A Load Balancing Scheme for Congestion Control in MPLS Networks *

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

Abstract

In this paper we develop a Load Balancing scheme for networks based on the MPLS framework. The proposed algorithm (DYLBA - Dynamic Load Balancing Algorithm) implements a local search technique where the basic move is the modification of the route for a single Label Switched Path. Experiments under a dynamic traffic scenario show a reduced rejection probability especially with long-lived connection requests, thus providing a better use of resources when compared to existing constraint-based routing schemes for traffic engineering in MPLS networks.

1. Introduction

One of the most interesting applications of MPLS in IP-based networks is Traffic Engineering (TE) [3]. The main objective of TE is to optimize the performance of a network through an efficient utilization of the network resources. The optimization may include the careful creation of new Label Switched Paths (LSPs) through an appropriate path selection mechanism, the re-routing of existing LSPs to decrease the network congestion and the splitting of the traffic between many parallel LSPs.

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.

The paper is organized as follows. Section 2 provides a brief overview of Traffic Engineering for congestion control in MPLS networks. Then the context and the motivations for our proposal are highlighted in Section 3, while the proposed algorithm is explained in Section 4. The results are analyzed in Section 5.

*This work was supported by the Italian Ministry for University and Scientific Research under the ADONIS (Algorithms for Dynamic Optical Networks based on Internet Solutions) project and FIRB-CNIT "Grid computing".

2. Traffic Engineering for Congestion Control 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 on the electronic 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.

The IETF RFC 3272 classifies congestion control schemes according to the following criteria [2]:

- **Response time scale**: it can be characterized as *long* when it refers to capacity upgrades of the network carried out in weeks-to-months time scale, *medium* (minutes, days) when it refers to response schemes relying on a measurement system monitoring traffic distribution and network resources utilization that subsequently provides feedback to online or offline traffic engineering mechanisms (e.g. to set-up or to adjust some LSPs in MPLS networks to route traffic trunks away from congested resources), *short* (picoseconds, seconds) when it refers to packet level processing function such as passive or active queue management systems (e.g. Random Early Detection - RED).
- **Reactive vs. preventive**: reactive congestion management policies react to congestion problems by initiating relevant actions to reduce them, while preventive policies prevent congestion on the basis of estimates of future potential problems (e.g. distribution of the traffic in the network).

Proceedings of the Eighth IEEE International Symposium on Computers and Communication (ISCC'03)
1530-1346/03 $17.00 © 2003 IEEE

- **Supply side vs. demand side**: supply side congestion management policies increase the capacity available to traffic demand in order to decrease congestion (e.g. balancing the traffic all over the network), while demand side congestion management policies control the traffic to alleviate congestion problems.

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, 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 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 [13] with two counterexample topologies, 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 re-optimization 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 to perform comparisons.

3. 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. In the rest of the paper we will consider only the routing of bandwidth guaranteed connections. As in [9], we assume that Service Level Agreements (SLA) are converted into bandwidth requirements for the LSP. We assume also an online context with connection requests arriving 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 itself. 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 the path congestion. 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*.

Our Load Balancing problem can be defined as following:

> LOAD BALANCING — Given a physical network and an existing traffic requirement between every ingress-egress pair (bandwidth required per connection), find a routing of the LSPs to maximize the available capacity in the network.

4. 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 algorithms in optical networks [10, 12]. The idea is to minimize the congestion of the network by performing local modifications. For each tentative move, the most congested link is located and one of its crossing LSPs is rerouted along an alternate path.

Figure 1 gives the pseudo-code of the proposed Dynamic Load Balancing Algorithm (DYLBA). The scheme is similar to the congestion control mechanism introduced in [5, 6], that is based on an IP context where connections are routed through a destination-based routing. 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. The algorithm is triggered only when the set-up of a new LSP causes the detection of network congestion (when only x residual bandwidth is left on some link). First, a set of alternate paths to reroute an LSP crossing one of the congested links is found. Once the most promising move is selected, the rerouting of the traffic over the alternate LSP is executed.

