UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

LAZY REROUTING FOR MPLS TRAFFIC ENGINEERING

Elio Salvadori, Roberto Battiti and Filippo Ardito

March 2003

Technical Report # DIT-03-011

Lazy Rerouting for MPLS Traffic Engineering

Elio Salvadori, Roberto Battiti, Filippo Ardito
Università di Trento, Dipartimento di Informatica e Telecomunicazioni
via Sommarive 14, 38050 Povo (TN), Italy
Email: {salvador,battiti,ardito}@dit.unitn.it

***Abstract*— We propose novel Traffic Engineering (TE) schemes for congestion control in MPLS networks based on a reactive mechanism. While most existing TE schemes to prevent network congestion rely on constraint-based routing (CBR), the proposed algorithms use a local search technique where the basic move is the modification of the route for a single Label Switched Path (LSP). Because modifications cause a temporary disruption in the network, a "laziness" criterion implies that moves are executed only when absolutely necessary or when the situation is very close to requiring it. Two versions of the algorithm are proposed: in the first one, called FID, an already established LSP is rerouted when a certain level of network congestion is detected, while in the second, called Lazy FID (or LFID), it is rerouted when a new LSP request cannot be satisfied. Experiments under a dynamic traffic scenario show a reduced rejection probability especially with long-lived and bandwidth consuming connection requests, thus proving a better network resource utilization compared to existing CBR schemes in MPLS networks, while guaranteeing a reduced computational complexity.**

I. Introduction

Traffic Engineering (TE) is an interesting application of MPLS in IP-based networks [1]. TE main objective is to optimize the performance of a network through an efficient utilization of network resources. The optimization may include the careful creation of new Label Switched Paths (LSP) through an appropriate path selection mechanism, the re-routing of existing LSPs to decrease network congestion and the splitting of traffic between many parallel LSPs.

According to IETF RFC 3272, TE schemes for congestion control can be classified according to the their *response time scale* and their congestion management policies (*reactive* or *preventive*) [2]. Most of the proposed schemes are *preventive*, they allocate paths in the network in order to prevent congestion. The two best known mechanisms in the literature in MPLS networks are Constraint-Based Routing (CBR) and traffic splitting. The first has its roots in the well-known Quality-of-Service routing problems in IP networks and refers to the calculation of LSP paths subject to various constraints (e.g. available bandwidth, maximum delay, administrative policies). The second mechanism, traffic splitting, balances the network load through optimal partitioning of traffic to parallel LSPs between pairs of ingress and egress nodes.

The preventive behavior leads to a common drawback: when LSPs are set-up and torn-down dynamically, these schemes can lead to inefficiently routed paths and to future blocking conditions over specific routes. One of the better performing CBR schemes, called MIRA (Minimum Interference Routing Algorithm), is based on an heuristic dynamic online path selection algorithm [3]. The key idea, but also the intrinsic limitation of the algorithm, is to exploit the *a priori* knowledge of ingress-egress pairs to avoid routing over links that could "interfere" with potential future paths set-up. These "critical" links are identified by MIRA as links that, if heavily loaded, would make it impossible to satisfy future demands between some ingress-egress pairs. The drawbacks are: (i) the identification of the "critical" links leads to a severe computation complexity caused by the maximum flow calculation performed each time a new LSP has to be established (ii) the algorithm cannot estimate bottlenecks on links that are "critical" for clusters of nodes [4] (iii) MIRA can lead to an unbalanced network utilization because it does not take into account the current traffic load in routing decisions [5][6].

Only a few *reactive* congestion control schemes have been proposed. Holness et al. [7] propose a mechanism called Fast Acting Traffic Engineering (FATE): the ingress LER (Label Edge Router) and the core LSR (Label Switched Router) react on information received from the network regarding flows experiencing significant packet losses by dynamically routing traffic away from a congested LSR to the downstream or upstream underutilized LSRs. The authors describe the procedure for congestion detection and its impact on the signalling mechanisms, but do not include any simulation on network performance. A method that considers elastic traffic and exhaustive search is proposed by Casetti et al. [8]. Jüttner et al. [9] propose an algorithm for the optimal routing of new LSPs based on the re-routing of an already established LSP when there is no other way to route the new one: at higher network utilization levels, on-demand CBR-based LSP setup can experience failures. In order to fit the new LSP demands, instead of a global reoptimization of all LSP paths, one quickly reoptimizes a single LSP. The algorithm is based on an Integer Linear Programming (ILP) formulation of the rerouting problem and an heuristic method for practical cases is proposed because of the excessive computation required by ILP. The simulations consider only static paths that stay in the network forever once established. Unfortunately the authors do not specify the traffic model used to run the algorithm, and this does not allow us to perform comparisons with our proposal.

