A scalable and decentralized fast-rerouting scheme with efficient bandwidth sharing

Simon Balon¹, Laurent Mélon², Guy Leduc^{*}

Université de Liège, EECS Department, Research Unit in Networking (RUN), Institut Montefiore, B28, B-4000 Liège, Belgium

Abstract

This paper focuses on the protection of virtual circuits (Label Switched Paths, LSPs) in a (G)MPLS (Generalised Multi-Protocol Label Switching) network. The proposed algorithm is designed to protect traffic with strong delay requirements such as EF (Expedited Forwarding) ordered aggregates in a DiffServ domain. Indeed, for this type of application, we need fast restoration in case of failure. The duplication of all the packets in a 1+1 end-toend restoration scheme consumes a large amount of bandwidth. Furthermore, end-to-end recovery with bandwidth sharing schemes are usually considered to be far too slow. Local fast-rerouting is a solution which can compete with restoration times and bandwidth consumption offered by SONET self-healing rings. Our scheme includes a sophisticated resource aggregation mechanism based on the concepts of "backup-backup aggregation" and "backup-primary aggregation". The path selection algorithm is also designed to efficiently reduce the resource usage. Moreover, when considering LSPs at different preemption levels, our algorithm is able to correctly calculate the amount of bandwidth that can be preempted despite the sharing of resource. We show that our approach, though local, can compete with the state-of-the-art end-to-end recovery schemes in terms of resource consumption. The major contribution of our scheme, the "backup-primary aggregation", was then also used in the context of end-to-end recovery and improved its performance substantially. To be able to save a maximum amount of bandwidth in a decentralised implementation, the nodes that compute backup LSPs need to obtain a certain amount of link-state information. We propose a solution where the nodes learn almost all the information they need with RSVP messages. This drastically reduces the information that needs to be flooded in the whole network and is the first scalable decentralised solution capable of sharing a large amount of bandwidth.

Key words: Fast recovery, Resilience, Protection, Survivability, Rerouting, MPLS, Resource sharing, Backup LSP

1 Introduction

The deployment of (G)MPLS [1] and DiffServ in core networks allows ISPs (Internet Service Providers) to traffic engineer their network to improve their efficiency and to offer Quality of Service (QoS) to their customers. Indeed, in MPLS-enabled networks, it is possible to establish tunnels, called Label Switched Paths (LSPs), along which packets corresponding to a given forwarding equivalence class (FEC) are routed. LSPs can be established to "emulate" the classical hop-by-hop routing of IP, but can also be source-routed, allowing to precisely define how traffic aggregates must be routed. Signalling protocols associated with MPLS, such as RSVP-TE [2] (Resource reSerVation Protocol with Traffic engineering Extensions), support bandwidth provisioning of LSPs and offer end-to-end QoS guarantees.

Today, link and node failures are frequent [3]. When an element of the network fails, all the traffic passing through this element is lost (at least during the recovery procedure), which can really decrease the QoS perceived by all the users of the network. For this reason, there is a real need to develop algorithms and protocols that will allow the network to quickly recover from any failure it may encounter. Such algorithms were studied since virtual circuits exist, but they were mainly solved off-line with the use of optimisation theory (Linear Programming / Integer Programming problems). In on-line traffic engineering, the network must be able to compute and establish a path from an ingress router to an egress one and to protect it against failures, based on a request defining the bandwidth and QoS requirement of the traffic.

1.1 Related works

A classical way to achieve reliability is to use schemes referred to as 1+1 and 1:1 protection (in [4], V. Sharma et al. introduce these concepts in the context of MPLS). For each LSP we want to establish, we compute two completely disjoint paths from the ingress to the egress. The best of the two is the primary path, the other is the backup path. In the 1+1 scheme both paths are used simultaneously: all packets are duplicated at the ingress and sent on both paths. The egress node continuously monitors both inputs and selects the "best" one. This way of ensuring protection has the advantage of fast receiver-driven recovery upon failure but is of

^{*} Corresponding author.

Email addresses: balon@run.montefiore.ulg.ac.be(Simon Balon),

laurent.melon@bsb.com(Laurent Mélon),

leduc@run.montefiore.ulg.ac.be(Guy Leduc).

¹ Research Fellow of the Belgian National Fund for the Scientific Research (F.N.R.S)

² Work realized as Research Fellow of the Belgian National Fund for the Scientific Research (F.N.R.S)

course very costly in terms of bandwidth. An approach to limit the cost of the 1+1 protection is presented in [5]. Since sharing cannot be used, the goal is to achieve efficiency by improving path selection. In order to fullfill this objective, the authors transpose the minimum interference criteria (used in MIRA [6]) by replacing the concept of maximum flow by maximum 2-route flow. Following this criteria, they try to route demands such that the reduction of the maximum 2-route flow for the different ingress-egress pairs is as small as possible. Doing so, they hope that it will be possible to accept more demands in the future. This is confirmed by the simulations that show a low rejection ratio for this solution.

In the 1:1 scheme, only the primary path is used to forward packets while the backup path is in "standby" mode. If a failure occurs (on the primary path), a message is sent to the ingress which swaps the backup and the primary path. Obviously the 1:1 protection induces far more delay than the 1+1. Failure has to be detected, a message must propagate to the ingress node which must then swap active and standby paths. For this reason other approaches have also been envisaged. In fact restoration strategies can be divided into two classes: end-to-end recovery and local recovery (often called "fast re-routing"). In a local scheme re-routing is handled by the node directly preceding the failure on the primary path or more generally by a node "close" to the failure. The idea is to establish a set of backup LSPs each one protecting the primary path against the failure of one particular node (or link). Experimental results demonstrate that GMPLS with RSVP can be applied to optical/electrical mesh networks to yield ultra-fast provisioning and restoration times competitive with SONET rings [7].

The advantage of the 1:1 solution is that significant bandwidth saving can be realized. Indeed if we assume that only a single failure may happen in the network at any given time, not all backup paths can be activated simultaneously. Resources that must be reserved for independent backup paths can thus be shared. Some information must be made available to the device that computes primary and backup paths if we want to achieve the best possible aggregation of resources [8-10]. The authors of [9] present an algorithm able to protect the network against link failures with a relatively small amount of data. The extra bandwidth consumption is limited to 63-68%, which is a significant improvement compared to previous methods. M. Kodialam and T.V. Lakshman show in [8] that a partial information scenario which uses only aggregated and not per-path information can achieve efficient dynamic routing of locally restorable bandwidth guaranteed path. The partial information flooded in the whole network specifies the part of each reserved link bandwidth which is reserved by primary paths and the part reserved by backup paths. The same kind of reasoning as [8] in an end-to-end instead of a local protection scheme is applied in [11]. Our work is related to [8] and presents some similarities. One of the main advantages of our solution compared to the solution in [8] is that we can achieve the efficiency of the complete information scheme with nearly the cost of the partial information scheme.

Paper [12] integrates QoS constraints in the computation of backup paths. Y. Bejerano et al. consider both bottleneck QoS constraints such as bandwidth, and additive QoS constraints such as delay and jitter. In that paper, the authors provide algorithms that find a primary path satisfying QoS requirements combined with a restoration topology. A restoration topology is a set of bridges, each of which circumvents a (different) part of the primary path. The proposed solution may violate the delay constraint for restoration paths, while the primary paths always satisfy the QoS constraints. One of the key contribution of this paper is the concept of adjusted delays, which allows existing path computation algorithms to be adapted in order to identify suitable restoration topologies.

In [13], W.D. Grover and D. Stamatelakis present the concept of p-cycle. In pcycles, spare links are connected into cycles, but the method is different from selfhealing rings because each preconfigured cycle contributes to the restoration of more failure scenarios than a ring can. One p-cycle can be used for the restoration of one span *on* the cycle (like self-healing rings), but also for the restoration of one span *off* the cycle if both ends of this span are located on the cycle. This reduces the amount of bandwidth that is consumed for restoration purpose, comparing to selfhealing rings, so that the authors claim that p-cycles provide ring-like restoration speed with mesh-like capacity usage. The concept of p-cycles can be applied in IP networks [14].

1.2 Structure of the paper

We provide a solution for the problem of precomputed local rerouting paths where bandwidth is reserved for backup paths. One backup LSP protects one primary LSP against one specific failure (node or link). This scheme can protect important traffic aggregates that require rapid restoration in case of failure and cannot wait for a global restoration. Typically, traffic with strong QoS requirements need this kind of guarantee. Local restoration requires the establishment of many protection LSPs for which the bandwidth has to be reserved. This can lead to a big waste of bandwidth if not done properly. The best way to reduce the bandwidth waste is to make the assumption that two resources (link or node) cannot fail at the same time. When a failure occurs in the network, we suppose that the preceding failure has been recovered ³. This assumption allow us to share bandwidth between two backup LSPs that will never be active together because they protect different resources.

