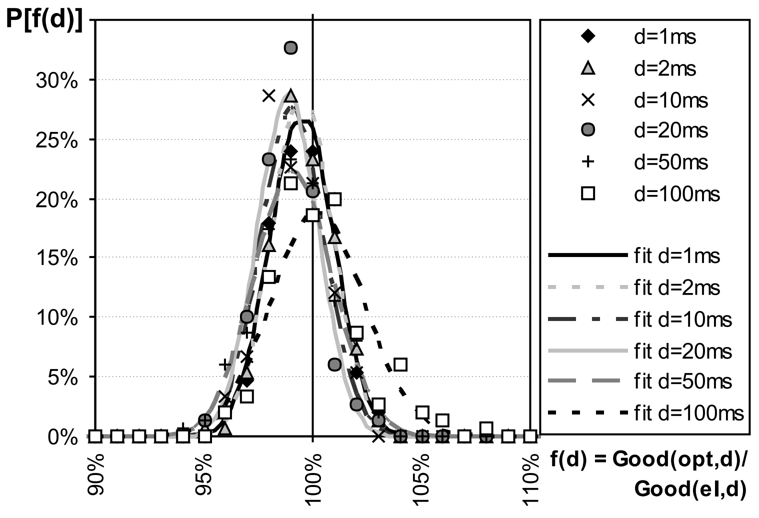


Fig. 16. Histograms (with a resolution of 1%) and normal fits for relative amount of goodput attained using "optical" strategy, compared to "electrical". A marker at (x,y) for a particular d means that y% of the simulation results had f(d) within [x,x+1%). Good(.) is the goodput attained by the total of all TCP flows in the period (10 s,11.5 s].

and optical cases are more or less equivalent. For somewhat larger differences in RTT (d=10 ms, 20 ms, 50 ms) there is a slight advantage in favor of the electrical case, and for even larger differences (d=100 ms) the optical strategy proves to be preferable. Note that the exact point where optical strategy becomes the better one depends on the buffer size for interface from LSR2 to LSR3: if this buffer were smaller, the electrical case would get worse for smaller changes in RTT (cf. smaller period of overlap when merging packet streams coming from working and backup paths will be sufficient to cause buffer overflow).

All the results confirm the conclusions of the qualitative discussion at the beginning of this section: when the difference in RTT between working and backup paths is small, the electrical approach is preferable compared to the optical scenario. However, if the change in RTT is so large that in case of electrical MPLS it results in buffer overflows at the merge point, then providing protection at the optical layer may lead to better results (TCP-wise speaking).

Table 6. Comparison of different changes in RTT. The second column gives ratio of RTT using backup paths vs. that for the primary paths. The third column represents the x-value corresponding to the average of f(d), i.e. the peak of the normal fit in Fig. 16, minus 1 (this is Good(opt,d)/Good(el,d)−1). The rightmost column indicates the percentage of simulation results where f(d)=Good(opt,d)/Good(el,d) is greater than 100%.

d (δ=5 ms)	ΔRTT	average change in goodput optical vs. electrical	% of cases where optical is better than electrical
1 ms	0.00%	−0.01%	50.00%
2 ms	+1.67%	+0.02%	46.67%
10 ms	+15.00%	−0.43%	36.00%
20 ms	+31.67%	−0.62%	29.33%
50 ms	+81.67%	−0.39%	41.33%
100 ms	+165.00%	+0.73%	59.33%

8. Case study: comparison of MPLS protection mechanisms

8.1. Purpose and simulation scenario

With the previous simulation studies, we have gained insight in the effects of protection speeds and changing path lengths on TCP behavior. In the case study presented here, we apply that knowledge to the main GMPLS protection mechanisms.

The topology used for those simulations is depicted in Fig. 17. We will consider four scenarios, corresponding with the three MPLS recovery techniques discussed in Section 2, and a loop-back variant. The protection actions that will be taken after the failure of link LSR1–LSR2 (at t=5 s, cf. scenario of Table 2) and when the link has come up again (at t=10 s) for each of the four scenarios, are the following:

- **local:** This scenario will use local protection actions. After the link failure, at t=5 s+δ, a protection switch will be carried out, as depicted in the upper part of Fig. 17, to a backup path that is d_l longer than the original working path. After the link has come up again, LSR1 (resp. LSR2) again forward packets along link LSR1–LSR2, and both flows are merged at LSR2 (resp. LSR1).

