Routing in a MPLS network featuring preemption mechanisms

François Blanchy, Laurent Mélon*, Guy Leduc

Research Unit in Networking
Electrical Engineering and Computer Science Department
University of Liège

{blanchy,melon,leduc}@run.montefiore.ulg.ac.be

Abstract – In the context of Multiprotocol Label Switching (MPLS), we propose an integration of a computationally efficient preemption mechanism into a very dynamic Label Switched Paths (LSPs) routing scheme described in a previous paper. This preemption scheme includes a quick heuristic able to select the most interesting LSPs to reroute from a link. The efficiency of this algorithm is obtained through a very rigid but seemingly appropriate policy concerning which LSPs are appropriate for preemption. A feedback from preemption on the routing process is proposed, in order to regulate the rate of the reroutings and to tune the balance between stability and continuous reorganization in a network. This integrated traffic engineering entity is tested on the accuracy of the heuristic, the relevance of the feedback and the whole integration, the influence of the routing policy on the preemption mechanism and the reliability of the managed network.

Keywords : MPLS, Traffic engineering, preemption

1 Introduction

With Multiprotocol Label Switching (MPLS), good opportunities exist to control and regulate traffic trunks in a network by routing them along Label Switched Paths (LSPs). Those LSPs can be provided with explicit routes, making the Traffic Engineering possibilities much wider. They can also be given certain guarantees on their allocated bandwidth along with QoS properties and a priority or precedence level. The latter allows the Traffic Engineering entity (the routing mechanism) at work in the topology to remove or cripple some disposable LSPs to favor more important ones, for example to keep the integrity of the bandwidth guarantees on a link. There is then the possibility for some LSPs to preempt others of lesser priorities which must be rerouted.

Some protocols like RSVP-TE [1] or CR-LDP [2] now make this possibility of explicit routing a reality, and a lot of work is being done in the field of traffic engineering on the optimal organization of the LSP mapping ([3], [4] and [5] are a few examples). Fewer work has been done on preemption mechanisms, and, to our knowledge, even less on the relationship between the routing and preemption procedures and the impact of this on the overall performance. Two major papers on this topic, [6] and [7], center on the selection of the LSPs to reroute from a link once the amount of bandwidth to preempt is known. [6] also gives some inkling on the overall impact in terms of network availability. [7] additionally offers the interesting possibility for some LSPs to reduce their transmission rate rather than being simply cast out.

In this paper we want to focus on the integration of a preemption mechanism in a generic LSP routing scheme. The latter comes from [8] and will be briefly presented in section 2. From our point of view, this integration required two main developments that will be looked upon in section 3. First, an efficient algorithm is needed to select the preempted LSPs on a link. Indeed, we feel that the flexible algorithm proposed in [7] is too costly for a task that is often trivial. By adding some rigidity to the preemption policy, we are able to make a substantial gain in computational efficiency. Secondly, we will introduce a feedback of the preemption mechanism on the route selection. Indeed, the preemption of a weaker LSP causes its rerouting, which has a practical cost in terms of network management resources. For this reason, we want to have the possibility to penalize in the routing procedure the use of a link where preemption is needed. This will enable us to play on the balance between stability and perpetual reorganization of the network. We will finally present in section 4 a collection of results that, we hope so, will illustrate the relationship between routing and preemption and the relevance of the integrated approach. Our conclusions will close this document.

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

2 Background : a generic LSP routing scheme

Our work in the field of MPLS focuses on the development of an effective *on-line decentralized* scheme for routing LSPs throughout a given network ([8]).

In an *on-line* approach, the LSP mapping is done in an incremental way, all LSPs being added to the network one after the other in the order of arrival. When computing a route, there is no awareness of any other LSP running through, only a synthetic knowledge of each link state in the topology. On the contrary, an *off-line* procedure is aware of all LSPs inside the network and computes the best (re)organization for all of them at the same time (a very expensive task). Both approaches can be implemented in either a *centralized* (the computation is done by a server) or a *decentralized* way (the computation is done at an ingress router). Of course, the *on-line* way of thinking is rather meant for a *decentralized* scheme, whereas the *off-line* one is more suitable for a *centralized* implementation because of the computing power required.

A generic on-line LSP routing scheme can usefully be split into three independent components : a *score function* used to compare route "qualities", a *constraint predicate* that tells whether a link can be used to carry the new LSP and of course a computation algorithm based on the first two components able to choose a path for the LSP.