Our first contribution is an improved bandwidth sharing scheme. When a failure occurs in the network, the traffic that makes a detour frees some bandwidth on some part of the primary path. In some cases, this bandwidth can be used without

 $^{^3}$ This assumption is implicit in all the 1+1 or 1:1 protection schemes. Indeed, if we want to protect traffic against double failures in the network, we have to protect backup paths against failures as well.

additional cost by another backup LSP protecting the failed resource, as it is explained in detail in section 2. This can decrease the bandwidth cost of both local and end-to-end protection. In section 3, we present an in-depth description of our algorithm. Section 4 explains how the available bandwidth is computed on a link when preemption levels are used.

Our second contribution is a method to compute the local backup paths in a decentralised manner. Indeed, to be able to share efficiently the bandwidth and so reduce the bandwidth cost of the protection, the entity which computes the backup paths has to know a certain amount of information, as we explain in section 5. The first (naïve) idea is to flood all the information in the whole network using the TE (Traffic Engineering) extensions of the intra-domain routing protocol. As this approach is obviously not scalable, we propose an original method that is capable of sharing a large amount of bandwidth while being scalable.

Simulation results in section 6 give numerical values of the bandwidth gain which can be achieved using our technique. Finally, we conclude our work in section 7. In appendix A, we justify our design choice of section 5.

2 Algorithm overview

Our algorithm offers improvements in two main areas compared to previously presented solutions: (1) reduction of the bandwidth consumed by both local and endto-end protection schemes by improving bandwidth aggregation and (2) a method to handle preemption levels when bandwidth aggregation is used in the network. Moreover, we provide an efficient and scalable way for the computing nodes to obtain the essential information to optimise bandwidth sharing.

2.1 Bandwidth sharing

2.1.1 Backup-Backup bandwidth sharing

As already explained, resource sharing is possible under the assumption that, at any given time, at most a single failure will occur in the network. If this assumption holds, two backup LSPs protecting two distinct nodes will never be activated together. If these backup LSPs use some common links, we can reserve on these links only the maximum bandwidth requirement of both LSPs instead of their sum. Figure 1 presents this scenario. This type of bandwidth sharing can greatly decrease the bandwidth that need to be reserved for backup paths and is used in all classical 1:1 protection schemes.



Fig. 1. $Backup_1$ protects LSP_1 from failure of node N_2 . $Backup_2$ protects LSP_2 from failure of node N_3 . Since $Backup_1$ and $Backup_2$ will never be used simultaneously, they can share bandwidth on link $N_1 - N_5$.

2.1.2 Primary-Backup bandwidth sharing

It is possible to improve further the scheme if we consider that when a backup path is activated because of a failure, some bandwidth is not used any more on the primary path ([15]). Indeed as soon as the failure is detected, the node responsible for the local backup will swap service and recovery paths. Very rapidly the circumvented links of the primary path will see their bandwidth consumption reduced. This bandwidth can thus be used by other backup LSPs protecting the same failed resource. Figure 2 details the situation.



Fig. 2. The two primary LSPs (LSP_1 and LSP_2) will fail together when N_2 fails. $Backup_2$, protecting LSP_2 , can share bandwidth with LSP_1 on link N4 - N5. $Backup_1$, protecting LSP_1 , is not shown on the figure.

2.1.3 Path computation

In many traffic engineering schemes, a central server using some sort of mixedinteger programming algorithm computes an optimal mapping between requests and LSP paths. This approach usually requires hours of computation and is not very robust mainly because the global optimum discovered is very sensitive: a few changes in the set of requests lead to a completely different set of paths. In our approach, path computation is completely decentralised and real-time. The LSP requests are received sequentially by the ingress nodes that compute and establish the LSPs one after the other. Consequently our scheme can combine easily with TE (Traffic Engineering) algorithms following the same decentralised philosophy. Of course, it can also combine with a centralised scheme.

In this paper we will always assume without loss of generality that the primary LSP follows the shortest path according to a certain metric (usually a hop count). The reader is referred to [16,6,17] for other TE schemes to establish primary paths. When the primary path is known, we compute the set of backup LSPs required to prevent any possible node failure along this path. If a backup path cannot be found under the node-failure assumption⁴, we assume that only a link failure will occur and compute a new backup path. If it fails again, the request is rejected.

To use bandwidth efficiently, the path computation algorithm has to choose paths where the resource sharing is high. To compute the backup path we associate with each link a cost corresponding to the increment of bandwidth required if the backup LSP goes through the considered link. Dijkstra's algorithm is then used to compute the shortest path starting at the node preceding the protected node of the primary path towards the egress node. We stop the algorithm when it reaches a node that belongs to the primary path after the protected node.

2.2 Preemption levels

Preemption levels (see [18]) are used to define some LSPs as being "more important" than others. When establishing an LSP, it can preempt the bandwidth reserved by LSPs having a lower preemption level. In case of failure, LSPs with a higher preemption level will also be restored first.

Handling preemption levels has no impact on the bandwidth sharing efficiency. However combining resource sharing and preemption levels in the same scheme requires some special care. This problem will be explained more extensively in section 4.

⁴ Obviously it is impossible to protect the path against the failure of the egress node. However it is possible to protect the link between the penultimate node and the egress. The backup computed for this purpose only needs to be link-disjoint with the primary path.

3 In-depth description

For the clarity of the rest of this paper we will first define a few terms and functions. In figure 3, we can see the differences between the link and the node protection.



Fig. 3. Link or Node protection

The following two notations come from [4].

Definition 1 The Point Of Repair (POR) is the LSR (Label Switching Router) where the switching is done between the primary and the backup path at the moment of the failure.

Definition 2 The Path Merge LSR (PML) is the LSR where the backup path merges with the primary path.

A network is represented by a multi-valued graph $\mathcal{G} = (\mathcal{X}, \mathcal{U})$ where \mathcal{X} is a set of nodes and \mathcal{U} a set of directed links between these nodes. Each link $L_{ii} \in \mathcal{U}$ between nodes $N_i \in \mathcal{X}$ and $N_j \in \mathcal{X}$ is associated with a set of values:

- C_{ij} : the capacity of the link.
- R_{ij} : the total bandwidth reserved on the link.
- $R_{ij}[p]$: the total bandwidth reserved at preemption level p^{5} .
- $P_{ij}[p]$: the total bandwidth reserved at preemption level p for primary LSPs.

•
$$P_{ij} = \sum_{p=0}^{P-1} P_{ij}[p].$$

- $B_{ii}(L_{kn})[p]$: the total bandwidth used by backup LSPs at preemption level p in case of failure of link L_{kn} .
- $B_{ij}(L_{kn}) = \sum_{p=0}^{P-1} B_{ij}(L_{kn})[p].$
- $B_{ij}(N_k)[p]$: the total bandwidth used by backup LSPs at preemption level p in case of failure of node N_k .
- B_{ij}(N_k) = ∑^{P-1}_{p=0} B_{ij}(N_k)[p].
 F_{ij}(L_{kn})[p] : the total bandwidth freed by primary LSPs at preemption level p in case of failure of link L_{kn} . • $F_{ij}(L_{kn}) = \sum_{p=0}^{P-1} F_{ij}(L_{kn})[p]$.

⁵ In section 4 we explain how to compute $R_{ij}[p]$ such that $R_{ij} = \sum_{p=0}^{P-1} R_{ij}[p]$. Note that we do not need $R_{ij}[p], \forall p$ to compute R_{ij} (see equation 3).

- $F_{ij}(N_k)[p]$: the total bandwidth freed by primary LSPs at preemption level p in case of failure of node N_k .
- $F_{ij}(N_k) = \sum_{p=0}^{P-1} F_{ij}(N_k)[p].$

P is the number of preemption levels. In a practical implementation the source node of each link is responsible for maintaining this set of values up-to-date. Now, let us have a look at figure 4. This figure represents the bandwidth utilisation of a fixed link L_{ij} . We can see that the reserved bandwidth for the primary LSPs (P_{ij}) is the sum of the reserved bandwidth for each of the primary LSPs using this link. On the figure, we can see that there are four primary LSPs that pass on link L_{ij} $(b_1, b_2, b_3$ and b_4 are the values of the bandwidth reserved for the four primary LSPs). The bandwidth reserved for the primary LSPs is thus not shared at all. On the other hand, the bandwidth reserved for the different backup LSPs $(R_{ij} - P_{ij})$ Sfrag replacements^{is} not the sum of the reserved bandwidth for each of the backup LSPs that pass on

the considered link. Indeed, the bandwidth is shared between the backup LSPs that protect different resources. X_1 , X_2 , X_3 and X_4 are four resources (link or node) that are supposed not to fail at the same time. The total reserved bandwidth on link L_{ij} is R_{ij} . The free bandwidth on link L_{ij} is $C_{ij} - R_{ij}$.