- **path:** When using path protection, signaling is necessary to inform the LSRs at the edges that they should switch. Therefore, the switch at LSR0 (resp. LSR3) will be carried out later than in the previous scenario: at t=5 s+δ+a (resp. at t=5 s+δ+c). Again, traffic will be flowing along a path that is longer than the original path (increase with d_p), and flows will be merged at LSR3 (resp. LSR0). When the link is up again, the switch-back operation again needs signaling; this implies also the switch-back operation will be carried out later than in the local protection case. Note that this is less of a problem than in the case of the switch-over case: packets still sent along the backup path some time after 10 s will not get lost (as opposed to those sent along the working path right after the failure).

- **loopback:** This is short for "local loop-back", the protection mechanism as presented in Section 2. In this case, no signaling is required: at t=5 s+δ, the protection switch is carried out. The backup path now is d_p+2$\cdot a$ longer than the original path for flows from LSR0 to LSR3 (see Fig. 17); the reverse path LSR3–LSR2–LSR3–LSR7–LSR6–LSR5–LSR0 is d_p+2$\cdot c$ longer than the corresponding original. When the link has come up again, LSR1 (and LSR2 for the reverse direction) resume forwarding along link LSR1–LSR2.

- **loopbackvar:** A disadvantage of local loop-back clearly is the presence of loops in the backup path. With this variant scenario, we combine local loop-back and path protection: at t=5s+δ, a switch to the loop-back path is carried out, and a signal is sent to the end nodes of the path (LSR0 resp. LSR3). Thus, at t=5 s+δ+a the LSR0 stops forwarding along LSR0–LSR1, but switches to the backup path as in path protection. The revertive actions after the link failure has been resolved (after t=10 s) are the same as in path protection.

Each of the above scenarios has been simulated for four sets of topology parameters, as listed in Table 7. With a+c=50 ms and b=1ms, this resulted in an average propagation RTT of 216ms. The other parameters used are the common parameters listed in Section 4.2.

8.2. Switch-over

From the description of the scenarios above, it is clear that for a given topology parameter set, **local** protection will perform better than any of the other protection schemes. Indeed, the

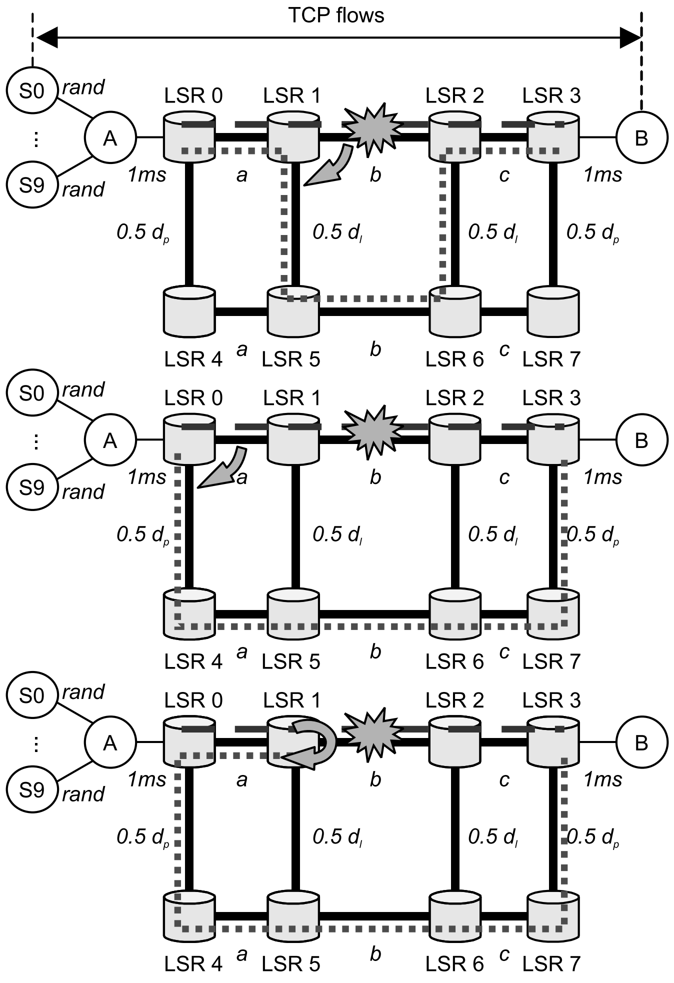


Fig. 17. Simulation topology and scenarios used to study the MPLS protection mechanisms; from top to bottom: local protection, path protection, local loop-back (for simplicity, only the path from LSR1 to LSR3 is shown, and not the reverse one). The access links had a bandwidth that was 80% of that of the backbone links. The times next to the links are the propagation delays used; for each of the access links Sx–A, it was independently chosen, using a uniform distribution, from [10 ms,100 ms].