In this paper we present novel schemes to reduce the congestion in an MPLS network by using load balancing mechanisms based on different *local search* heuristics. The key idea is to efficiently re-route LSPs from the most congested links in the network, in order to balance the overall links load and to allow a better use of the network resources. While CBR

is based on a preventive mechanism to avoid congestion, the proposed scheme acts only when a congested network state is detected, thus it can be considered a *reactive* scheme. Network congestion can be detected in two main ways: either when the load on some network links is dangerously close to the link capacity, or when a new LSP demand request cannot be satisfied. In this work we compare both alternatives with well-known CBR mechanisms.

The paper is organized as follows. Section II explains the context and the motivations for our proposal, while the proposed algorithms are explained in Section III. The results are analyzed in Section IV.

II. Problem definition and system model

The considered network consists of n routers. In MPLS terminology connections are set-up between an ingress-egress pair of routers. Each connection request arrives at an ingress router (or at a Network Management System in the case of a centralized route computation) which determines the explicit route for the LSP according to the current topology and to the available capacities at the IP layer. It is assumed that every router runs a link state routing protocol with extensions for link residual bandwidth advertisements.

A connection request i is defined by a triplet (i_i, e_i, b_i), where i_i and e_i specify the ingress and egress routers and b_i indicates the amount of bandwidth required. In the rest of the paper we will consider only the routing of bandwidth guaranteed connections. As in [3], we suppose that Service Level Agreements (SLA) are converted into bandwidth requirements for the LSP. We assume also an on-line context with connection requests arriving one at a time.

The corresponding LSPs will be routed through the network and, at each instant, one determines the *virtual load* of a link by summing the bandwidth b_i of the connections passing through the link. The difference between the link capacity and the virtual load gives the *residual bandwidth*. The minimum residual bandwidth over all links of a network is called the *available capacity* of the network. This value identifies the *most congested links*.

Given an MPLS network with connection requests arriving dynamically, the objective of our on-line algorithm is to balance the allocation of the already established LSPs in the network to reduce the rejection probability for future traffic demands.

III. A Load Balancing Algorithm for Traffic Engineering

We consider algorithms that are based on a sequence of small steps (i.e., on local search from a given configuration) because global changes of the routing scheme can be disruptive to the network. A similar approach has been proposed in papers about Logical Topology Design and Routing in WDM Optical Networks [10], [11]. For each tentative move, the most congested links are identified and one of its crossing LSPs is rerouted along an alternate path. The scheme is similar to the congestion control mechanism introduced in [12], that considered connections routed through a destination-based routing. In [13], a previous version of the algorithm called DYLBA (Dynamic Load Balancing Algorithm) is proposed. In DYLBA, the search for an alternate route is performed for all of the LSPs crossing the most congested links. Despite its encouraging results, this algorithm is computationally demanding due to the extensive search performed to find the best LSP to reroute.

Compared to DYLBA, the new scheme is based on a faster local search technique: the local search stops as soon as first improving alternate route for one of the LSPs is found, thus dramatically reducing the computational time. Moreover, a Widest-Shortest-Path (WSP) algorithm [14] is used instead of plain Min-Hop routing in order to improve the routes calculation for the LSPs in the network, by taking advantage of the available information regarding the link residual bandwidth.