2.1 The score function

The value of this function has a network-wide meaning and depends on the synthetic link states throughout the topology. This is the image of the objective function of the whole optimization procedure. Basically, this component allows a network operator to stress what kind of criteria should be favored throughout the network (*e.g.* low blocking probability thanks to a load balancing score, low delay and minimal use of the resources with a shortest path,...). In this document we will be concerned with two kinds of score functions : a classical shortest path involving the number of hops and a load balancing measure. The first is straightforward but the latter should require an explanation.

Let $L_{(i,j)}$ be the load on the link from a node i to a node j and $C_{(i,j)}$ be the capacity of this link. $\mathcal{U}$ is the set of all links in the topology.We have

$$\sum_{(i,j)\in\mathcal{U}} \left(\frac{L_{(i,j)}}{C_{(i,j)}} - \overline{\left[\frac{L}{C}\right]} \right)^2 + \alpha \sum_{(i,j)\in\mathcal{U}} \left(\frac{L_{(i,j)}}{C_{(i,j)}} \right)^2$$

as the load balancing function, with

$$\overline{\left[\frac{L}{C}\right]} = \frac{1}{U} \sum_{(i,j)\in\mathcal{U}} \frac{L_{(i,j)}}{C_{(i,j)}}$$

The α parameter allows us to play on the *"load-balancing versus traffic minimization compromise"* : a low α will favor longer paths in order to smooth the relative load deviation throughout the network, whereas a greater α will try to minimize the traffic.

In the case of $\alpha = 0$ it gives a low blocking probability by avoiding single points of congestion.

2.2 The constraint predicate

This component allows us to set the constraints on the optimization procedure. In particular, it can be used to specify that the capacity threshold on a link must not be exceeded when adding the new LSP. Let $L_{(i,j)}$ be the load on the link from a node i to a node j and $C_{(i,j)}$ be the capacity of this link, BW is the required bandwidth of the new LSP we have obviously

$$L_{(i,j)} + BW \leq C_{(i,j)}$$

as a clausis of the predicate. It can also be used to some extent to take care of QoS, overbooking, restoration and, as we will see, preemption and rerouting.

2.3 The computation algorithm

Finding a path that minimizes the score function using only the links allowed by the predicate is an *NP-complete* problem. For this reason we proposed in [8] an efficient approximation based on the *Bellman-Kalaba* algorithm ([9]) which has a very short response time necessary to a highly dynamic scheme. Basically it is a Constrained Shortest Path First (CSPF), the costs on each link being the score variation that would result from adding the new LSP to that link. However this aspect is not relevant in this document.

3 Integration of preemption mechanisms

Integrating preemption mechanisms in the generic scheme proposed above requires some refinements. A preemption level must be specified in each LSP request and the link states must differentiate their reserved bandwidth according to these levels. We have then

$$\text{LSP request} = (p, BW)$$

$$\text{Link state} = (C, RBW)$$

with p the preemption level, C the capacity of the link, BW a required bandwidth and RBW a vector summarizing the amount of reserved bandwidth per preemption level. Note that in this document, a higher p corresponds to a lower precedence level : $p = 0$ is the highest priority for an LSP that cannot be preempted.

On this basis, preemption can be introduced in the scheme simply by extending the constraint predicate : we now have on the link from nodes i to j

$$\sum_{k \leq p} RBW[k] + BW \leq C_{(i,j)}$$

the condition to be respected when adding a new LSP of priority p and of required bandwidth BW.

In practice, this means that LSPs with weaker preemption levels are not taken into account while evaluating the predicate, they might as well be inexistent. As a consequence, if the link is chosen to carry the new LSP, the capacity may be exceeded and some low-precedence LSPs might need to be rerouted. In this regard, there are two major issues :

- How to select the LSPs to preempt/reroute on a particular link?
- Since the high priority LSPs ignore completely the low priority ones, how can we avoid an explosion of reroutings?

The next two sections focus on both problems.