Let us define the notations and explain the meaning of the functions and variables. The most congested links, identified by the minimum available capacity in the network, are collected in the set $congestedLinkSet$ which is populated through the function $calculateNetworkLoad$. The set $candRerSet$ contains paths that are candidate to replace those passing through the most congested network links.

DYNAMIC LOAD BALANCING ALGORITHM

```
1. <congestedLinkSet> ← calculateNetworkLoad (x)
2. bestCandidateLoad ← +∞
3. candRerSet ← ∅
4. for each link (cFrom, cTo) ∈ congestedLinkSet
5.       for each LSP_i crossing (cFrom, cTo)
6.             removePartialLoad (LSP_i)
7.             find an alternate path A_LSP_i for LSP_i
8.             vl ← load on the alternate path A_LSP_i
9.             if (vl = bestCandidateLoad)
10.                 candRerSet ← candRerSet ∪ {A_LSP_i, LSP_i}
11.            else if (vl < bestCandidateLoad)
12.                 bestCandidateLoad ← vl
13.                 candRerSet ← {A_LSP_i, LSP_i}
14.            restorePartialLoad (LSP_i)
15. if (candRerSet ≠ ∅)
16.      {A_LSP_j, LSP_j} ← pickRandomElement (candRerSet)
17.      reroute traffic from LSP_j to A_LSP_j
```

Figure 1. The Dynamic Load Balancing Algorithm.

The first part includes the core of the algorithm (lines 4–14). We consider each congested link in $congestedLinkSet$, identified by its endpoints ($cFrom$, cTo). The internal loop collects the moves (i.e. the alternate LSPs) that lead to minimize the overall network congestion. For each LSP crossing the link ($cFrom$, cTo), one tries to reroute the path itself on an alternate route, accepting the move even if the new path does not increase the available capacity of the network. To do this, one temporarily removes the load of the LSP from the current link, and calculate a new path starting from the LER which originated the LSP itself, provided that the congested link is avoided. The best alternate paths in terms of maximum load are collected into the $candRerSet$. In particular, the current minimum is stored in $bestCandidateLoad$. If the load obtained after this LSP reroute is equal to $bestCandidateLoad$, then the alternate LSP is added to the candidate set; if it is smaller, the candidate set is reinitialized to the current alternate LSP and its load is stored as the new best. At the end of the alternate path research, the partial load associated to the original LSP is reallocated in order to allow the search of new alternate paths for different LSPs.

If the resulting set $candRerSet$ is not empty then one random element is selected from it, and the rerouting is effectively executed in the network (lines 15–17).

5. Simulation results

The performance of the proposed algorithm is evaluated through two different set of experiments. The first is focused on the feasibility of DYLBA in a simulated MPLS network context by using an extension of the network simu-

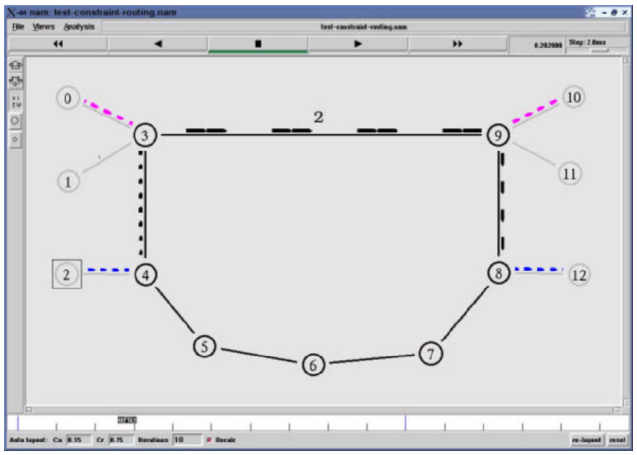

Figure 2. The simulation model.

lator (ns-2) called MNS [1, 11]. This simulator has the advantage to reproduce all the signalling mechanisms needed to set-up or tear-down LSPs in an MPLS network. The second set of experiments is based on a simulation program implemented in C++ and used to verify the path set-up rejection ratio of the proposed DYLBA algorithm compared with both Minimum-Hop routing Algorithm (MHA) and Minimum Interference Routing Algorithm (MIRA).