Two different versions are considered, according to the triggering mechanism used to start the load balancing routine:

1) First-Improve Dynamic load balancing, called FID(x). The parameter x indicates the threshold for the link residual bandwidth measured as a fraction of the link capacity, which determines when a link is considered congested. Each new request is routed with WSP, a modified Shortest-Path algorithm which runs on a graph where link weights are defined as $w_l{=}1/c_l$, where c_l is the residual available link capacity, ∞ if c_l is zero. After routing, if less than x residual bandwidth is left on some link, the dynamic load balancing algorithm is executed. As soon as the alternate route for one LSP has more residual bandwidth than the original route, the search stops and the LSP is rerouted. If the search cannot find any improving alternate LSPs in the network for all congested links, rerouting is not performed.
2) Lazy First-Improve Dynamic load balancing, called LFID. In this version, the dynamic load balancing is activated when a new LSP request arrives that cannot be satisfied. Now only links having the smallest residual bandwidth are considered congested. The algorithm goes through LSPs crossing them until any improving alternate route computed with WSP is found. If the search is successful, the LSP is rerouted over the path found and another attempt is made to establish the new LSP request. If it fails, the new request is rejected. In [9] the triggering mechanism is similar but the authors used ILP for reallocating the LSPs.

Figure 1 shows the pseudo-code of the load balancing algorithm. Let us first define the notation. The most congested links are collected in the $congestedLinkSet$ which is computed through the function $calculateNetworkLoad$. This function is different for the two implementations of the algorithm. In FID(x), the most congested links are all the links whose residual bandwidth is lower than the fraction x of their capacity. In LFID, the most congested links are all the links with the minimum (relative) residual bandwidth at the moment. The advantage in this case is that we do not need to

FIRST IMPROVE DYNAMIC LOAD BALANCING ALGORITHM

```
1. <congestedLinkSet> ← calculateNetworkLoad
2. betterLSPFound ← false
3. while (congestedLinkSet ≠ ∅) and (not betterLSPFound)
4.       (cFrom, cTo) ← pickElement (congestedLinkSet)
5.       LSPSet ← all the LSPs crossing link (cFrom, cTo)
6.       while (LSPSet ≠ ∅) and (not betterLSPFound)
7.            LSP_i ← pickElement (LSPSet)
8.            currResBdw ← residual bandwidth on the LSP_i's route
9.            removePartialLoad (LSP_i)
10.           find an alternate path A_LSP for LSP_i
11.           if (A_LSP is found)
12.                altResBdw ← residual bandwidth on the A_LSP's route
13.                if altResBdw > currResBdw
14.                     betterLSPFound ← true
15.                     oldLSP ← LSP_i
16.           restorePartialLoad (LSP_i)
17. if (betterLSPFound)
18.      reroute traffic from oldLSP to A_LSP
```

Fig. 1. The pseudo-code for the First-Improve DYLBA

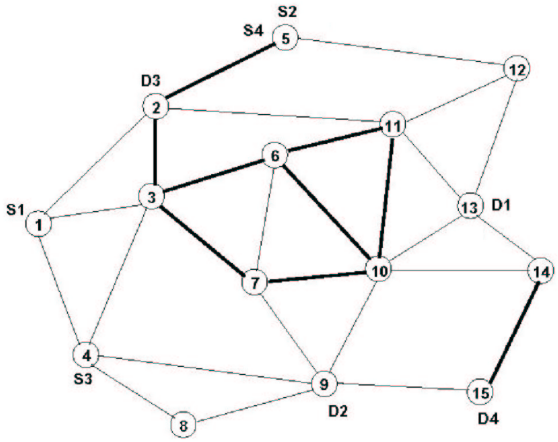

Fig. 2. The network topology used in the simulations.

TABLE I

BLOCKING PROBABILITY (AND IMPROVEMENTS W.R.T. MHA)

Algorithm	Bw $\in \mathcal{U}[1,3]$, $\lambda/\mu = 150$		Bw $\in \mathcal{U}[1,6]$, $\lambda/\mu = 75$	
MHA	$12.5 \cdot 10^{-2}$	-	$8.0 \cdot 10^{-2}$	-
MIRA	$9.4 \cdot 10^{-2}$	(–25.1%)	$3.9 \cdot 10^{-2}$	(–50.7%)
FID(0.01)	$8.2 \cdot 10^{-2}$	(–34.6%)	$3.3 \cdot 10^{-2}$	(–58.1%)
LFID	$8.6 \cdot 10^{-2}$	(–31.4%)	$3.3 \cdot 10^{-2}$	(–58.1%)

set a threshold, which is a critical parameter and depends on the traffic load in the network.