Table 7. Topology parameters used for the different scenarios.

scenario	d_l	d_p	a	b	c
A	1 ms	3 ms	2 ms	1 ms	48 ms
B	5 ms	15 ms	10 ms	1 ms	40 ms
C	10 ms	30 ms	20 ms	1 ms	30 ms
D	20 ms	60 ms	40 ms	1 ms	10 ms

Table 8. Comparison of parameters influencing TCP behavior for different MPLS protection mechanisms.

protection mechanism	protection switch delay	ΔRTT
local	δ	$2d_l$
path	$\delta + a$	$2d_p$
loopback	δ	$2d_p + 2a + 2c$

switching time (δ) will be the smallest, and so will the increase in RTT (given by $2{\cdot}d_l$). A comparison of those two parameters for the various schemes is summarized in Table 8.

For **path** protection, the increase in RTT d_p has been chosen larger than for local protection, which we expect to lead to worse goodput figures (see Section 7). Moreover, due to an extra signaling delay, the switching time δ_P will be bigger as well ($\delta + a$ for path from LSR0 to LSR3), resulting in more losses than with local protection, again with a negative impact on TCP goodput. The local **loop-back** scheme solves the issue of faster switching time, but enlarges the increase in RTT by $2a + 2c$, i.e. twice the propagation delays along links LSR0–LSR1 and LSR2–LSR3. From the simulations presented in Section 6, an increase in RTT has a comparable diminutive impact on goodput as an increase in switching delay δ of the same order of magnitude (e.g. compare (δ=5 ms,d=100 ms) and (δ=100 ms,d=10 ms) in Fig. 14). Therefore, we expect that local loop-back in the simulated topologies will perform not much better than path protection, in terms of goodput.

The **loop-back variant**, which keeps the fast switching time δ of local loop-back, but removes the superfluous loops in the paths, may perform slightly better than local loop-back, as the eventual RTT will be smaller. However, the switch at LSR0 from the loop-back path to the path without loop is similar to the switch-back operation discussed in Section 7. Consequently, it is foreseeable that the net advantage will be diminished.

This qualitative discussion is confirmed by the simulation results. In Fig. 18 we plot the goodput evolution for the different MPLS protection mechanisms for each of the different topology parameters used. As expected, the different topologies do not result in big differences in goodput for the local protection scheme (Fig. 18(a)): the differences are similar to those observed in Fig. 12 (Section 6) for comparable changes in RTT. For path protection, we observe larger

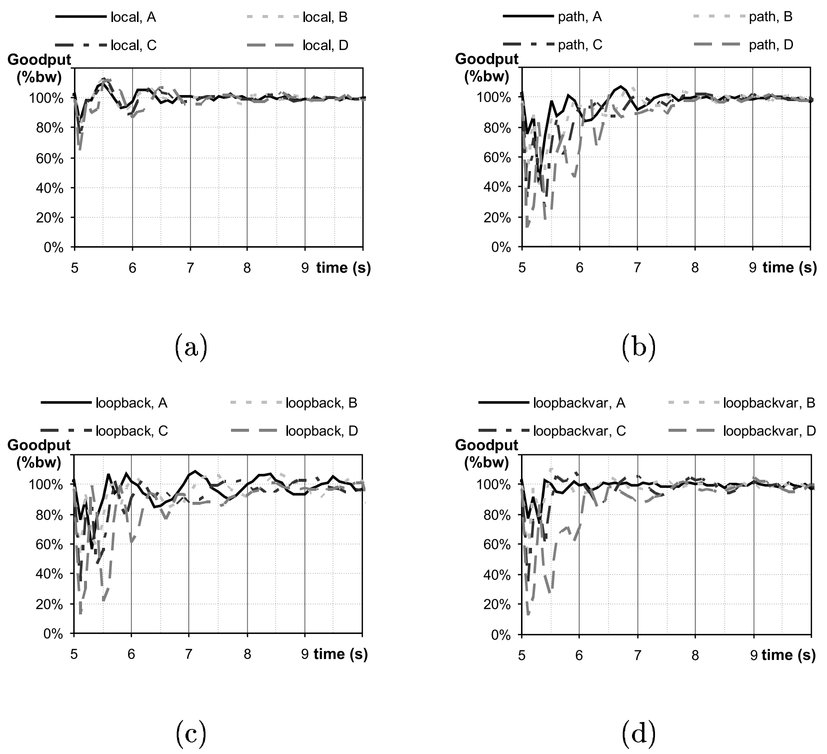
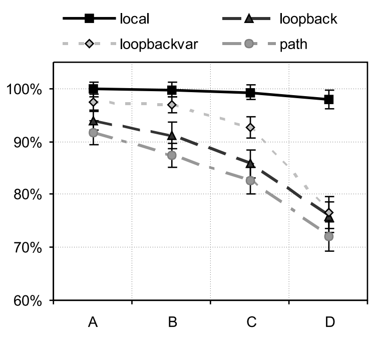