3.1 Selection of the preempted LSPs

Once a path has been computed for a new LSP, it will be established thanks to explicit LSP routing mechanisms like those of RSVP-TE [1] or CR-LDP [2]. As the necessary bandwidth reservations may lead to exceed the link capacity threshold, some LSPs of weaker preemption level must be preempted (and rerouted). The problem is to choose which ones are the most suitable (politically as well as practically) to be cast out. Note that this is a local decision and is independent of the overall routing procedure (though the reverse is not necessarily true, as we shall see in next section). This particular issue is tackled elegantly in [7]. The selected LSPs are chosen on the basis of three criteria : the number of LSPs to preempt, the precedence of the preempted LSPs, the excess of bandwidth gained. There is also the possibility for some LSPs to accept a rate reduction in order to avoid rerouting. They propose a heuristic to solve this optimization problem. We have done similar work though without the rate reduction opportunity. Our selection method is somewhat different and computationally more efficient, though at the cost of flexibility. Indeed, we have in mind a very dynamic scheme with a very short response time.

The same three criteria are used to make a choice but in a more direct manner :

- The most important aspect is the precedence level. An LSP of level p must not be preempted as long as any LSP of weaker precedence $p + 1$ remains.
- Secondly, a set of LSPs is a better choice than another set if it has a lower cardinality and the same maximal precedence level among its members.
- Thirdly, a set of LSPs is a better choice than another set with the same cardinality and the same maximal precedence level if the preempted bandwidth is lower.

Finding the best set of LSPs to preempt in a link on the basis of those three criteria can be shown to be an *NP-complete* problem. Looking for a dynamic, computationally efficient algorithm, we have to settle for an approximation which will sometimes put a bias on the third criteria (the least important one). Under these conditions, this approach allows for a very efficient implementation. Remember that RBW is a vector of the link state summarizing the amount of reserved bandwidth per preemption level. We suppose that we have for each link (at the input interface node) some vectors L^p gathering all LSPs of preemption level p for all possible levels. These must be kept ordered by decreasing reserved bandwidth ($L^p[0]$ is the greediest LSP of level p). Let $precedence$ be the preemption level of the new LSP coming through, p_{max} the lowest authorized precedence level and $bskt$ the set of to-be-preempted LSPs. nd is the bandwidth to gain on the link from node i to node j in order to respect its capacity, with

$$nd = \max(0, \sum_{k \leq p_{max}} RBW[k] + BW - C_{(i,j)})$$

We can proceed using the following algorithm :

```
$p := p_{max};\ bskt = emptyset();$
**while** $nd > 0$ **do**
        **if** $p \leq precedence$
          **then** $ERROR;$
        **fi**
        **if** $nd \geq RBW[p]$
          **then** **foreach** $l \in L^p$ **do**
                  $bskt := bskt + l;$
              **od**
              $nd := nd - RBW[p];$
              $p := p - 1;$
          **else**
              **while** $nd > 0$ **do**
                  **if** $L^p[0] \leq nd$
                    **then**
                        $i := 0;$
                    **else**
                        $i := bin\_place(L^p, nd) - 1;$
                  **fi**
                  $bskt := bskt + L^p[i];$
                  $nd := nd - L^p[i]\text{->}BW;$
                  $shift(L^p[i]);$
              **od**
        **fi**
**od**
```

Note that the $bin\_place(A, b)$ procedure computes the index b would have if it were inserted in the vector A (ordered by decreasing values), using a binary search. The $shift(A[b])$ procedure removes the element of index b in A and moves down all elements of superior index, which can be done very efficiently.

At some point in the algorithm, there is an error case. This occurs when there is not enough preemptable bandwidth available to balance the need of the new LSP. In this case, the reservation must be rejected and the establishment fails. How can this happen? The routing procedure is done at a distant ingress node whose knowledge is dependent on periodic link state updates. Because of race conditions or too long a span of time between updates, the routing process can choose a certain link thinking erroneously that it has enough bandwidth or preemptable bandwidth to sustain the new LSP.

This algorithm has a complexity $O(M)$, M being the number of LSPs carried through the link. The algorithm proposed in [7] requires to sort a vector of size M for each new run. It has then a complexity $O(M * \log_2 M)$ (the possible size of the vector prevents a quicker sorting procedure using *address computation*). It is important to note, however, that the complexity of our approximation can be decomposed into $O(L + K + \log_2 N)$ where N is the number of LSPs present on the link with the highest priority encountered among those preempted, K the number of preempted LSPs of that level and L the number of different priority levels in which preemption occurred. Practically, we will see (section 4) that K remains very low (close to 1) and L is bounded by the (low) number of preemption levels. Under these conditions, the evolution of the computational cost of our scheme falls down to $O(\log_2 N)$ or $O(\log_2 M)$, whereas the approach in [7] does not have any best case.