From figure 4 and above definitions, we have:

$$B_{ij}(N_k)[p] = \sum_{\forall m: L_{mk} \in \mathcal{U}} B_{ij}(L_{mk})[p]$$
(1)

$$F_{ij}(N_k)[p] = \sum_{\forall m: L_{mk} \in \mathcal{U}} F_{ij}(L_{mk})[p]$$
(2)

$$R_{ij} = P_{ij} + \max\left(0, \max_{L_{kn} \in \mathcal{U}} \left(B_{ij}(L_{kn}) - F_{ij}(L_{kn})\right), \max_{N_k \in \mathcal{X}} \left(B_{ij}(N_k) - F_{ij}(N_k)\right)\right)$$
(3)

Equations 1 and 2 express that a node failure is equivalent to the failure of all its incoming links. We consider incoming links because it is the same upstream nodes that will activate the same backup paths in case of a node failure or in case of the failure of all its incoming links. Note that in equation 3 we have to consider the maximum over all possible link failure scenarios even if we are protecting against node failure because it is not mathematically guaranteed that the worst case bandwidth consumption will be obtained when faced with a node failure. Indeed consider the failure of link " N_3 - N_2 " on figure 2. In this scenario, both Backup₂ and LSP₁ will be used simultaneously while it is not the case if node " N_2 " goes down. Of course, in most practical situations the worst case will be a node failure.

An LSP request is composed of:

- the source or ingress node: *src*;
- the destination or egress node: dst;
- the required bandwidth 6 : bw;
- the priority: *p*.

3.1 Link state management

Each node N_i has to maintain and update the link state information for all the links that originate at node N_i . This section will explain how the values of P_{ij} , B_{ij} and F_{ij} have to be updated for all links L_{ij} when a primary or backup LSP is established. Let \mathcal{P}_{pr} be a primary LSP of preemption level p and required bandwidth bw. The path of this LSP is the ordered set $\mathcal{P}_{pr} = \{N_{y_0}, N_{y_1}, \ldots, N_{y_n}\}$, as shown on figure 5. Let \mathcal{P}_{bu} be a backup LSP. The path of this LSP is the ordered set $\mathcal{P}_{bu} = \{N_{x_0}, N_{x_1}, \ldots, N_{x_n}\}$. Let s be the index for which $N_{y_s} = N_{x_0}$ (= POR), and e be the index for which $N_{y_e} = N_{x_n}$ (= PML) i.e. the node where primary and backup paths merge. The backup path protects node $N_{y_{s+1}}$ and link $L_{y_sy_{s+1}}$.

When the primary LSP is established, links $L_{ij} \in \mathcal{P}_{pr}$ ⁷ must be updated according

⁶ In this paper we assume that a single value defines the bandwidth required by each LSP. In a DiffServ context this corresponds to using L-LSPs or E-LSPs with a single OA (Ordered Aggregate). Extensions of the presented algorithms to handle E-LSPs with multiple OAs is straightforward and will be presented in our future works. Interested readers are invited to read [19] for further information on how to combine DiffServ and (G)MPLS and for a definition of L-LSPs.

⁷ The notation $L_{ij} \in \mathcal{P}_{pr}$ denotes $\exists t : N_{y_t} = N_i \land N_{y_{t+1}} = N_j$.

$$P_{ij}[p] \leftarrow P_{ij}[p] + bw \tag{4}$$

When the backup LSP \mathcal{P}_{bu} is established, all links $L_{ij} \in \mathcal{P}_{pr} \cup \mathcal{P}_{bu}$ must be updated according to:

$$B_{ij}(L_{y_sy_{s+1}})[p] \leftarrow B_{ij}(L_{y_sy_{s+1}})[p] + bw \quad \text{if } L_{ij} \in \mathcal{P}_{bu}$$

$$\tag{5}$$

$$F_{ij}(L_{y_sy_{s+1}})[p] \leftarrow F_{ij}(L_{y_sy_{s+1}})[p] + bw \quad \text{if } (L_{ij} \in \mathcal{P}_{pr}) \land i < e \land j > s$$
(6)

PSfrag replacements

The purpose of equation 6 is to free bandwidth on the primary path between the POR and the PML in case of failure. To apply equation 6, a backup LSP must also be signalled on part of the primary path (between the POR and the PML) in case of decentralised deployment. When $P_{ij}[p]$, $B_{ij}(L)[p]$ and $F_{ij}(L)[p]$ have been updated according to equations 4 to 6, $R_{ij}[p]$ can then be recomputed by means of a procedure described in section 4.



Fig. 5. Primary and backup paths

3.2 Path computation

In this section we will explain the primary and backup paths computations. We will also highlight what kind of information is needed to make these computations. For the moment we assume that a centralised entity has access to all the link state information in the whole network. In section 5 we will explain how it is possible to distribute these computations.

3.2.1 Primary path computation

We assume that local protection will be required for important flows that are sensitive to delay and do not accept the delay of end-to-end protection in case of failure.

to:

For these flows we assume that the primary path has to be optimised on its own because it can have some strict delay constraints (the backup paths will be computed next). For example, we can use a simple Dijkstra's algorithm [20] to find the constraint shortest path ⁸ from *src* to *dst*, considering a cost of one for all the links of the network (leading to a min hop path). Note that we can also rely on any other more suitable technique to compute the primary LSPs. We are aware that optimising primary and backup paths together, instead of sequentially, could lead to a better global resource sharing. Indeed, our choice to take the primary path as a constraint for the backup paths computations limits our search to a subset of the whole state space. On the other hand, this combined method would lead to less optimised primary LSPs, which is a more severe shortcoming given that primary LSPs are used almost all the time.

The computed primary path can be described by an ordered set $\mathcal{P} = \{N_{x_0}, N_{x_1}, \dots, N_{x_n}\}$ with $N_{x_0} = src$ and $N_{x_n} = dst$.

3.2.2 Backup paths computation

Given the primary path \mathcal{P} and the node we try to protect $N_{x_{k+1}} \in \mathcal{P}$, we introduce $Inc_{ij}(N_{x_{k+1}}, bw)$ (resp. $Inc_{ij}(L_{N_{x_k}N_{x_{k+1}}}, bw)$) which represents the increase of R_{ij} when a backup LSP requiring bw units of bandwidth and protecting node $N_{x_{k+1}}$ (resp. link $L_{N_{x_k}N_{x_{k+1}}}$) uses link L_{ij} . We have $Inc_{ij}(*, bw) = R'_{ij} - R_{ij}$ where R'_{ij} is the new reserved bandwidth obtained after the new LSP establishment. If $R'_{ij} > C_{ij}$ then $Inc_{ij}(*, bw) = \infty$ (capacity constraint). R_{ij} and R'_{ij} can be calculated using equations 1, 2 and 3.

Each link L_{ij} is assigned a cost K_{ij} given by

• if we protect against failure of node $N_{x_{k+1}}$

```
if (i = N_{x_{k+1}} \lor j = N_{x_{k+1}})
K_{ij} = \infty
else if (Inc_{ij}(N_{x_{k+1}}, bw) = 0)
K_{ij} = \varepsilon
else
K_{ij} = Inc_{ij}(N_{x_{k+1}}, bw)
```

⁸ A constraint shortest path is defined as the shortest path that respects (in our case) the bandwidth constraint. It is equivalent to a shortest path on a pruned topology where all the links violating the capacity constraint are removed.

• if we protect against failure of link $L_{N_{x_k}N_{x_{k+1}}}$

$$\begin{split} &\text{if } (i = N_{x_k} \wedge j = N_{x_{k+1}}) \\ & K_{ij} = \infty \\ &\text{else if } (Inc_{ij}(L_{N_{x_k}N_{x_{k+1}}}, bw) = 0) \\ & K_{ij} = \varepsilon \\ &\text{else} \\ & K_{ij} = Inc_{ij}(L_{N_{x_k}N_{x_{k+1}}}, bw) \end{split}$$

Dijkstra's algorithm is run from root N_{x_k} until the next marked node by the procedure is N^* with $N^* \in \{N_{x_{(k+1)}}, ..., N_{x_n}\}^9$. If no valid node-disjoint path is found, then select link failure protection. If it fails once again, reject the request. We have introduced the small number ε which is used instead of zero, to favour the selection of the minimum hop path if all Inc_{ij} are null.