For each iteration cycle, we consider each congested link in $congestedLinkSet$, identified by its endpoints ($cFrom$, cTo). For each LSP_i crossing the link ($cFrom$, cTo), taken in decreasing bandwidth request order, one tries to reroute it on an alternate route. This move is accepted only if the new path increases the available capacity of the network, calculated as the minimum (absolute) residual bandwidth available on the route of LSP_i, which is $currResBdw$. To perform this operation, one temporarily removes the load of LSP_i from the current link, and finds a new path (A_LSP) using WSP starting from the ingress LER (I-LER) which originated the LSP itself, provided that all the links considered congested ($congestedLinkSet$) are avoided. If an alternate path for LSP_i is found, the minimum (absolute) residual bandwidth available on it is calculated as $altResBdw$ and compared to $currResBdw$. If the available capacity of the network is improved, the move is accepted and in the last part of the algorithm the rerouting is executed (lines 17–18).

Let us consider the worst-case computational complexity. The proposed algorithm is composed of nested cycles. Let n be the number of nodes in the network and m its number of links. The number of iterations for the loop at line 3 is only bounded by the number of links in the network, while for the loop at line 6 is bounded by the number of LSPs that cross the congested link, called k. The computation of the alternate path for the selected LSP_i using Dijkstra' shortest-path algorithm requires $O(nm)$. This can be improved to $O(nlogn + m)$ by using a priority queue with Fibonacci heap in the implementation. Functions $removePartialLoad$, $restorePartialLoad$ and calculations of residual bandwidth on LSP route have complexity at most $O(n)$, since an LSP cannot contain more than n hops. All the other functions require a constant computational time. The value of k depends on the average bandwidth of the LSPs in the network and the link capacities. Therefore the complexity of our algorithm is not more than $O(knm^2)$. As demonstrated in the experimental section, the actual empirical complexity is much lower than this worst-case bound.

IV. SIMULATION RESULTS

The simulations are carried out by using the network topology of [3], see Figure 2. The links are all bidirectional with a capacity of 120 units (thin lines) and 480 units (thick lines). These values are taken to model the capacity ratio of OC-12 and OC-48 links. In order to compare our schemes with MIRA, traffic requests are limited only to the ingress and egress router pairs (S_1,D_1), (S_2,D_2), (S_3,D_3) and (S_4,D_4). However, it is important to highlight that our algorithms allow to relax this strong constraint.

The experiments compare our algorithms with Minimum-Hop Algorithm (MHA) and Minimum Interference Routing Algorithm (MIRA). Connection requests arrive between each ingress-egress pair according to a Poisson process with an average rate λ, and their holding times are exponentially distributed with mean $1/\mu$. Ingress and egress router pairs for each LSP set-up request are chosen randomly. The network is loaded with 20000 requests during one trial.

The first results (Table I) show the blocking probability of MIRA and both implementations of our algorithms FID(x) (with $x = 0.01$) and LFID, comparing them with MHA. The rejection ratios shown are average values calculated over 10 runs. The first experiment considers the same traffic distribution used by Kodialam et al. [3], i.e. bandwidth demands for LSPs uniformly distributed between 1 and 3 units and

$\lambda/\mu = 150$ for each ingress-egress router pair. MIRA and the proposed algorithms perform all better than MHA, but while LFID perform slightly better than MIRA (with a 6% decrease in blocking probability), FID(0.01) performs the best (10% ca. over MIRA). The second experiment considers LSPs with higher capacity on average, i.e. the bandwidth demands are uniformly distributed between 1 and 6 units, but considering half the λ/μ ratio compared to the previous case. MIRA and the proposed algorithms perform all better than MHA, while both FID(0.01) and LFID perform similarly, showing a 7% decrease in blocking probability with respect to MIRA.

These first results show that our algorithms perform slightly better than MIRA, especially if the traffic considered is characterized by bandwidth-consuming LSPs. This can be explained by the implicit mechanism of First-Improve DYLBA, which reroutes an LSP away from the most congested link and considers first the LSPs with higher bandwidth request, thus guaranteeing a faster network congestion reduction.

