UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

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

A REACTIVE SCHEME FOR TRAFFIC ENGINEERING IN MPLS NETWORKS

Elio Salvadori, Roberto Battiti and Mikalai Sabel

December 2002

Technical Report # DIT-02-0098

A Reactive Scheme for Traffic Engineering in MPLS Networks

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

***Abstract*— In this paper we develop a new Traffic Engineering scheme 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 algorithm uses a local search technique where the basic move is the modification of the route for a single Label Switched Path (LSP). 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 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.**

I. Introduction

One of the most interesting applications of MPLS in IP-based networks is Traffic Engineering (TE) [2]. The main objective of TE 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.

The main approach to MPLS Traffic Engineering is the so-called Constraint Based Routing (CBR). In this paper we present a new scheme to reduce the congestion in an MPLS network by using a load balancing mechanism based on a local search method. 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 network congestion, the proposed scheme acts only when network congestion 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 due to traffic load already installed in the network. In this work we compare both alternatives with well-known CBR mechanisms.

The paper is organized as follows. Section II provides a brief overview of Traffic Engineering schemes in MPLS networks. Then the context and the motivations for our proposal are highlighted in Section III, while the proposed algorithms are explained in Section IV. The results are analyzed in Section V.

II. Traffic Engineering in MPLS networks

One of the crucial problems a Service Provider has to deal with is how to minimize congestion in its network. In packet switching networks, congestion is related to delays and therefore reducing congestion implies better quality of service guarantees and reduced maximum traffic load in the routers. In networks based on circuit switching, reducing congestion implies that spare bandwidth is available on every link to accommodate future connection requests or to maintain the capability to react to faults in restoration schemes.

According to IETF RFC 3272, Traffic Engineering schemes for congestion control can be classified according to the their *response time scale* and their congestion management policies (*reactive* or *preventive*) [1].

Most of the proposed TE 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 type of 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.

One of the most cited CBR schemes, called MIRA (Minimum Interference Routing Algorithm) [9], is based on an heuristic dynamic online path selection algorithm. The key idea 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 main weaknesses of this scheme are the computation complexity caused by the maximum flow calculation to identify the “critical” links and the unbalanced network utilization. As Wang et al. demonstrated in [12] with two counterexample topologies (the “concentrator graph” and the “distributor graph”), MIRA cannot estimate bottlenecks on links that are “critical” for clusters of nodes. Second, it does not take into account the current traffic load in

routing decisions [4]. Let's consider the case where a source-destination pair is connected by two or more routes with the same residual bandwidth. When a new LSP set-up demand arrives, one of these routes will be chosen to satisfy the request. This implies that after this LSP has been set-up, all the links belonging to the other routes become critical according to the definition given above. This means that all the subsequent requests between the same router pair will be routed over the same route while all the other routes remain free thus causing unbalanced resource utilization. Moreover, when the LSPs are set-up and torn-down dynamically, this scheme can lead to inefficiently routed paths and to future blocking conditions over specific routes. This drawback is common to all CBR schemes proposed in the literature, and is due to their implicit preventive behavior.

Only a few *reactive* congestion control schemes have been proposed in the literature. Holness et al. [7] propose a mechanism called Fast Acting Traffic Engineering (FATE) to control the congestion in an MPLS network. 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 taking appropriate remedial action, i.e., by dynamically routing traffic away from a congested LSR to the downstream or upstream underutilized LSRs. The authors describe in detail the procedure for congestion detection and its impact on the signalling mechanisms, but do not include any simulation about the real impact of FATE on the network performance. Jüttner et al. [8] 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. This scheme is based on the idea that 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 it is preferable to proceed with a quick reoptimization of a single LSP. The optimization algorithm is based on an Integer Linear Programming (ILP) formulation of the rerouting problem, and the authors propose an heuristic method to provide efficient solution in practical cases. The simulations performed consider only static paths, i.e. once established, they will stay in the network forever. 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.

III. Problem definition and system model

The considered network consists of n routers. A subset of ingress-egress routers between which connections can be potentially set-up is specified. 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. To perform the explicit route calculation and the load balancing algorithm, each router in the network (or the NMS in the case of a centralized mechanism) needs to know the current network topology and the residual capacities of each link, to identify the most congested ones. It is assumed that every router in the MPLS network runs a link state routing protocol with extensions for link residual bandwidth advertisements.