3.2.2.1 Needed information To compute the cost K_{ij} for all the links of the network, we need some information. We claim that R_{ij} , P_{ij} , $B_{ij}(X)$ and $F_{ij}(X)$ are sufficient to compute K_{ij} , if X is the resource we want to protect (link or node). Indeed, to compute the increment of bandwidth which would result from the establishment of the backup path on link L_{ij} , we must check whether the new value of $B_{ij}(X) - F_{ij}(X)$ is greater than the *old* value of $R_{ij} - P_{ij}$.

3.2.2.2 About optimality It is worth noting that our procedure is not optimal for two reasons. First of all, it is not a network-wide optimum. It is quite obvious because requests are treated one after the other. This means that choices made for any particular LSP will never be re-evaluated in the future. But this procedure is also not optimal at the LSP level, i.e. does not lead to find the set of LSPs minimising the increase of bandwidth reservation. Indeed, all the backup LSPs (one for each node of the primary path) are calculated one after the other. Once again the sequentiality prevents the algorithm to find an optimal solution.

Despite being sub-optimal, our simulations have shown that this algorithm was a good heuristic. We also tested an enhanced version of the algorithm. In this version, once all the backup paths of one primary path are computed, we try to improve the sharing of each of them (by changing their path) with the knowledge of all the other backup paths until no more bandwidth gain is observed. In very rare cases it leads to a global improvement on the total bandwidth consumed at the network level.

⁹ In case of node protection, the end of the backup tunnel cannot be $N_{x_{k+1}}$ because all its connected links have an infinite weight.

In all other cases it leads to no improvement at all or even to an increase in the bandwidth consumption.

Moreover the type of solution we propose is designed to be used in a dynamic environment where relatively small LSPs (compared to the links capacity) are added and removed permanently, following users needs. In this context, a form of statistical multiplexing makes the ordering of establishment less relevant.

4 Preemption levels aggregation

When combining both preemption levels and resource sharing we must be careful that $R_{ij}[p]$ must correctly reflect the amount of bandwidth which can effectively be preempted (if required) at each preemption level. Indeed, the amount of bandwidth assigned to each preemption level has to reflect the fact that by removing all LSPs at a given level a certain amount of bandwidth will be freed. For primary paths, we have to reserve at level p the sum of the bandwidth required by all LSPs at level p. The introduction of backup LSPs and bandwidth sharing makes things a bit more complex. Indeed when using protection, removing an LSP does not necessarily free any resource : if we recall equation 3, we see that a decrease of $B_{ij}(N)$ only has an impact on R_{ij} if node N is the one that maximises the difference. The consequence is that the preemption of a given quantity of bandwidth will sometimes require that we tear down a set of LSPs whose total bandwidth is bigger than the required bandwidth. To do so the LSPs we try to establish must have a preemption level higher than all the LSPs in this set.

Important remark : In this paper, preemption levels are numbered in decreasing order of priority. Level 1 is thus more important than level 2.

An example is given in tables 1 and 2. The bandwidth that must be reserved for backup LSP₁ and LSP₂ can be limited to $max(BW(LSP_1), BW(LSP_2)) = 10$ Mbps because they protect two distinct nodes. A new LSP with preemption level 1 would only be able to preempt bandwidth from LSP₁. But despite the fact that LSP₁ requested 10 Mbps of bandwidth, removing it will only free 5 Mbps because of sharing.

LSP	Failure	Bandwidth	Preemption Level
1	N_x	10 Mbps	2
2	N_y	5 Mbps	1

Table 1

Sharing with preemption levels: LSPs

As explained earlier preemption levels are used to give priority to certain LSP requests. If a link is completely filled then it is still possible to establish a new LSP

Preemption level	Bandwidth
$R_{ij}[1]$	5 Mbps
$R_{ij}[2]$	5 Mbps
R_{ij}	10 Mbps

Table 2

Sharing with preemption levels: $R_{ij}[p]$

through this link by preempting resources belonging to less important LSPs. But the bandwidth reserved on a link is the result of three terms (cf. equation 3) which are composed in different proportions for each preemption level.

The algorithm computing $R_{ij}[p]$ is composed of two phases. The first one consists of computing an intermediate result $G_{ij}(L)[p]$ and $G_{ij}(N)[p]$. The second phase computes $R_{ij}[p]$ using this result.

4.1 Phase 1

The value $G_{ij}(L)[p]$ (resp. $G_{ij}(N)[p]$) represents, up to preemption level p, the bandwidth that must be reserved on link L_{ij} to be able to forward traffic in case of failure of link L (resp. node N), in addition to the bandwidth that is reserved for primary LSPs.

$$G_{ij}(L)[p] \leftarrow \max\left(0, \sum_{k=0}^{p} B_{ij}(L)[k] - \sum_{k=0}^{p} F_{ij}(L)[k]\right)$$
$$\forall p, L: 0 \le p < P, L \in \mathcal{U}$$

$$G_{ij}(N)[p] \leftarrow \max\left(0, \sum_{k=0}^{p} B_{ij}(N)[k] - \sum_{k=0}^{p} F_{ij}(N)[k]\right)$$
$$\forall p, N : 0 \le p < P, N \in \mathcal{X}$$

The algorithm is based on the following idea: we do not need to reserve extra bandwidth at level p if, up to that level, a sufficient amount of bandwidth will be freed by the failure we consider. However we should note that $G_{ij}(X)[p]$ (X being either a node or a link) can never be negative even if $\sum F_{ij}(X)[p] > \sum B_{ij}(X)[p]$ because it would mean we have to "unreserve" bandwidth that is used by active primary paths (recall that $F_{ij}(X)[p]$ is already reserved).

Figure 6 shows the situation. Up to level 1, a failure will free more bandwidth than needed by the backup LSPs. For this reason $G_{ij}(X)[0] = G_{ij}(X)[1] = 0$. For $p \ge 2$, $\sum_{k=0}^{p} B_{ij}(X)[k] - \sum_{k=0}^{p} F_{ij}(X)[k] > 0$. The values of $G_{ij}(X)[2]$, $G_{ij}(X)[3]$ and $G_{ij}(X)[4]$ are represented graphically on the figure.

 $G_{ij}(X)[p]$ is the balance up to level p between the required backup bandwidth and the freed primary bandwidth. If $G_{ij}(X)[p_0] > 0$ for a particular p_0 , this means that we must add a new reservation of bandwidth at level p_0 to correct the difference. If we assume that the correction has already been done for all $p < p_0$, the new reservation at level p_0 must consist of $G_{ij}(X)[p_0] - G_{ij}(X)[p_0 - 1]$ units of bandwidth. PSfrag replacements

This reasoning is only for a particular failure. As any node in the network can fail, we have to define a new vector $M_{ij}[p]$ accounting for the maximum difference between the total bandwidth required and freed considering all possible failures. This is the purpose of phase 2.



Fig. 6. Preemption level selection

4.2 Phase 2

We introduce the vector $M_{ij}[p]$ given by :

$$M_{ij}[p] \leftarrow \max\left(\max_{L \in \mathcal{U}} \left(G_{ij}(L)[p]\right), \max_{N \in \mathcal{X}} \left(G_{ij}(N)[p]\right)\right) \\ \forall p : 0 \le p < P$$

This vector plays the same role as $G_{ij}(X)[p]$ but at the network-wide level. Now that we have such a failure independent value we can compute :

$$\begin{cases} R_{ij}[0] \leftarrow P_{ij}[0] + M_{ij}[0] \\ R_{ij}[p] \leftarrow P_{ij}[p] + M_{ij}[p] - M_{ij}[p-1] \\ \forall p: 0$$

This formula reflects the computations we made in the example of tables 1 and 2. It should be pointed out that the difference $M_{ij}[p] - M_{ij}[p-1]$ can be negative which looks a bit surprising at first. Indeed it means we have to reserve less bandwidth at

level p than the sum of the bandwidth requirements of all primary LSPs. In fact this just means that a certain amount of bandwidth initially reserved at level p has been upgraded to level $p_0 < p$ to be aggregated with backup LSPs.

5 The signalling problem

We will now study how the nodes can obtain the information they need to compute all the paths in a decentralised scheme. In sections 3.2.1 and 3.2.2, we saw which information is needed to compute primary paths, backup paths and the reserved bandwidth on a link. In this section, we will understand where this information is needed and the solution we propose to achieve our objective, i.e. to stay scalable while achieving optimal bandwidth sharing.