In order to evaluate in more detail the proposed algorithms, different set of experiments have been performed. The first set considers a uniform distribution of traffic among all the ingress-egress pairs (symmetric traffic). The second set of experiments considers a non-uniform distribution (asymmetric traffic). In particular, it is assumed that ingress-egress pair (S_1,D_1) generates a traffic rate which is four times higher than the other pairs, on average. This set allows us to highlight MIRA limitations regarding the traffic load condition over the MPLS network (see Section I). Two experiment subsets are performed for two different bandwidth distribution per LSP (maximum bandwidth 3 and 6).

Figure 3 presents the results for symmetric traffic. Two different bandwidth distributions per LSP are considered: the first has a maximum bandwidth of 3 units, while the second 6 units, thus simulating the case of bigger connections on average. Rejection values are calculated over 10 runs: the error-bars are not shown in the plots because they are hardly visible, in the order of 4%. Our algorithms perform slighlty better than MIRA in these traffic conditions. The upper plot shows that FID(0.01) performs slightly better than LFID for high values of λ/μ, due to the implicit reaction mechanism: the load balancing algorithm is triggered whenever some link overcomes the congestion threshold, thus keeping a better distribution of the load over the network.

Figure 4 presents the results for asymmetric traffic. Two different bandwidth distributions per LSP are considered as above. The plots show that these traffic conditions lead to a higher blocking probability on average compared to the previous set of experiments. Capone et al. [5] proved that MIRA does not perform well when asymmetric traffic is applied to the ingress-egress pairs. The proposed schemes perform much better than MIRA mainly thanks to the independence of our algorithms from the traffic conditions. As in the first set of experiments, FID(0.01) performs slightly better than LFID.

Table II shows a comparison between the two proposed schemes FID(0.01) and LFID over 20000 LSP requests, by using symmetric traffic with bandwidth demands for LSPs

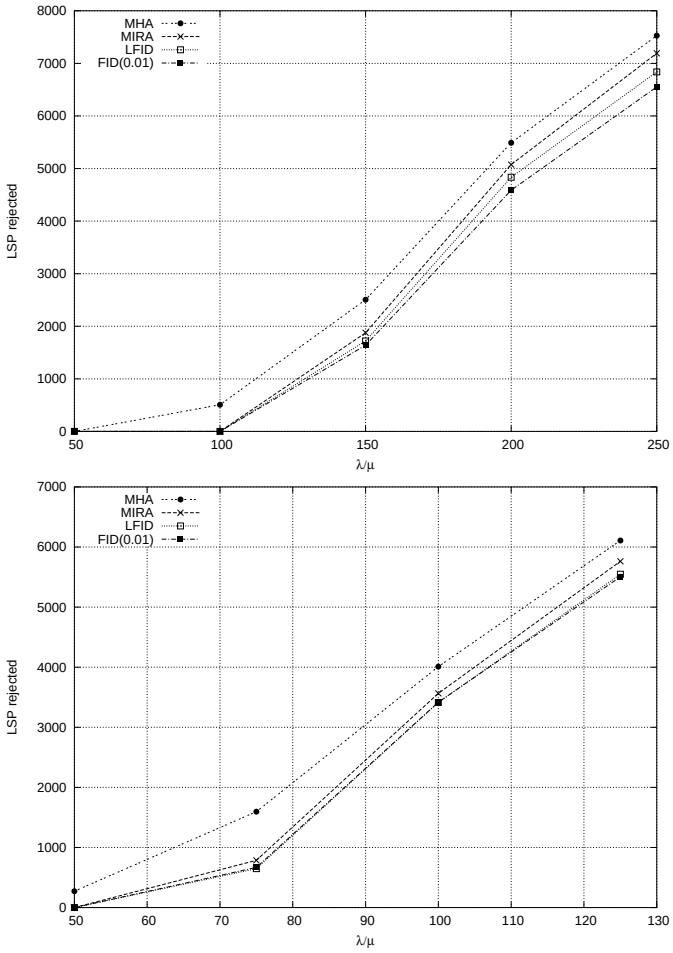

Fig. 3. Symmetric traffic: number of rejected LSPs vs. λ/μ for maximum bandwidth equal to 3 (upper plot) or equal to 6 (lower plot)