5.1. Feasibility of DYLBA in MPLS networks

Figure 2 represents the simulation topology, which has nodes 0, 1 and 2 as traffic source hosts, nodes 10, 11 and 12 as traffic sink hosts and nodes 3–9 as LSR nodes. The topology is a special case of the "concentrator topology" [13] where the MIRA scheme fails: in fact, all the links have capacity equal to 1 Mbps apart from the link between the LSR nodes 3 and 9, with capacity equal to 2 Mbps. To further highlight the limitations of MIRA, the delay associated to all links between nodes 4 and 8 is set to 20 msec, while for the other links it is set to 10 msec.

5.1.1. Network using MIRA scheme only. In order to show the main disadvantages of MIRA, two examples are explained in the following.

Example 1. Let's consider first the set-up of three LSPs with the same bandwidth in the network depicted in Figure 2, where each LSR applies the MIRA algorithm to obtain the constraint-based route. The resulting explicit route for each LSP set-up in the specified order is:

Set-up order	Mbps	S	D	Explicit route
LSP_1	1.0	2	12	4–3–9–8
LSP_2	1.0	1	11	3–9
LSP_3	1.0	0	10	blocked

As shown in [13], according to MIRA algorithm, the link between the nodes 3 and 9 is not a critical link for any individual ingress egress pair, while it is when all ingress-egress pairs are considered. This explains why LSP_1 is routed through nodes 4–3–9–8 thus blocking the request LSP_3.

Example 2. Another example showing the inefficiencies of MIRA is the following. The three LSPs have now different values of bandwidth demand. The resulting explicit route for each LSP set-up in the specified order is:

Set-up order	Mbps	S	D	Explicit route	Average delay
LSP_1	0.6	2	12	4–3–9–8	66.7 ms
LSP_2	0.6	1	11	3–9	33.84 ms
LSP_3	1.0	0	10	3–4–5–6–7–8	130.58 ms

This second example shows that MIRA could lead to bad optimized paths throughout the network because, if LSP_1 is torn-down just after the set-up of the LSP_3, LSP_3 gets routed over an inefficient path, causing an unjustified high end-to-end delay to the carried traffic (130.58 msec). This delay could dangerously affect the overall QoS of the network, e.g. by imposing long routes to LSP that could carry delay-sensitive traffic.

5.1.2. Network using DYLBA algorithm. To overcome the limitations highlighted in the previous section, we run the same simulations on MNS where the proposed algorithm (DYLBA) is implemented to properly balance the load throughout the MPLS network.

Example 1. In the first example, the MIRA scheme failed to route the third LSP. By using DYLBA, the set-up of the three LSPs produces these results:

Set-up order	Mbps	S	D	Explicit route	Notes
LSP_1	1.0	2	12	4–3–9–8	
LSP_2	1.0	1	11	3–9	3→9 congested
LSP_1	1.0	2	12	4–5–6–7–8	LSP rerouted
LSP_3	1.0	0	10	3–9	

In fact, as soon as the link between nodes 3 and 9 gets congested, the algorithm is executed, and an alternate path for LSP_1 is found. Once LSP_1 is rerouted away from the link 3–9, LSP_3 can found its route over the shortest path.

Figure 3 shows the throughput of the traffic collected by the destination nodes 10, 11 and 12, corresponding to LSP_1, LSP_2 and LSP_3 respectively. The jitter suffered by the traffic carried by LSP_1 due to the path rerouting is lower than 50 msec, while the end-to-end delay for each LSP is listed in the following table:

Set-up order	Mbps	S	D	Average delay
LSP_1	1.0	2	12	90.84 ms
LSP_2	1.0	1	11	34.47 ms
LSP_3	1.0	0	10	35.29 ms

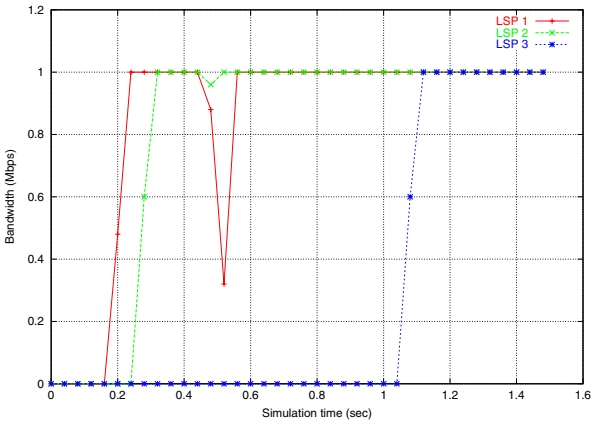

Figure 3. Example 1: the throughput for each LSP.