5.1 Where must the information be available?

First, we can say that the information is needed where the computations are made. If all the computations are made by a centralised server, the solution is simple: it computes all the (primary and backup) paths and thus it knows all the LSPs of the network. The objective of this section is to discuss how our proposal can be decentralised.

The **computation of the primary path** is made by the ingress node. Indeed, this node is the most appropriate because it receives the request for the creation of the LSP.

The computation of the reserved bandwidth on a link $(R_{ij}[p], \forall p)$ is made by the node (N_i) immediately upstream of the link. This computation influences the admission control of a new LSP (primary or backup) on that link.

For the **computation of the backup path**, different solutions are possible. This computation can be made by:

- The ingress of the primary LSP. In this case, the ingress computes all the backup paths that protect all the links and nodes we want to protect on the primary path. After all the computations, it forwards these backup paths to the nodes on the primary path which will establish them.
- The POR. In this case, this is the node immediately upstream of the link or node we want to protect (the POR) that computes the backup path. It is also this node that establishes the backup path.
- The node to be protected or the node immediately downstream of the link to be protected. This node computes the backup path and sends it to the POR which establishes it.

In appendix A, we compare these three solutions. We have chosen to use the third one which appears to be excellent. Indeed, as we see in this appendix, the bandwidth cost is minimum and we can extend RSVP to support this solution without requiring any additional signalling protocol.

5.2 Establishment of the LSPs

In this paragraph, we present a preview of our signalling solution. The computation of the backup paths is distributed between all the nodes of the primary path. They are computed when the RESV message¹⁰ of the primary path is sent by the egress back to the ingress node. At this time, each node, one after the other, will compute one backup path protecting itself or its upstream link. It will send the computed backup path to the upstream node. This node will establish this backup path, compute a new backup path protecting itself or its upstream link and send it to its upstream node. This operation is repeated until the ingress node of the primary path is reached. In addition to the backup path, each node must send to its upstream node the union of the links already used by previously computed backup paths, as this information will be used by upstream nodes to compute other backup paths.

Each node keeps in memory some fields of the RSVP messages it transmits. Since RSVP is a soft state protocol, the path is refreshed regularly. If the path is not refreshed, the LSP (and its associated information) disappears.

Each node which computes a backup path keeps this knowledge in memory. Besides, only a very limited amount of information needs to be flooded in the linkstate routing protocol, e.g. OSPF(-TE).

5.2.1 Establishment of a primary LSP

Each node on the path of the LSP decides whether to accept or reject the LSP. This decision is made considering the reserved bandwidth of the downstream link (capacity check). Obviously, to decide whether to accept or reject the LSP on a downstream link, the RSVP message must contain a flag specifying that it is a primary LSP. Once it is accepted, the node stores the value of the LSP's bandwidth (which updates the value of $P_{ij}[p]$).

¹⁰ This is a message of the RSVP protocol ([2]), which is used to establish LSPs in MPLS networks.

5.2.2 Establishment of a backup LSP

The same kind of admission control is also needed at the establishment of a backup LSP. As with primary paths, a flag of the RSVP message must specify the type of the LSP (i.e. backup in this case). Each node N_i has in memory the present value of $P_{ij}[p]$, $B_{ij}(X)[p]$, \forall_j, X, p . Indeed, N_i can compute these values thanks to the primary and backup RSVP messages seen at the establishment of the LSPs. To take primary-backup aggregation into account, N_i also needs to known $F_{ij}(X)[p]$. For this purpose, we propose to forward a specific message on all the nodes of the primary path between the POR and the PML. We will call this message the $F_{ij}_message$. This message must specify the primary LSP to which it is related and the resource (X) which is protected.

5.3 Computation of the LSPs

If we follow the procedure described in section 5.2, each node N_i is able to compute the primary reserved bandwidth $(P_{ij}[p], \forall p)$ and the total reserved bandwidth $(R_{ij}[p], \forall p)$ on all the links that originate at N_i . We propose that each node N_i floods $P_{ij}[p]$ and $R_{ij}[p], \forall p$ in the LSAs¹¹ of OSPF-TE together with the capacity $(C_{ij})^{12}$. Thanks to this flooding, every node in the network knows $C_{mn}, R_{mn}[p]$ and $P_{mn}[p]$ $\forall p$ and $\forall_{mn} \mid L_{mn} \in \mathcal{U}$, i.e. for all the links of the network. Notice that we flood in the network the same amount of information than the partial information scheme of [8].

5.3.1 Computation of the primary LSPs

All the nodes know the free bandwidth of all the links. So all the nodes, including the ingress node, are able to compute primary LSPs.

5.3.2 Computation of the backup LSPs

As we have seen in section 3.2.2, a node which computes a backup path has to know some information about other LSPs protecting the same resource. Our idea is the following: "If it is always the same node that computes all the backup paths protecting a certain resource, this node knows almost all he has to know to compute them!" So, we propose to associate with each resource a dedicated node which computes the backup LSPs protecting it. The information obtained by the backup path computations will be used for future backup path computations.

¹¹ LSA stands for Link State Advertisement.

¹² OSPF-TE already floods the capacity C_{ij} and the free bandwidth $C_{ij} - R_{ij}$ ([21]).

In our proposal, for a node N, the dedicated node is N itself, and for a link L_{ij} , the dedicated node is N_j . We can see on figure 7 how the POR can ask for the computation of a path. Arrow 1 means: "Compute for me a path protecting you, or similarly, protecting this link between me and you." The protected node computes a backup path and sends it back to the POR (arrow 2), which establishes it. We propose to include this message exchange in the RSVP protocol (it is theoretically possible). The arrow 1 message is included as an object in the PATH message of the primary LSP and the arrow 2 message is included in the RESV message (see example in section 5.5).



Fig. 7. Example of message exchange

With this scheme, each node N_i knows all the backup paths of the network protecting itself and its incoming links because it has computed these backup paths. More formally, each node N_i knows: $B_{mn}(L_{ji})[p]$ and $F_{mn}(L_{ji})[p]$, $\forall p, \forall_{m,n,j} \mid L_{ij} \in \mathcal{U}$ and $L_{mn} \in \mathcal{U}$; and thus $B_{mn}(N_i)[p]$ and $F_{mn}(N_i)[p]$, $\forall p \forall_{mn} \mid L_{mn} \in \mathcal{U}$ (see equations 1 and 2).

5.3.3 Data flow summary

Table 3 presents the structure of the information database at node N_i , $\forall_j \mid L_{ij} \in \mathcal{U}$, $\forall_k \mid L_{ki} \in \mathcal{U}, \forall_{mn} \mid L_{mn} \in \mathcal{U}, \forall p$. This table shows how N_i obtains information and how it is exported. Figure 8 shows the same information, but in a more convenient way. Figure 9 shows the part of the whole $B_{xx}(L_{xx})[p]$ table which is kept at each node. If M is the number of links in the network and K is the number of neighbours of node N_i then the size of the whole table would be M^2 . Out of this table, only K(2M - K) values are stored locally and $(M - K)^2$ are not used at all. Besides, none are flooded.

5.4 Simplification of signalling

The transmission of the F_{ij} _messages can be a problem. Indeed, they require an additional protocol on the primary path between the POR and the PML. But if we look a little deeper, we can see that we do not actually need them. Indeed, we can put them in primary RSVP refresh PATH messages. The only drawback of this method is that this introduces an additional delay (we must wait for the next PATH refresh message).

	Information	Obtained by	Exported
	$P_{ij}[p]$	RSVP-TE of primary LSPs	Flooded with OSPF-TE
	$P_{mn}[p], \forall m \neq i$	OSPF-TE	Kept locally
	$B_{ij}(L_{mn})[p], \forall n \neq i$	RSVP-TE of backup LSPs	Kept locally to compute $R_{ij}[p]$
	$F_{ij}(L_{mn})[p], \forall n \neq i$	$F_{ij}_message$	Kept locally to compute $R_{ij}[p]$
	$R_{ij}[p]$	Computation using	Flooded with OSPF-TE
		$P_{ij}[p], B_{ij}(L_{mn})[p], F_{ij}(L_{mn})[p]$	
	$R_{mn}[p], \forall m \neq i$	OSPF-TE	Kept locally
	$B_{mn}(L_{ki})[p]$	Computation of backup paths	Kept locally
	$F_{mn}(L_{ki})[p]$	Computation of backup paths	Kept locally
PSfrag replace	$mEnts^{(L_{kj})[p], \forall j \neq i, \forall m \neq i}$	Not needed	
	$F_{mn}(L_{kj})[p], \forall j \neq i, \forall m \neq i$	Not needed	

Table 3

Database details at node N_i



Fig. 8. Data flow at node N_i

Doing so, all the nodes between the POR and the PML regularly send primary PATH refresh messages which contain F_{ij} objects. When a backup path is closed, these nodes stop sending the F_{ij} objects in the primary PATH refresh messages. Doing so, these nodes (including the protected node) can update their database.