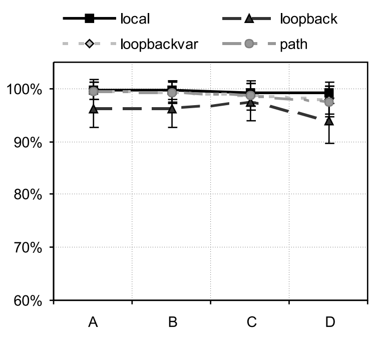
Fig. 18. Goodput evolution comparison for different protection mechanisms for each set of topology parameters (Table 7), resulting in different protection switching time (Table 8, second column). Goodput was plotted with a resolution of 100 ms (meaning that value at t gives goodput in (t−100 ms,t].

drops in goodput and slightly slower recovery when changing the topology parameters. Again, this is in accordance with our previous findings. For the loop-back scenario, we see a similar evolution: the goodput evolution graphs alone do not show remarkable differences between the loop-back and path protection schemes either.

The curves for the loop-back variant shows a more surprising behavior. For the smallest changes in RTT (scenarios A, B, C), we see a drop that is very similar to the standard loop-back mechanism, but the recovery happens faster, which is due to the reduction in RTT at the time of switching to the backup path without loops. This behavior is in accordance with our qualitative discussion, as the goodput indeed lies between local protection and local loop-back. For the last scenario D, the recovery from the drop does not seem to happen faster than in the local loop-back case. This is because in this case the overlap between the packets still flowing along backup path and those directly forwarded along LSR0–LSR4 lasts too long: packets go lost. This is the same effect we saw in Section 7 (see Fig. 15): there the penalty of the longer RTT also showed only when it was large enough to cause real losses instead of just out-of-order deliveries.



(a) Switch-over: Goodput in (5 s,7.5 s]



(b) Switch-back: Goodput in (10 s,11.5 s]

Fig. 19. Comparison of MPLS protection mechanisms. A marker for protection mechanism p and scenario s with a value of y% means that averaged over our 150 runs, Good(p,s) was y% of the maximal achievable goodput, i.e. the goodput that would be reached when every packet would be sent only once, and the access bandwidth would be fully used all the time.

A more crisp overview of this comparison is presented in Fig. 19(a), where we have compared the goodput attained by the whole of all TCP flows during (5 s,7.5 s], which is the interval where the different mechanisms show distinct behavior. In conclusion, this figure shows that for the considered topologies local protection is the best thing to do (from a TCP point of view). As choosing between local loop-back and path protection is choosing between larger switch time or longer backup paths, they do not differ that much. The local loop-back variant, which eliminates the unnecessary loop in the backup path proves to be useful only when the loop is not very big (in terms of packets that can be in transit on it) with respect to the buffer capacity in the LSR that has to merge loop-free and loop-back paths.

8.3. Switch-back

In the previous section, we focussed on the protection switch itself. As discussed before, when a switch-back is performed when the failed link has been restored to a fault-free condition, this has again impact on the TCP flows. As discussed in Section 7, the only parameter that has an impact here is the difference in RTT between working and backup paths. This is the smallest for local protection, and thus we expect it to perform best. Path protection (and the local loop-back variant) has a lower difference in RTT than local loop-back, and —as opposed to the switch-over operation— therefore will perform better in this case. This is all confirmed by the results presented in Fig. 19(b).

9. Conclusion

In future IP-over-WDM networks, GMPLS may prove to be an excellent tool to administer and control the network. This network will be layered, in the sense that it will be built using technologies having different granularities of the traffic flows, and use different switching techniques (cf. packet, timeslot, lambda, fiber, . . . switching). GMPLS also opens the way to implement different resilience mechanisms, including various protection switching techniques. In this paper, we have focussed on the effects of protection switching on TCP, one of the most important protocols used by the clients of the network, to assist in a comparative study of the various mechanisms, and the layers at which they may be implemented (esp. optical or electrical).