A connection request i is defined by the vector (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. We assume also that requests for LSPs arrive one at a time without knowledge of future demands. These LSPs will be routed through the network according to some routing scheme. 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 on each link of a path indicates how congested is the path. The minimum residual bandwidth on each link 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.

IV. A Load Balancing Algorithm for Traffic Engineering

The main goal of our scheme is to dynamically balance the utilization of network resources in an MPLS network through a local search algorithm. 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 algorithms in optical networks [10], [11]. The idea is to increase the available capacity of the network by performing local modifications. For each tentative move, the most congested links are located and one of its crossing LSPs is rerouted along an alternate path. The scheme is similar to the congestion control mechanism introduced in [5], [6], that considered connections routed through a destination-based routing. In [3], 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. In fact in this new version the local search stops as soon as the algorithm finds the first improving alternate route for one of the LSPs, thus dramatically reducing the computational time.

Starting from this scheme, two different versions are considered, according to the triggering mechanism used to start the load balancing routine:

1) First-Improve DYLBA, called FID(x). The parameter x indicates the threshold for the link residual bandwidth

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 <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

measured as a percent of the link capacity, which determines when a link is considered congested. After routing each new request by using simple Minimum Hop Algorithm (MHA) routing, if less than x residual bandwidth is left on some link, the dynamic load balancing algorithm is executed. This trigger mechanism is the same used in the cited version of DYLBA [3]. 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 DYLBA, called LFID. In this second version, the dynamic load balancing is used 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 LSP is found. If the search is successful, the found LSP is rerouted and another attempt is made to establish the new LSP request. If it fails, the new request is rejected. The mechanism is similar to the one proposed in [8], but compared to this work we use a local search algorithm to reallocate an LSP inside the network.

Figure 1 gives 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 x percent of their capacity. In LFID, the most congested links are all the links with the minimum residual bandwidth at the moment. The advantage in this case is that we do not have to set a threshold, which is a critical parameter and depends on the traffic load level in the network.

For each iteration cycle, we consider each congested link

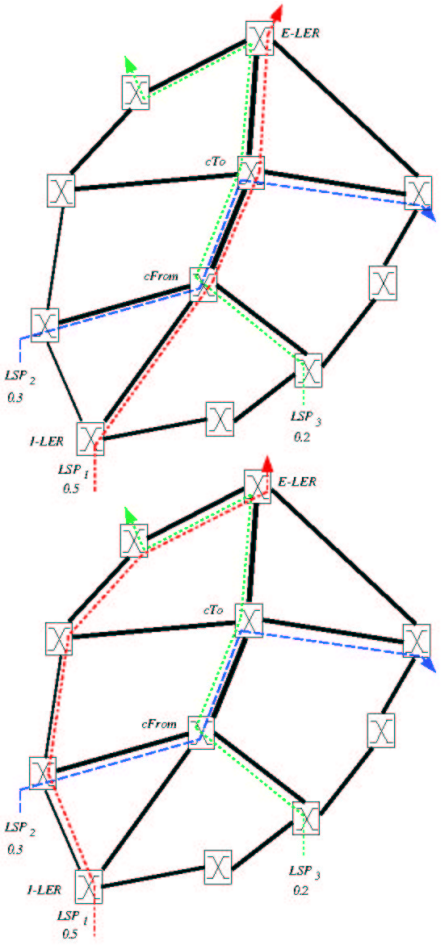

Fig. 2. The rerouting mechanism of the DYLBA algorithm: (upper plot) when link congestion is detected, (lower plot) after the rerouting of the LSP_1

in $congestedLinkSet$, identified by its endpoints ($cFrom$, cTo). For each LSP_i crossing the link ($cFrom$, cTo), taken without any specific 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 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 MHA starting from the ingress LER (I-LER) which originated the LSP itself, provided that the congested link ($cFrom$, cTo) is avoided. If an alternate path for LSP_i is found, the 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 in the network (lines 17–18).