5.5 A simple example

In this section, we will show an example to clarify the explanations. We consider the topology of figure 10. In order to avoid complex notation, we will not mention preemption levels in this example. This does not remove any generality to our pro-



Fig. 9. Information size

posal. Implicitly, all the mentioned values are specified for the specific preemption level of the requested LSP.



Fig. 10. Topology of the example

Node 1 receives a request for the establishment of an LSP of b units of bandwidth from node 1 to node 6. Node 1 computes a primary LSP. Node 1 knows the free bandwidth on all the links because they have been flooded by the extended routing protocol (e.g. OSPF-TE). The computed path is $N_1 \rightarrow N_2 \rightarrow N_5 \rightarrow N_6$. Once the primary path is computed, node 1 establishes it. It sends a PATH message to node 2 (see figure 11a). In this PATH message, an RSVP object asking for a local restoration is added. Node 2 accepts the request and forwards the PATH message to node 5 (figure 11b). Node 5 accepts the request and forwards the PATH message to node 6 (figure 11c). Each node on the primary path has seen the PATH message so it knows that a protection LSP must be computed.

Node 6 computes a backup path that protects itself. As it is not possible, it computes a path protecting link L_{56} : $N_5 \rightarrow N_4 \rightarrow N_6$. Node 6 sends this path to node 5 as an object in the RESV message (figure 11d). Node 5 also computes a backup path

that protects itself: $N_2 \rightarrow N_4 \rightarrow N_6$. It sends this path to node 2 with the RESV message. At the same time, it can establish the backup LSP which protects link L_{56} (figure 11e) ¹³. Now, it is node 2 that receives the RESV message. This node computes a backup path that protects itself: $N_1 \rightarrow N_3 \rightarrow N_4 \rightarrow N_6$. Node 2 sends this path to node 1 with the RESV message. At the same time, node 2 establishes the backup path protecting node 5 (figure 11f). In this case, as the protected part of the primary LSP is greater than 2 links, node 2 has to send an F_{ij} _message to node 5. This message indicates that the failure of link L_{12} will free the primary bandwidth between node 1 and node 6. We can remark that although there are three backup paths, there is only one F_{ij} _message that must propagate in the network. Now, node 1 receives the RESV message from node 2 meaning that the primary LSP is established. Finally, node 1 establishes the backup path protecting node 2 (figure 11g).

We will now study in more detail when nodes store information in their memory. We can follow in table 4 which information is known at which moment by which nodes. For example, the first three lines show that when the PATH message of a primary path is forwarded by a node, this node stores this information. The fourth line shows that when a node computes a backup path, this node will keep a piece of information about this backup path. The fifth and sixth lines show that the nodes on the path of a backup LSP also store some information.

In table 5, we can see which information is known by each node at the end of the process. It is the same information as in table 4, but sorted by node and type. We can see that this information is in agreement with the information of table 3 and figure 8.

6 Simulation results

This simulation section is composed of two parts. The first part contains a deep analysis of the results of the simulation of our algorithm on a first topology. In the second part, we briefly present some results obtained on three other topologies.

6.1 Detailed results

For this part of the simulation, we have used a randomly generated topology which was composed of 50 nodes and 102 full-duplex links. Among the 50 nodes, 30 were chosen to act as border routers. We affected to each ingress-egress pair a probability. The topology used has been "perfectly engineered" thanks to a *Generalised*

 $^{^{13}}$ For reason of clarity, we do not show the PATH and RESV messages used for the establishment of the backup LSPs.









(c)



(b)

(d)



(g)

Fig. 11. Example details 24

Step	Node	Updated information	Obtained from
а	1	P ₁₂	RSVP (primary)
b	2	P_{25}	RSVP (primary)
c	5	P_{56}	RSVP (primary)
d	6	$B_{54}(L_{56})$ $B_{46}(L_{56})$ $F_{56}(L_{56})$	56) Local BP computation
e	5	$F_{56}(L_{56}) B_{54}(L_{56})$	RSVP (backup)
	4	$B_{46}(L_{56})$	RSVP (backup)
	5	$B_{24}(L_{25}) B_{46}(L_{25}) F_{25}(L_{25}) F_{56}(L_{25})$	25) Local BP computation
f	2	$F_{25}(L_{25}) B_{24}(L_{25})$	RSVP (backup)
	4	$B_{46}(L_{25})$	RSVP (backup)
	2	$B_{13}(L_{12}) B_{34}(L_{12}) B_{46}(L_{12}) F_{12}(L_{12})$	12)
		$F_{25}(L_{12}) - F_{56}(L_{12})$	Local BP computation
g	5	$F_{56}(L_{12})$	$F_{ij}_message$
	1	$F_{12}(L_{12}) B_{13}(L_{12})$	RSVP (backup)
	3	$B_{34}(L_{12})$	RSVP (backup)
	4	$B_{46}(L_{12})$	RSVP (backup)

Table 4

Evolution of the information transmission

maximum concurrent flow algorithm. By "perfectly engineered", we mean that a load of 100% is reachable throughout the network if the ingress-egress probabilities are respected and the LSP-bandwidth are infinitesimal. This has been done because we realized it was useless to engineer the traffic on a network with engineering inconsistencies such as huge links following very small ones (they can only reach a very small relative load). And indeed, the *Generalised maximum concurrent flow* approximation we used is close to the behaviour used by some network engineers we have met (*e.g. "double up the link capacity when it reaches 50% of load"*). Network engineering in the context of fault protection is still a very active domain of research (cf. [22,23]).

The most important value when designing a restoration scheme is the "cost" of such a protection in terms of additional bandwidth reserved for backup LSPs. We will call it "network oversubscription" and represent it by γ . It is given by:

$$\gamma = \frac{\sum_{L_{ij} \in \mathcal{U}} R_{ij} - \sum_{L_{ij} \in \mathcal{U}} P_{ij}}{\sum_{L_{ij} \in \mathcal{U}} P_{ij}}$$

 γ measures the network-wide bandwidth reservation increase caused by the backup LSPs compared to the unprotected case.

Node	Туре	Information		
1	P_{ij}	P ₁₂		
	$B_{ij}(L)$	$B_{13}(L_{12})$		
	$F_{ij}(L)$	$F_{12}(L_{12})$		
2	P_{ij}	P_{25}		
	$B_{ij}(L)$	$B_{24}(L_{25})$		
	$F_{ij}(L)$	$F_{25}(L_{25})$		
	$B_{mn}(L_{ki})$	$B_{34}(L_{12}) B_{46}(L_{12}) B_{13}(L_{12})$		
	$F_{mn}(L_{ki})$	$F_{12}(L_{12})$ $F_{25}(L_{12})$ $F_{56}(L_{12})$		
3	$B_{ij}(L)$	$B_{34}(L_{12})$		
4	$B_{ij}(L)$	$B_{46}(L_{56}) B_{46}(L_{25}) B_{46}(L_{12})$		
5	P_{ij}	P_{56}		
	$B_{ij}(L)$	$B_{54}(L_{56})$		
	$F_{ij}(L)$	$F_{56}(L_{56})$ $F_{56}(L_{12})$ $F_{56}(L_{25})$		
	$B_{mn}(L_{ki})$	$B_{24}(L_{25}) B_{46}(L_{25})$		
	$F_{mn}(L_{ki})$	$F_{25}(L_{25})$		
6	$B_{mn}(L_{ki})$	$B_{54}(L_{56}) B_{46}(L_{56})$		
	$F_{mn}(L_{ki})$	$F_{56}(L_{56})$		

Table 5

State of the tables of the nodes

Four algorithms for node-failure protection are presented in the following results. The first one (labelled "LOCAL") is as the name suggests a local recovery scheme using only the basic "backup-backup" bandwidth sharing scheme. The second one, "LOCAL with FBW¹⁴", is an enhanced version taking into account the concept of "primary-backup aggregation" (using the vector $F_{ij}(F)$). The same difference exists between the two algorithms labelled respectively "END-TO-END" and "END-TO-END with FBW". It should be noted that the "END-TO-END" algorithm is similar in its behaviour to the algorithm presented in [9] which we consider to be the state-of-the-art in end-to-end resource aggregation.