TABLE II

PERFORMANCE OF THE PROPOSED REACTIVE SCHEMES

Algorithm	LSPs rerouted	Routes computed	Num. of congLink	Num. of LSPs per congLink
FID(0.01)	8152.7	1107K	3.6 ± 2.7	61.8 ± 6.4
LFID	1368.5	260K	3.3 ± 2.2	62.1 ± 3.8

uniformly distributed between 1 and 3 units and λ/μ = 150. The first column shows the number of LSPs rerouted during the algorithm run, while the second column shows the number of alternate routes considered during the simulation before finding the best path to reroute, a measure of the computational time spent by the algorithm. From these values it can be noticed that, even if it is the best performing scheme in term of rejected requests, the FID(0.01) scheme leads to the highest number of re-routings in the network and requires more computation. In addition to requiring less computation and reroutings, LFID has the additional advantage of being independent from the threshold value. The third column shows the average number of links considered congested by our

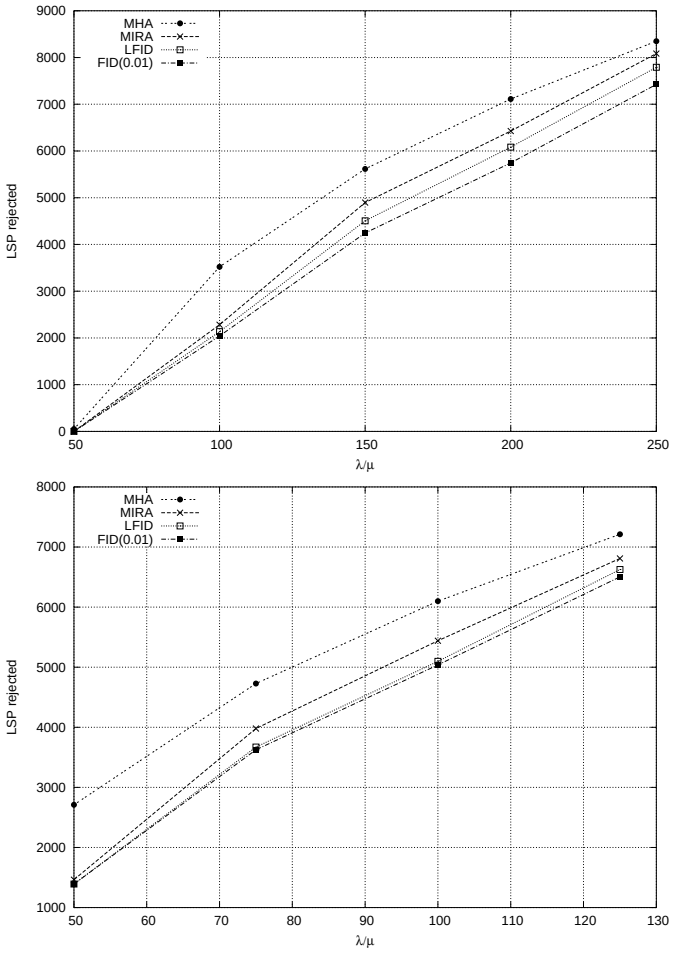

Fig. 4. Asymmetric traffic: number of rejected LSPs vs. λ/μ for maximum bandwidth equal to 3 (upper plot) or equal to 6 (lower plot)

algorithms during their run, while the fourth column shows the number of LSPs crossing each congested link on average. These values prove that the number of links considered congested is always a small percentage of the total number of links (for the specific network we are considering in our experiments, it is between 2% and 12%). This means that the empirical computational complexity is much lower than the worst-case bound calculated in Section III.

The results demonstrate that it is possible to obtain better rejection ratios with the proposed reactive schemes than with MIRA and at a reduced computational cost. In fact, when a MIRA CBR scheme is employed, each time a new LSP needs to be routed over the network, the maximum flow between each ingress-egress pair nodes in the network has to be computed in $O(n^2\sqrt{m})$ before applying the traditional $O(nm)$ shortest-path algorithm [3]. In our reactive scheme instead, each LSP is installed directly by using Dijkstra algorithm. The load balancing algorithm is triggered only when network congestion is detected, and this happens for a small percentage of LSPs which need to be rerouted. In particular for LFID, the proposed algorithm is computed only for 7.5% of the total number of LSPs, see Table II.