As we mentioned previously, this algorithm yields an approximation on the optimal selection. Let us show this with a small example. A new LSP requires a bandwidth gain of 135Mb among four LSPs consuming 60, 50, 45 and 30Mb. The scheme presented above will select the 60, 50 and 30Mb LSPs for rerouting, whereas the best choice would be to reroute the 60, 45 and 30Mb LSPs. The accuracy and relevance of this approximation will be put under test in section 4.

It is important to note that if the LSP population on a link is big enough, the number of LSPs preempted and in need of rerouting is very small (generally one, sometimes two). Indeed the chances to find a LSP of the same size but with the lowest precedence (therefore being optimal from any point of view we could take) rises quickly. In this case, the more costly though flexible algorithm described in [7] will give similar results to the quicker, rigid scheme presented here. For this reason, we think our approach is more suited to a dynamic scheme routing lots of small volatile LSPs in a short span of time, whereas the method described in [7] is particularly interesting with bigger static LSPs.

3.2 Safeguard against rerouting explosion

The danger with such a scheme comes from the aggressivity in the routing process of a new potential LSP towards those already active but with a lower precedence. Up to now, the latter are completely ignored by the routing process. A fully loaded link can indeed be used without concern for the reroutings induced even though some other (less optimal) path could be used without requiring any preemption procedure. This can be troublesome if a routing policy like a shortest path is used. In that case, all LSPs with the same source and destination will try to use the one optimal shortest path between those two nodes. When the bottleneck of this route is full, weaker LSPs are cast out each time a new LSP is routed, though a small detour could avoid the need for preemption. Moreover, a rerouted LSP will probably take this small detour, making the overall network situation no better or worse except that the computational cost and the disturbance to existing flows are higher. As a consequence, there is a need for a feedback from the preemption mechanism on the routing procedure : a path requiring some reroutings must be penalized in comparison to others that do not involve preemption.

The penalization mechanism should ideally follow these few guidelines :

- The penalization should be less (or zero) if the link is really interesting (brings a low variation on the score function) but should grow increasingly harder as it makes the overall situation worse.
- Its effect must be tunable in order to adjust the balance between stability and perpetual reorganization (the latter being, in our view, a sane feature but only up to a certain limit).
- It must scale well with the score function. The penalization must not be so great that it completely dominates the score function. The routing procedure would then correspond to a "minimal reroutings path" whatever the case. On the other hand, it must not be so weak that it cannot affect the choice of the optimal path.
- It must affect the routing procedure in similar proportions whatever the score function used.

The principal difficulty comes from the fact that the score variations brought by using a certain link cannot be normalized. Indeed we want to stay general and independent from the score function. As a consequence, a simple additive penalization cannot be used. Moreover some link may bring a negative score variation when an LSP is added, improving the overall situation (in the case of load balancing for instance, by smoothing out the link load disparity). A pure multiplicative penalization might then favor using links with a negative cost (score variation) since their penalization would be negative, and of course the absolute value is not an acceptable solution. As a consequence, we make a distinction between positive and negative score variations, so that the penalization is discontinuous in regard of the cost of the link.

If $\Delta score$ is the score function variation brought by using the link to carry the new LSP, $\Delta score'$ is the corresponding penalized link cost, BW is the bandwidth requested by the new LSP, $need$ is the minimal amount of bandwidth that must be preempted (remember that the routing procedure has no knowledge of the LSPs running through the link), β a tuning parameter, we choose to penalize the potential selection of the link in this way :

$$\Delta score' = \begin{cases} (1 + \beta * \frac{need}{BW}) * \Delta score \\ \quad if \quad \Delta score > 0 \\ \Delta score \quad otherwise \end{cases}$$

This induces that a LSP that does not increase the score function (which should not be frequent) will not be penalized. Another LSP is crippled increasingly with the ratio of the bandwidth to gain : if all the LSP resources need to be preempted, the cost of the link is multiplied by the user defined β parameter. This simple mechanism, though very simple, has appeared relevant and sufficient in our simulations (section 4).