6.1.1 Results

Figure 12 presents the evolution of γ when we progressively add LSPs to the network. The vertical position of each algorithm is not surprising. However it should

¹⁴ FBW is an acronym for Freed Bandwidth.

be noted how the introduction of vector F improves the performance of the recovery. In terms of resource consumption, local restoration with FBW is as good as end-to-end recovery without FBW. On this topology, the "distance" between the best local approach and the best end-to-end scheme is less than 10%, a price we consider quite cheap to benefit from very short restoration delays.

The decrease of γ that occurs after the establishment of 2000 primary LSPs should be pointed out. This behaviour is due to the method to choose the path of the primary LSP. Indeed, as the primary always follows the minimum hop path, many primary LSPs will tend to overlap while enough bandwidth remains available on this minimum hop path. This tends to create regions where links are used to protect only a small number of nodes. This situation does not create a lot of opportunities to aggregate bandwidth. If we take a look at figure 13, we will see that when 2000 primary LSPs and corresponding backups are established in the network, the mean reserved bandwidth is close to 45%. This mean load suggests that some links are completely filled. The following requests thus have to make a detour to avoid the saturated links. Because of this, backup LSPs are now established in other parts of the network where a more important sharing can be realized.

This is confirmed by figure 14 which shows the mean number of non-null elements in vector $F_{ij}(F)$, i.e. the evolution of $\frac{1}{|\mathcal{X}|} \sum_{F \in \mathcal{X}} sign(F_{ij}(F))^{15}$. Despite not being shown in this paper the same kind of behaviour is observed for the density of vector $B_{ij}(F)$. The slope of the curve increases shortly after 2000 primary LSP establishments indicating that backup LSPs are now using links where no bandwidth has been reserved for protecting the same node. This suggests that choosing our primary paths in a smarter way could have a big impact on the amount of sharing.

This simulation proves the interest of including restoration mechanism in the MPLS layer. Indeed lower layer recovery schemes such as SONET self healing rings impose a high level of oversubscription (> 100 %, see [24] for details).



Fig. 12. Evolution of network oversubscription

 $^{^{15}}$ sign(x) is equal to 1 if x is positive, to -1 if x is negative and to 0 if x is equal to 0.



Fig. 14. Evolution of the number of non-null elements in vector F_{ij} , averaged over all node failures F

6.2 Results on other kinds of topologies

We have used three other topologies. The first two topologies were generated using the BRITE topology generation tool [25]: one using Waxman's method [26] and the other one using Barabasi-Albert's [27]. The third one is the topology of an operational network. The Waxman topology is composed of 50 nodes and 100 full-duplex links. We set the value for parameters α and β to 0.15 and 0.2. The Barabasi-Albert topology is composed of 50 nodes and 97 full-duplex links. The operational network is composed of about 20 nodes and 40 full-duplex links. For the Waxman (WAXMAN) and the Barabasi-Albert (BA) topologies, we have generated an LSP between each pair of nodes. The size of an LSP is chosen according to a uniform random distribution between 5 and 10 units of bandwidth (for comparison, all the links of both topologies have a bandwidth of 1000 units). For the operational network, we have used real traffic measurements (represented in a traffic matrix) to fix the size of the LSP requests. The procedure to obtain the traffic matrix of the operational network is similar to the one described in [28]. The primary paths were computed with the DAMOTE algorithm [16], which is a good

	WAXMAN	BA	Operational Network
LOCAL	52.7	65.8	111.4
LOCAL with FBW	50.0	63.2	108.5
E2E	48.6	62.3	93.7
E2E with FBW	43.8	57.4	83.4

Table 6

Oversubscription

primary path computation algorithm according to simulations in [28,29].

Table 6 shows the oversubscription values for the three topologies. These values are given after the establishment of all the LSPs leading to a mean reservation of about 40% for WAXMAN and BA topologies and 20% for the operational network topology. We can notice that absolute values for the oversubscription on the WAX-MAN and BA topologies are close to the values found in the previous section. On the smaller operational network topology, we notice that the oversubscription level is higher. But even in this case, compared to SONET restoration for which the oversubscription is over 100% in all cases and does not protect against node failures, these results are competitive. Finally, The relative reduction of bandwidth when primary-backup aggregation is used (i.e. with FBW) can go up to 10.3% in the case of end-to-end protection. This means that this kind of aggregation is not limited to local restoration and provides good results for end-to-end protection as well.

7 Conclusion and future work

The contribution of this paper is threefold. First of all we improved the best known bandwidth sharing scheme without sacrificing simplicity. This new aggregation technique is able to provide a substantial decrease of the network oversubscription of both local and end-to-end protection schemes. The second interest of this paper is to explain the modification required to handle correctly the notion of "preemption levels". The third contribution is to provide a scalable way to implement our efficient bandwidth sharing solution in a distributed way.

Our results show that fast-rerouting is a viable approach to protect traffic that can only accommodate very short interruptions. They also suggest that routing the primary path in a smarter way could help reduce the resource usage further. This topic will probably be an active domain of research for our future work.

Acknowledgment

This work has been partially supported by the European commission (ATRIUM project and E-NEXT NoE) and the Walloon Region (TOTEM project).

References

- [1] Eric C. Rosen, Arun Viswanathan, and Ross Callon. Multiprotocol Label Switching Architecture, rfc3031. www.ietf.org, January 2001.
- [2] D. Awduche and al. Applicability statement for extensions to RSVP for LSP-tunnels, rfc3210. www.ietf.org, December 2001.
- [3] Athina Markopoulou, Gianluca Iannaccone, Supratik Bhattacharyya, Chen-Nee Chuah, and Christophe Diot. Characterization of Failures in an IP Backbone Network. In *Proc. of IEEE INFOCOM 2004*, volume 23, pages 2307 – 2317, Hong Kong, March 2004.
- [4] V. Sharma, Metanoia, and F. Hellstrand. Framework for Multi-Protocol Label Switching (MPLS)-based Recovery, rfc3469. www.ietf.org, 2003.
- [5] Koushik Kar, Murali Kodialam, and T.V. Lakshman. Routing Restorable Bandwidth Guaranteed Connections using Maximum 2-Route Flows. In *Proc. of IEEE INFOCOM 2002*, volume 21, pages 113 – 121, 2002.
- [6] Murali S. Kodialam and T. V. Lakshman. Minimum Interference Routing with Applications to MPLS Traffic Engineering. In *Proc. of IEEE INFOCOM 2000*, pages 884–893, 2000.
- [7] Guangzhi Li, Jennifer Yates, Robert Doverspike, and Dongmei Wang. Experiments in Fast Restoration using GMPLS in Optical / Electronic Mesh Networks. In *Optical Fiber Commun. Conf.*, March 2001.
- [8] Murali Kodialam and T.V. Lakshman. Dynamic Routing of Locally Restorable Bandwidth Guaranteed Tunnels using Aggregated Link Usage Information. In *Proc.* of *IEEE INFOCOM 2001*, pages 376 – 385, April 2001.
- [9] Guangzhi Li, Dongmei Wang, Charles Kalmanek, and Robert Doverspike. Efficient Distributed Path Selection for Shared Restoration Connections. In *Proc. of IEEE INFOCOM 2002*, volume 21, pages 140 – 149, June 2002.
- [10] Chunming Qiao and Dahai Xu. Distributed Partial Information Management (DPIM) Schemes for Survivable Networks - Part I. In *Proc. of IEEE INFOCOM 2002*, volume 21, pages 302 – 311, June 2002.
- [11] Murali Kodialam and T.V. Lakshman. Dynamic Routing of Restorable Bandwidth-Guaranteed Tunnels Using Aggregated Network Resource Usage Information. *IEEE/ACM Transactions on Networking*, 11(3):399 – 410, June 2003.