Figure 2 shows an example of the rerouting process of our algorithm. Each link of the depicted network has capacity equal to 1. The bandwidth demand for each LSP is a fraction of the link capacity. In the upper plot, the link ($cFrom$, cTo) is detected as congested, and the algorithm triggers the local

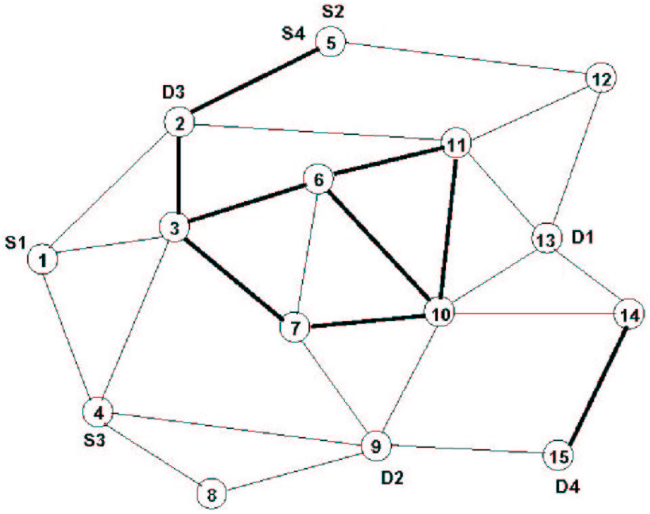

Fig. 3. The network topology used in the simulations.

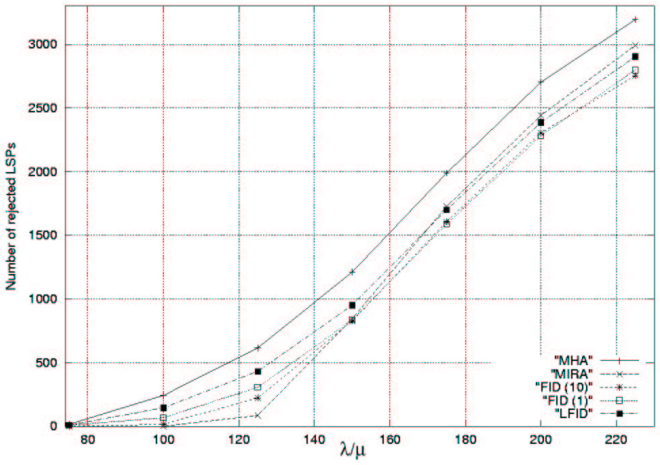

Fig. 4. Number of rejected LSPs vs. λ/μ.

search over the LSPs that cross the link. The LSP whose alternate path guarantees the maximum available capacity in the network (i.e. the minimum network congestion) is LSP_1, so the ingress router I-LER reroute the related traffic over this new path (see Figure 2 (lower plot)).

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 $O(knm^2)$.

V. Simulation results

The simulations are carried out by using the network topology of [9], see Figure 3. 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. Path 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).

All the experiments are carried out by considering the dynamic behavior of our algorithms, Minimum-Hop Algorithm (MHA) and Minimum Interference Routing Algorithm (MIRA). LSPs 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 10000 LSP set-up requests during one trial.

Table I shows a comparison between Minimum-Hop routing Algorithm (MHA), Minimum Interference Routing Algorithm (MIRA), DYLBA algorithm of [3] and both implementations of First-Improve DYLBA FID(x) and LFID. Each element of the table reports the average number of rejected LSP over 5 runs and its standard deviation (within brackets).

The first two sets of experiments are carried out by considering that bandwidth demands for LSPs are uniformly distributed between 1 and 3 units. By using the same traffic distribution considered by Kodialam et al. in their work [9] (λ/μ = 150 for each ingress-egress router pair), both FID(10) and FID(1) perform slightly better than MIRA on average, while LFID performs the worst. For an LSP's average lifetime longer than the previous one (λ/μ = 200), even LFID performs better than MIRA, while it is still worse than FID(x). The second two sets of experiments are carried out by considering LSPs with higher capacity on average, i.e. the bandwidth demands are uniformly distributed between 1 and 12 units. In this case, for both values of λ/μ, both FID(x) and LFID perform better than MIRA.

These first results show that our algorithm performs slightly better than MIRA, especially if the traffic considered is characterized by long-lived LSPs. Furthermore, the higher the LSP's capacity on average, the better our algorithms perform. This can be explained by the implicit mechanism of First-Improve DYLBA, which reroute an LSP away from the most congested link, thus guaranteeing a faster network congestion reduction.