This issue of minimizing the number of reroutings is also explored in [10]. The approach followed is to choose the path with the lowest maximal priority among the LSPs preempted on all its links. In case of equality of the impact brought by two links, the widest path is used (the highest free bandwidth). One drawback is that the preemption minimization feature becomes the main policy at work in The traffic Engineering mechanism. In that sense, we think our approach is more flexible since it is a secondary policy that can be combined to any other Traffic Engineering main policy (such as load-balancing) with an impact that is user-tuned.

Note that in [2] a kind of safeguard against rerouting explosion is achieved by differentiating between holding priority (preemption level when the LSP is actual) and a lower setup priority (preemption level for a would be LSP). The difference between these two priorities allows to some extent to reduce the instability in the network. However, there is no feedback on the routing procedure and this scheme still do not take into account the amount of LSPs with weaker preemption level rerouted. In fact, this two-priority scheme is not in competition with our approach : both work at a different conception level and could combine very well.

4 Simulations and results

In this section, we will make a survey of the relevance and the impact on reliability of the preemption mechanism we have described. The results are all obtained with the same generic approach :

- A topology (20 nodes) is randomly generated. 10 ingress nodes are selected and a source-destination probability distribution along with a requested bandwidth probability density are created. The latter is chosen so that the average number of LSPs on a link is close to 10, a kind of worst-case configuration for the LSP selection heuristic. The capacities on each link are then "engineered" much like a human designer would do it (a more complete description can be found in [8]).
- Some LSP requests respecting these probabilities are generated with an equiprobable precedence level taken between 0 (most important) and 2 (most disposable).

Those LSPs are added up to a network load of 80%, a difficult congested situation favoring lots of reroutings. This network load is then kept stationary by removing and establishing some LSPs as required. The idea is to simulate the evolution of the network state in the long run in order to average a few interesting values.

- In the three following sections 5000 LSP requests are generated. The number of reroutings, of failed reroutings, of establishment failures are used as indicators of performance and must be put in relationship with these 5000 steps.

This test is organized in three parts. We will first study the validity of our selection algorithm for the preempted LSPs (section 3.1). We will then try to show that the feedback given to the routing process by the preemption penalization mechanism has the required characteristics (section 3.2). Finally, we will track the influence of preemption on reliability, as well as the "preemption - score function" relationship.

4.1 Validation of the heuristic used for the selection of the preempted LSPs

The approximative selection of our heuristic is compared with that obtained with a mixed-integer solver used to find the true optimal solution. Let us stress a few points of importance. Both methods of selection will always choose the same number of preempted LSPs. Moreover, with both approaches, all LSPs of a given priority will be preempted before some LSPs with the directly superior precedence level are chosen for rerouting. The only difference comes from the choice of the LSPs to preempt in the highest preempted level, in order to minimize the excess of bandwidth gained. A mixed-integer solver finds the right combination minimizing this excess, whereas our heuristic only finds an approximation with the same number of LSPs within this level. However, when there is only one LSP to choose in the highest preempted precedence level, both solutions are rigorously identical. Finally, the practical difference between both approaches should be close to nothing and indeed we want to show that our heuristic is very accurate.

We have run a simulation using our algorithm and compared each approximative selection of the LSPs to preempt with the one a mixed-integer solver would have made (though the LSPs rerouted in practice where those selected by the heuristic). The mean bandwidth preempted with the approximation exceeded that of the optimal approach by only 3.76% which is more than acceptable. Of course, this could have been be expected, since the average number of LSPs selected for preemption is generally very close to 1: 1.26 for the heuristic and 1.34 for the optimal method in two independent tests which showed no practical difference in terms of reliability (failed establishments or reroutings) or stability (number of reroutings). We already know that the heuristic is exact when it chooses only one rerouting. One preempted LSP being an optimum hard to tackle, we can wonder if the use of a more flexible mechanism is really worth the cost. Indeed, with such a low number of LSPs selected (though the number of LSPs on a link has been chosen quite small to make this test a worst case), the heuristic has a complexity close to $O(log_2(N_{LSPs}))$.