We have considered interaction between switched flows and other flows already present along (parts of) the backup route, and analyzed the influence of the speed of protection. From this analysis, we have concluded that only if fast protection switching is pushed to the limit (sub-50ms), this may become a problem. Therefore, it is probably not advisable to push fast protection switching to the limit: not only is it not very useful (at least from a TCP perspective) to have extremely fast protection switching, a somewhat longer delay may even prove to be better (again from a TCP user's point of view).

Furthermore, we compared the effect of the changing RTT (stemming from a longer backup path) on TCP behavior for different switching times. The change in RTT indeed has a negative impact on TCP, and the more when switching times increase. The results also pointed out that no straightforward answer can be given to the question whether it is best to have a fast protection mechanism using longer (non-optimal) backup paths, or rather a slower mechanism that finds a backup path that is of about the same length as the working path: the differences in TCP goodput are small, and depend on the exact timing and topology parameters.

We also investigated the effect of a switch-back operation performed once the network failure has been restored. We compared the cases of optical (non-merge-capable) and electrical (merge-capable) protection switching. When the difference in path lengths between backup and recovery path gets large, the advantage of merging disappears: the optical non-merge-capable technique performs slightly better.

From the joint results of these case studies, we may conclude that providing protection at the optical layer has the advantage that it avoids interaction between TCP flows between different endpoints. When working and backup paths show substantial differences in length (compared to buffer sizes at electrical interfaces), the penalty of the technology being non-merge-capable on the TCP goodput seems to be negligible. Yet, this comparison needs to be put in perspective by the study of e.g. the bandwidth requirements associated with the various mechanisms, as reported upon in [15] (showing that recovery at the optical layer suffers from higher capacity requirements, esp. for local protection).

In a final section, we presented a case study analyzing the differences between three well-known MPLS protection mechanisms. This analysis indicated that, TCP-wise speaking, local protection proved to perform best. Furthermore, local loop-back does not offer much advantage over path protection (small advantage at switch-over, slight disadvantage at switch-back). Eliminating the loop when using the loop-back mechanism is only advantageous when the loop is short enough. Again, this ranking of protection mechanisms has to be counterbalanced by criteria other than TCP goodput (e.g., [14] illustrating the expensiveness —in terms of bandwidth— of local protection, compared to other recovery mechanisms).

Acknowledgements

Part of this work has been supported by the European Commission through the IST-projects LION and DAVID, and by the Flemish Government through the IWT-project ITA/980272/INTEC. The first author is a Research Assistant of the Fund for Scientific Research – Flanders (F.W.O.–Vl., Belgium).

References

[1] Evaluating the size of the Internet, NetSizer Internet Growth Forecasting Tool, Telcordia (Online): *http://www.netziser.com*

[2] The European Information Technology Observatory, Frankfurt, Germany (Online): *http://www.eito.com*

[3] K. Claffy, G. Miller, K. Thompson: The nature of the beast: Recent traffic measurements from an Internet backbone, Proc. of INET 98, (Geneva, Switzerland, July 1998). *http://www.caida.org/outreach/papers/Inet98*

[4] S. Leinen cited on *http://www.cs.columbia.edu/~hgs/internet/traffic.html*, Feb. 2001.

[5] A. Oram, ed.: Peer-to-Peer / Harnessing the Power of Disruptive Technologies, O'Reilly, March 2001.

[6] K. Struyve, N. Wauters, P. Arijs, D. Colle, P. Demeester, P. Lagasse: Application, design an evolution of WDM in GTS's pan-european transport network, IEEE Communications Magazine, vol. 38, no. 3, (March 2000), pp. 114–121.

[7] Architecture for the Automatic Switched Transport Network, ITU-T G.ason, version 0.4, July 2001.

[8] N. Ghani, S. Dixit, T.-S. Wang: On IP-over-WDM Integration, IEEE Communication Magazine, vol. 38, no. 3, (March 2000), pp. 72–84.