These results prove that our algorithm can overcome some of the limitations of MIRA in term of resource utilization. In fact, as highlighted in Section II, MIRA can lead to inefficiently routed paths when LSPs are set-up or torn-down

Bdw	λ/μ	MHA	MIRA	DYLBA(10)	DYLBA(1)	FID(10)	FID(1)	LFID
1..3	150	1211.0 (40.2)	844.4 (38.7)	859.8 (44.9)	818.4 (47.9)	826.4 (47.9)	836 (52.2)	950.6 (28.4)
1..3	200	2702.6 (65.2)	2441.8 (86.6)	2360.4 (66.6)	2266.4 (75.5)	2304.4 (95.3)	2282.4 (62.7)	2386.0 (60.8)
1..12	150	5634.8 (41.4)	5526.4 (26.0)	5356.4 (49.4)	5395.2 (42.4)	5337.2 (28.4)	5448 (21.4)	5472.4 (14.8)
1..12	200	6310.0 (48.9)	6198.2 (50.5)	6005.4 (51.7)	6040.6 (39.7)	5963.2 (55.1)	6090.6 (50.6)	6108.4 (14.9)

TABLE I
NUMBER OF BLOCKED REQUESTS

dynamically. This effect has a stronger impact on the network performance when the LSP requests have higher holding times. Figure 4 depicts the number of rejected LSP demands for different values of the ratio λ/μ and LSP's bandwidth demands uniformly distributed between 1 and 3 units. This plot shows that MIRA performs better than FID(x) schemes only in the interval $75 \leq \lambda/\mu \leq 150$, while for λ/μ greater than 175, even LFID performs better.

Bdw	λ/μ	FID(10)	FID(1)	LFID
1..3	150	9315 (2000K)	8483 (822K)	1062 (373K)
1..3	200	7865 (4270K)	7512 (2175K)	1785 (985K)
1..12	150	8609 (1500K)	4654 (324K)	3358 (434K)
1..12	200	7839 (2200K)	4525 (527K)	3213 (555K)

TABLE II
NUMBER OF LSP REROUTED OVER 10.000 REQUESTS

Table II shows a comparison between the two proposed schemes FID(x) and LFID in term of number of LSP rerouted over 10000 LSP requests. The number within brackets shows the number of alternate routes each algorithm has considered during the simulation before finding the best path to reroute. This last parameter gives us an estimate 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(x) scheme with an high value of threshold (10%) can lead to the highest number of re-routings in the network, while requiring the greatest number of computations among all. By using a lower value of threshold (1%), FID(x) performs better than the previous one both in term of rerouted LSPs and computational time. LFID has the advantage of being independent from the threshold value, due to a different triggering mechanism (see Section IV for details). This lead to the lowest values of rerouted LSP in the network, and also to a reduced computational time.

VI. CONCLUSIONS

In this paper a new online algorithm to dynamically balance the load in an MPLS network has been presented. Simulation results show that our algorithm can perform better than Minimum Interference Routing Algorithm (MIRA), one of the most cited scheme in the literature for the congestion control in MPLS, by improving the network utilization and reducing the LSPs rejection probability.

Both FID(x) and LFID show their best results when the traffic inserted in the network is characterized by high-capacity consumption and long-lived LSPs. In fact, with bigger LSPs on average, the proposed schemes can reduce the network congestion in a faster way due to their reactive mechanisms which reroute an LSP away from the most congested links. When considering LSPs staying in the network for long time, a reactive congestion control scheme behaves better than a preventive scheme, because it can dynamically adjust inefficiently routed paths.

Among the two proposed schemes, LFID has the advantage to minimize the number of LSPs to be routed in the network to balance the traffic and therefore has the lowest computational time.

REFERENCES

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

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

[3] R. Battiti and E. Salvadori. A Load Balancing Scheme for Congestion Control in MPLS Networks. Technical report, Università di Trento, Dipartimento di Informatica e Telecomunicazioni, November 2002.

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

[5] M. Brunato, R. Battiti, and E. Salvadori. Load Balancing in WDM Networks through Adaptive Routing Table Changes. In *Networking*, number 2345 in Lecture Notes in Computer Science, pages 289–301, Pisa - Italy, May 2002. Springer Verlag.

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

[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] 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 International Conference on Computer Communications and Networks*, pages 107–113, Las Vegas - Nevada, October 2000.

[9] 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.

[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] 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.