4.2 Relevance of the preemption penalization feedback on the routing procedure

Figures 1 and 2 show the evolution of the number of reroutings along with the repartition per preemption levels. The shortest path is used in figure 1 while a load balancing score function ($\alpha = 0$) is involved in figure 2. Obviously, the penalization scales quite well with both score functions: the number of reroutings (per preemption level or globally) decreases smoothly with an increase of β. Of course, this number cannot fall below a certain level. For β big enough, the only reroutings are those that are absolutely necessary (because there is no free path available, indeed 80% of network load is a difficult configuration). Increasing β in this situation will not have any significant effect. Another interesting point is that the penalization mechanism, though formulated in a generic manner, produces the same proportional evolution of the number of reroutings with β under both score functions. The tuning through the β parameter can then be performed somewhat independently of the routing criteria in use. Finally, the requirements formulated in section 3.2 are all satisfied and the preemption mechanism has the possibility to perform a significant feedback on the routing procedure.

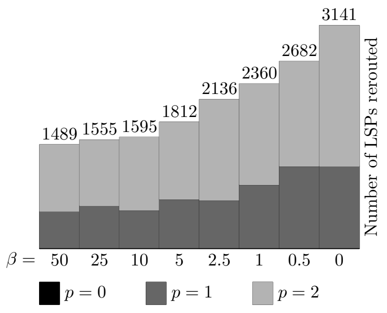

Figure 1: Rerouting penalization under a shortest path score function

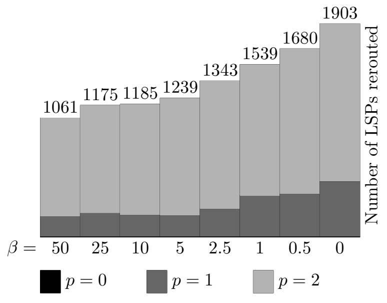

Figure 2: Rerouting penalization under a load balancing score function

4.3 The "preemption - score function" relationship and its influence on reliability

Figure 3 shows the number of failures of all kind (routing + rerouting) without preemption mechanism ($None$), with penalized rerouting ($\beta = 5$) and finally with unpenalized rerouting ($\beta = 0$). Whatever the score function in use, the most obvious point is that without preemption the failures are equally distributed among the three levels of precedence. When preemption is added to the scheme though, the level 0 LSPs almost never fail, constituting a fully reliable connection, the level 1 LSPs have an acceptable chance of failure, making an intermediate category, whereas the level 2 LSPs endure the largest part of interruption and lack of available route, which gives them a status of cheap and unguaranted, disposable service. This possibility to make a distinction between classes of service is, of course, the main reason for using preemption mechanisms.

There is still another lesson to learn when we compare the results obtained with the two different objective functions. In the case of a shortest path, the overall reliability is made worse by the introduction of preemption mechanisms. Indeed, there are lots of difficult bottlenecks since no procedure striving to avoid congestion is at work in the network, and any rerouting is very risky. Moreover, there is generally one single shortest path between two nodes, so that all level 0 LSPs are gathered on this path, casting out level 2 and level 1 LSPs. The latter will use the second shortest path, forbidding access to the level 2 LSPs. As a result, there are lots of pure level 0 links or pure level 1 links which is not suitable. Because of this for instance, the level 1 LSPs don't even gain any advantage in reliability from the use of preemption. With load balancing, on the contrary, the overall reliability is improved by the possibility of (penalized) preemption. Indeed, the smooth repartition of the load through the network makes rerouting an acceptable risk. Besides, an appropriate repartition of all precedence levels is maintained on each link. This allows level 1 LSPs to be preempted in very few occasions.

More than being interesting from a pure routing strategy point of view, the load balancing score function can bring a useful increment to preemption mechanisms with a gain in reliability and relevance.

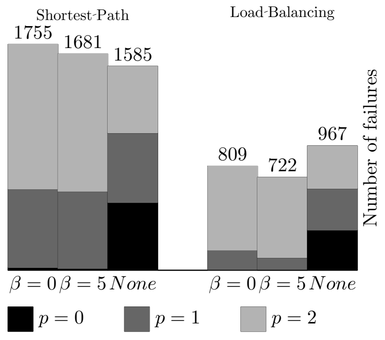

Figure 3: Impact of preemption on reliability / influence of the score function

5 Conclusion

The integrated routing-preemption solution discussed in this paper has two main advantages : its practical implementation is possible and it provides what is expected from a relevant preemption mechanism with a remarkably low computational cost.