[9] O. Abdul-Magd, D. Awduche, C. Brownmiller, J. Eaves, R. Hoebeke, H. Ishimatsu, M. Lazer, G. Li, M. Mayer, A. Nagarajan, L. Neir, S. Patel, E. Varma, Y. Xu, Y. Xue, J. Yates: A Framework for Generalized Multi-protocol Label Switching (GMPLS), Internet Draft, draft-many-ccamp-gmpls-framework-00.txt, Work In Progress, July 2001.
http://www.watersprings.org/pub/id/draft-many-ccamp-gmpls-framework-00.txt

[10] E. Mannie (ed.), et al.: Generalized Multi-Protocol Label Switching (GMPLS) Architecture, Internet Draft, draft-ietf-ccamp-gmpls-architecture-00.txt, Work In Progress, June 2001.
http://www.watersprings.org/pub/id/draft-ietf-ccamp-gmpls-architecture-00.txt

[11] P. Ashwood-Smith, A. Banerjee, L. Berger, G. Bernstein, J. Drake, Y. Fan, K. Kompella, E. Mannie, J.P. Lang, B. Rajagopalan, Y. Rekhter, D. Saha, V. Sharma, G. Swallow, Z. Bo Tang: Generalized MPLS – Signaling Functional Description, Internet Draft draft-ietf-mpls-generalized-signaling-06.txt, Work In Progress, October 2001.
http://www.watersprings.org/pub/id/draft-ietf-mpls-generalized-signaling-06.txt

[12] X. Xiao, L. M. Ni: Internet QoS: the big picture, IEEE Network Magazine, vol. 13, no. 2, (March 1999), pp. 8–18.

[13] D. Colle, S. De Maesschalck, C. Develder, P. Van Heuven, A. Groebbens, J. Cheyns, I. Lievens, M. Pickavet, P. Lagasse, P. Demeester: Data-Centric Optical Networks and their Survivability, IEEE Journal on Selected Areas in Telecommunications, vol. 20, no. 1, (January 2002).

[14] D. Colle, P. Van Heuven, C. Develder, S. Van den Berghe, I. Lievens, M. Pickavet, P. Demeester: MPLS recovery mechanisms for IP-over-WDM networks, Photonic Network Communications Magazine, vol. 3, no. 1/2, (January/June 2001), pp. 23–40.

[15] D. Colle, A. Groebbens, P. Van Heuven, S. De Maesschalck, M. Pickavet, P. Demeester: Porting MPLS-recovery techniques to the MPλS paradigm, Optical Networks Magazine, vol. 2, no. 4, (July/August 2001), pp. 29–47.

[16] V. Sharma, B.-M. Crane, S. Makam , .K. Owens, C. Huang , F. Hellstrand, J. Weil, L. Andersson, B. Jamoussi, B. Cain, S. Civanlar, A. Chiu: Framework for MPLS-based Recovery, Internet Draft, draft-ietf-mpls-recovery-frmwrk-03.txt, Work in Progress, July 2001.
http://www.watersprings.org/pub/id/draft-ietf-mpls-recovery-frmwrk-03.txt

[17] K. Owens, V. Makam, V. Sharma, B. Mack-Crane, C. Haung: A Path Protection/Restoration Mechanism for MPLS Networks, Internet Draft, draft-chang-mpls-path-protection-03.txt, Work in Progress, July 2001.
http://www.watersprings.org/pub/id/draft-chang-mpls-path-protection-03.txt

[18] R. Goguen, G. Swallow: RSVP Label Allocation for Backup Tunnels, Internet Draft, draft-swallow-rsvp-bypass-label-01.txt, Work in Progress, November 2000.
http://www.watersprings.org/pub/id/draft-swallow-rsvp-bypass-label-01.txt

[19] D. Haskin, R. Krishnan: A Method for Setting an Alternative Label Switched Paths to Handle Fast Reroute, Internet Draft, draft-haskin-mpls-fast-reroute-05.txt, Work in Progress, November 2000.
http://www.watersprings.org/pub/id/draft-haskin-mpls-fast-reroute-05.txt

[20] S. Floyd: A report on recent developments in TCP congestion control, IEEE Communications Magazine, vol. 39, no. 4, (April 2001), pp. 84–90.

[21] J. Postel: Transmission Control Protocol, RFC 793, Standards Track, September 1981.
http://www.ietf.org/rfc/rfc793.txt

[22] R. Braden, ed.: Requirements for Internet Hosts – Communication Layers, RFC 1122, Standards Track, October 1989.
http://www.ietf.org/rfc/rfc1122.txt

[23] S. Floyd, T. Henderson: The NewReno Modification to TCP's Fast Recovery Algorithm, RFC 2582, Experimental, April 1999.
http://www.ietf.org/rfc/rfc2582.txt

[24] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, H. Yu: Advances in network simulation, IEEE Computer, vol. 33, no. 5,(May 2000), pp. 59–67.
http://www.isi.edu/nsnam/ns/ns-research.html