- [12] Yigal Bejerano, Yuri Breitbart, Ariel Orda, Rajeev Rastogi, and Alexander Sprintson. Algorithms for Computing QoS Paths with Restoration. In *Proc. of IEEE INFOCOM* 2003, volume 22, pages 1435 – 1445, March 2003.
- [13] Wayne D. Grover and Demetrios Stamatelakis. Cycle-Oriented Distributed Preconfiguration: Ring-like Speed with Mesh-like Capacity for Self-planning Network Restoration. In *Proc. of IEEE ICC 1998*, pages 537 – 543, Atlanta, June 1998.
- [14] Demetrios Stamatelakis and Wayne D. Grover. IP Layer Restoration and Network Planning Based on Virtual Protection Cycles. *IEEE Journal on selected areas in communications*, 18(10), 2000.
- [15] L. Mélon, F. Blanchy, and G. Leduc. Decentralized local backup LSP calculation with efficient bandwidth sharing. In *Proc. of 10th International Conference on Telecommunications (ICT'2003)*, pages 929–937, Papeete, Tahiti, 23-28 Feb. 2003. IEEE Press.
- [16] F. Blanchy, L. Mélon, and G. Leduc. An efficient decentralized on-line traffic engineering algorithm for MPLS networks. In J. Charzinski, R. Lehnert, and P. Tran-Gia, editors, *Proc. of 18th International TELETRAFFIC CONGRESS - Providing Quality of Service in Heterogeneous Environments*, volume 5a, pages 451–460, Berlin, Germany, 31 Aug.-5 Sep. 2003.
- [17] Richard Rabbat, Kenneth P. Laberteaux, Nirav Modi, and John Kenney. Traffic engineering algorithms using MPLS for service differentiation. In *Proc. of IEEE ICC* 2000, pages 791–795, June 2000.
- [18] F. Blanchy, L. Mélon, and G. Leduc. A Preemption-Aware On-line Routing Algorithm for MPLS Networks. *Telecommunication Systems*, 24:187–206, 2-4, Oct.-Dec. 2003.
- [19] F. Le Faucheur and al. Multi-Protocol Label Switching (MPLS) Support of Differentiated Services, rfc3270. www.ietf.org, May 2002.
- [20] Dijkstra E. W. A note on two problems in connection with graphs. Numerische Mathematik, 1959.
- [21] D. Katz, K. Kompella, and D. Yeung. Traffic Engineering (TE) Extensions to OSPF Version 2, rfc3630. www.ietf.org, September 2003.
- [22] Yu Liu, David Tipper, and Peerapon Siripongwutikorn. Approximating optimal spare capacity allocation by successive survivable routing. In *Proc. of IEEE INFOCOM* 2001, pages 699–708, April 2001.
- [23] Yu Liu and David Tipper. Spare capacity allocation for non-linear cost and failuredependent path restoration. In Proc. of the Third International Workshop on Design of Reliable Communication Networks, DRCN 2001, Budapest, Hungary, October 7–10 2001.
- [24] Wayne D. Grover. *Mesh-Based Survivable Networks : Options and Strategies for Optical, MPLS, SONET, and ATM Networking.* Prentice Hall, 2003.

- [25] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers. BRITE: An Approach to Universal Topology Generation. In Proc. of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems - MASCOTS '01, Cincinnati, Ohio, Août 2001.
- [26] B.M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1671–1622, December 1988.
- [27] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999.
- [28] S. Balon, O. Delcourt, J. Lepropre, F. Skivée, and G. Leduc. A traffic engineering toolbox and its application to the GÉANT network. *submitted to IEEE eTransactions on Network and Service Management*, 2005.
- [29] F. Skivée, S. Balon, O. Delcourt, G. Leduc, and J. Lepropre. Architecture d'une boîte à outils d'algorithmes d'ingénierie de trafic et application au réseau GÉANT. In *Colloque Francophone sur l'Ingénierie des Protocoles (CFIP 2005)*, Bordeaux, France, March 29-April 1st 2005.

A Comparison to other possible signalling schemes

We will study in detail the three different solutions of section 5.1 and justify our choice. They differ in the node that computes the backup LSPs.

A.1 Presentation of the three possible locations for backup path computation

A.1.1 Computation by the ingress of the primary LSP

It is the ingress of the primary LSP that computes the backup paths protecting the links and nodes of the primary path. We consider two propositions to achieve this goal. The first proposition is that each node floods all the information (B..(L..)) and F..(L..)) in the whole network. Doing so, every node of the network (including the ingress node) can compute the backup paths. But this proposition is really not scalable.

The second proposition is that each node keeps the information concerning itself and sends it to the ingress node. Thus, firstly, the ingress node computes the primary LSP and establishes it. Secondly, each node on the primary path sends the information it owns to the ingress to allow it to compute the backup paths. When the backup paths are computed, the ingress sends each POR the backup path it has to establish. Furthermore, the POR forwards the backup path to the protected node so that it can keep this information in memory. The protected node has to be aware of this information because it will send it to the ingress of the future primary LSPs passing on it. With this scheme, each node N_i knows all the backup paths protecting itself, all the backup paths protecting the incoming links and all the backup paths using itself. Formally, this information is: $B_{mn}(L_{ji})$ and $F_{mn}(L_{ji})$, $\forall_{mn} \mid L_{mn} \in \mathcal{U}, \forall_j \mid L_{ji} \in \mathcal{U}$ and thus $B_{mn}(N_i)$ and $F_{mn}(N_i)$, $\forall_{mn} \mid L_{mn} \in \mathcal{U}$ (see equations 2 and 1).

A.1.2 Computation by the POR

It is the POR which computes the backup path. The POR is the node immediately upstream of the resource to protect. In case of link protection, there is only one node per protected link that can be the POR for the backup LSP. On the other hand, in case of node protection, there are multiple nodes that can potentially be a POR for a backup LSP protecting a particular node. Indeed, each neighbour of the node can be the POR. So, every potential POR must store the information about the node. Thus, after having established a backup LSP (see figure A.1), the POR sends the new information to the protected node (arrow 1) which forwards it to all its neighbours except the POR (arrows 2).



Fig. A.1. Example of exchange of messages in the case A.1.2

With this scheme, each node *i* knows ($\forall k$ which is *i* or a neighbour of node *i*): $B_{mn}(L_{jk})$ and $F_{mn}(L_{jk})$, $\forall_{mn} \mid L_{mn} \in \mathcal{U}, \forall_j \mid L_{ji} \in \mathcal{U}$ and thus $B_{mn}(N_k)$ and $F_{mn}(N_k)$, $\forall_{mn} \mid L_{mn} \in \mathcal{U}$ (see equations 2 and 1).

A.1.3 Computation by the node to protect

It is the solution we have chosen. This has already been explained in section 5.3.2.

A.2 Evaluation of the performance of the three solutions

In this section, we will estimate the cost of the three solutions. For these estimations, we need to introduce some new notations:

- *l* is the number of nodes on the primary path
- $x = |\mathcal{X}|$ i.e. the number of nodes in the network
- $u = |\mathcal{U}|$ i.e. the number of links in the network

- *s* is the mean size of the backup paths
- t is the mean degree of the nodes, i.e. the number of neighbours of the node
- C_1 and C_2 are constant values representing the size in bytes of the representation of (respectively) one record in the database and the identifier of one node.

The bandwidth cost of a message is computed this way:

size_of_the_message * number_of_links_crossed_by_the_message.

A.2.1 First solution

In the first solution, the ingress must get the information from all the nodes of the primary path. The information for one node N_i is $\begin{cases} B_{mn}(N_i) - F_{mn}(N_i) \\ B_{mn}(L_{ji}) - F_{mn}(L_{ji}) \end{cases}$, $\forall m, n. N_j$ is supposed to be the node preceding N_i on the primary path. Thus, the size of the information to transmit to the ingress node 16 for each node on the primary path is $2*u*C_1$. The information of the node just downstream the ingress node has to cross (l-1) links to reach the ingress, but the information of the node is: $1+2+\ldots+(l-1)=\sum_{k=1}^{l-1}k=\frac{(l-1)*l}{2}$. The total cost is $\frac{2*u*C_1*(l-1)*l}{2}$ for all the messages that go from each node of the primary path to the ingress node.

When the ingress node has computed all the backup paths, it has to send them to the PORs because it is the PORs that will establish them. The cost of one path message is $s * C_2$. So, with similar arguments like above, the total cost of these messages is $\frac{s*C_2*(l-1)*l}{2}$.

In conclusion, the total bandwidth cost of this first solution is $\frac{(s*C_2+2*u*C_1)*(l-1)*l}{2}$. Furthermore, this solution requires an additional signalling for every node to send its information to the ingress and for the ingress to send the computed backup path to the nodes on the primary path.

A.2.2 Second solution

The POR can compute the backup path. After having established the backup path, it has to transmit the computed path to the downstream node. This node will in turn transmit this information to its neighbours (see figure A.1). The bandwidth cost of this operation is $s * t * C_1$. This operation requires an additional signalling.

¹⁶ Here, we do not take into account the fact that in case of lightly loaded networks, a high number of B and F components may be equal to zero.

A.2.3 Third solution

The POR asks to the downstream node to compute the backup path protecting it. After the computation, the protected node sends the path to the POR which establishes it. The bandwidth cost of this operation is $s * C_2$. This operation may require an additional signalling protocol. But as we have seen, we can extend RSVP to support this scenario.