Its implementation is possible since RSVP-TE [1] and CR-LDP [2] are now a reality and provide the possibility for explicit routing, which is all that is required for the scheme we proposed in [8]. We are currently considering an implementation of this traffic engineering entity in a last generation platform from an industrial partner. The preemption mechanism described in this document is a simple internal upgrade of this routing procedure, with very few modifications required.

It provides what is expected from a preemption mechanism. The tests of section 4 showed that it allows us to make a distinction between three politically interesting classes of LSPs :

- High priority LSPs, sure to get through and never interrupted.
- Medium priority LSPs, with a high probability of getting through and seldom interrupted.
- Low priority LSPs, most subject to failures and interruptions.

This was only an example, but it shows interesting practical possibilities. This is especially interesting since this result is obtained with a very low computational cost thanks to an heuristic which has proved accurate. It is important to note, however, that this efficiency is obtained at the cost of flexibility. In the difficult situation we chose for the test, the number of reroutings remained acceptable. Besides, the latter is tunable through the feedback from preemption on the routing process. It seems then possible to avoid an excessive use of management resources coming from an explosion of LSP reroutings. This feedback also allows a provider to define the reorganization tempo in his network, in accordance with his own policies towards his clients.

A last advantage is that the overall reliability can be improved through the use of a smart routing process able to avoid congestion and to minimize the number of failed reroutings (*e.g.* using a load balancing objective function as in this document, but other alternatives exist). The relevance of the preemption mechanism itself is improved through an appropriate repartition of all priority levels on each link.

The main shortcoming of our preemption scheme is the lack of opportunity to reduce the rate of some willing LSPs instead of forcing reroutings. Of course, an adaptative rate policy like that proposed in [7] could be introduced in the scheme quite simply, provided that the computational efficiency is not put at stake.

Acknowledgement

The authors would like to express their gratitude to Stefaan De Cnodder from Alcatel and Olivier Bonaventure from the Catholic University of Louvain-la-Neuve, for providing interesting comments on an earlier version of this paper.

This work has been carried out in the IST-1999-20675 ATRIUM project funded by the European commission.

References

[1] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow. RSVP-TE: Extensions to RSVP for LSP tunnels:. RFC 3209, Dec 2001.

[2] B. Jamoussi, Editor, L. Andersson, R. Callon, R. Dantu, L. Wu, P. Doolan, T. Worster, N. Feldman, A. Fredette, M. Girish, E. Gray, J. Heinanen, T. Kilty, and A. Malis. Constraint-based LSP setup using LDP. RFC 3212, Jan 2002.

[3] Murali S. Kodialam and T. V. Lakshman. Minimum interference routing with applications to MPLS traffic engineering. In *INFOCOM (2)*, pages 884–893, 2000.

[4] P. Aukia, M. Kodialam, P. Koppol, T. Lakshman, H. Sarin, and B. Suter. RATES: A server for MPLS traffic engineering. RATES: A server for MPLS traffic engineering, IEEE Network Magazine, pp. 34–41, March/April 2000.

[5] S. Salsano, F. Ricciato, M. Listanti, and A Belmonte. Offline configuration of a MPLS over WDM network under time-varying offered traffic. INFOCOM, June 2002.

[6] Mohammad Peyravian and Ajay D. Kshemkalyani. Connection preemption: Issues, algorithms, and a simulation study. IEEE Infocom, April 1997.

[7] J.C. de Oliveira, C. Scoglio, I.F. Akyildiz, and G. Uhl. A new preemption policy for diffserv-aware traffic engineering to minimize rerouting. IEEE Infocom, June 2002.

[8] F. Blanchy, L. Mélon, and G. Leduc. An efficient decentralized on-line traffic engineering algorithm for MPLS networks. Available at http://run.montefiore.ulg.ac.be/˜blanchy, July 2002.

[9] R. Bellman and R. Kalaba. Shortest paths through networks. Dynamic Programming and Modern Control Theory, pp. 50-54, Academic Press, 1965.

[10] B. Szviatovszki, A. Szentesi, and A. Juttner. Minimizing re-routing in MPLS networks with preemption-aware constraint-based routing. Computer Communications 25 (1076-1083), 2002.

[11] E. Rosen et al. Multiprotocol label switching architecture. RFC 3031, Jan 2001.

[12] F. Le Faucheur and W. Lai. Requirements for support of diff-serv-aware MPLS traffic engineering. IETF Draft draft-ietf-tewg-diff-te-reqts-06.txt, September 2002.