V. Conclusions

Two new online reactive schemes to perform Traffic Engineering in MPLS called FID(x) and LFID have been presented. Simulation results show that our algorithms perform better than Minimum Interference Routing Algorithm (MIRA), a well-known scheme in literature to prevent congestion in MPLS networks, by reducing both the LSPs rejection probability and the computational complexity. Both of them show their best results when the traffic inserted into the network is characterized by high-capacity and long-lived connection requests. In fact, in the case of a bandwidth-hungry LSP that lasts for a long time, there is a higher risk of blocking future connection requests due to a suboptimal allocation. A reactive congestion control scheme behaves better than a preventive scheme, because it can dynamically adjust the inefficiently routed paths. Among the two proposed schemes, LFID has the advantage to minimize the number of LSPs to be rerouted in the network to balance the traffic and therefore has the lowest computational time and the lowest disruption of traffic.

References

[1] D. O. Awduche and B. Jabbari. Internet Traffic Engineering using Multi-Protocol Label Switching (MPLS). *Computer Networks*, (40):111–129, September 2002.

[2] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and Principles of Internet Traffic Engineering. RFC 3272, May 2002.

[3] K. Kar, M. Kodialam, and T.V. Lakshman. Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications. *IEEE Journal on Selected Areas in Communications*, 18(12):2566 –2579, December 2000.

[4] B. Wang, X. Su, and C.P. Chen. A New Bandwidth Guaranteed Routing Algorithm for MPLS Traffic Engineering. In *Proceedings of ICC*, volume 2, pages 1001–1005, New York - USA, 2002.

[5] A. Capone, L. Fratta, and F. Martignon. Virtual Flow Deviation: Dynamic Routing of Bandwidth Guaranteed Connections. In *Proceedings of IEEE Workshop on Quality of Service in Multiservice IP Networks*, pages 608–620, Milan, Italy, February 2003.

[6] R. Boutaba, W. Szeto, and Y. Iraqi. DORA: Efficient Routing for MPLS Traffic Engineering. *Journal of Network and Systems Management*, 10(3):309 –325, September 2002.

[7] F. Holness and C. Phillips. Dynamic Congestion Control Mechanism for MPLS Networks. In *SPIE's International Symposium on Voice, Video and Data Communications. Internet, Performance and Control Network Systems*, pages 1001–1005, Boston - MA, November 2000.

[8] C. Casetti, G. Mardente, M. Mellia, M. Munafò, and R. Lo Cigno. Online routing optimization for MPLS-based IP networks. Technical report, Politecnico di Torino, Dipartimento di Elettronica, November 2002.

[9] A. Jüttner, B. Szviatovszki, A. Szentesi, D. Orincsay, and J. Harmatos. On-demand Optimization of Label Switched Paths in MPLS Networks. In *Proceedings of IEEE ICCCN*, October 2000.

[10] A. Narula-Tam and E. Modiano. Load balancing algorithms for WDM-based IP networks. In *Proceedings of INFOCOM 2000*, pages 1010–1019, Tel-Aviv, Israel, March 2000.

[11] J. Skorin-Kapov and J. Labourdette. On minimum congestion routing in rearrangeable multihop lightwave networks. *Journal of Heuristics*, 1:129–145, 1995.

[12] M. Brunato, R. Battiti, and E. Salvadori. Dynamic Load Balancing in WDM Networks. *Optical Networks Magazine*, September 2003. In press.

[13] E. Salvadori and R. Battiti. A Load Balancing Scheme for Congestion Control in MPLS Networks. In *IEEE Symposium on Computers and Communications - ISCC*, 2003. Accepted for publication.

[14] R. Guerin, D. Williams, and A. Orda. QoS routing mechanisms and OSPF extensions. In *Proceedings of Globecom 1997*, volume 3, pages 1903 –1908, Phoenix, AZ, November 1997.