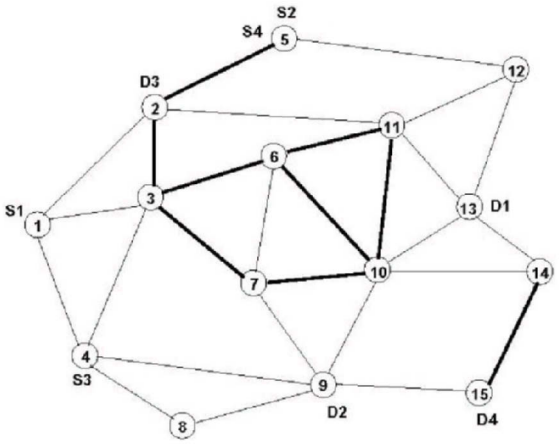

Figure 4. The network topology used in the simulations.

Example 2. In this example, the MIRA scheme badly routed the LSP demands, showing to be inefficient. With our DYLBA algorithm, the set-up of the three LSPs produces these results:

Set-up order	Mbps	S	D	Explicit route	Notes
LSP_1	0.6	2	12	4–3–9–8	
LSP_2	0.6	1	11	3–9	3→9 congested
LSP_1	0.6	2	12	4–5–6–7–8	LSP rerouted
LSP_3	1.0	0	10	3–9	

The most interesting result of this example can be highlighted by considering the end-to-end delay for each LSP:

Set-up order	Mbps	S	D	Average delay
LSP_1	0.6	2	12	90.93 ms
LSP_2	0.6	1	11	34.99 ms
LSP_3	1.0	0	10	33.99 sec

By comparing the average delays for LSP_1 and LSP_3 when DYLBA algorithm is applied with the same delays when only the MIRA scheme is working, it can be noticed that the traffic flowing through LSP_1 increases its delay by 24.23 msec due to the rerouting of the path itself, while the traffic flowing through LSP_3 decreases dramatically its delay by 96.59 msec. This is due to the capacity left free by DYLBA so that, at the set-up time, LSP_3 can cross its shortest path through the network.

5.2. Path set-up rejection ratio

This section describes the set of experiments used to compare the performance of DYLBA in term of path set-up rejection ratio with MHA and MIRA. These experiments are carried out by using network topology of [9], see Figure 4. 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.

All the experiments are carried out by considering the dynamic behavior of our algorithm (DYLBA) compared with both MHA and MIRA routing schemes. A critical parameter in our algorithm is the threshold used to detect congested links (see the parameter x in Section 4 corresponding to the residual bandwidth left on a link). LSPs arrive between each ingress-egress pair according to a Poisson process with an average rate λ, and the 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.

Table 1 shows the behavior of our algorithm (indicated by D(x)) with two different values of threshold x: 0.01 and 0.1, and compares it with both Minimum-Hop routing Algorithm (MHA) and Minimum Interference Routing Algorithm (MIRA). Each element of the table reports the average number of rejected LSP over 5 run trials and its relative standard deviation (within brackets).

The first two sets of experiments consider that bandwidth demands for LSPs are uniformly distributed between 1 and 3 units. By using the same traffic distribution considered in [9] (λ/μ =150 for each ingress-egress router pair), DYLBA(0.01) performs slightly better than MIRA on aver-

Table 1. Number of blocked requests

Algo	Bw $\in \mathcal{U}[1,3]$		Bw $\in \mathcal{U}[1,12]$	
	$\lambda/\mu = 150$	$\lambda/\mu = 200$	$\lambda/\mu = 150$	$\lambda/\mu = 200$
MHA	1211 (40)	2702 (65)	5634 (41)	6310 (48)
MIRA	844 (38)	2442 (86)	5526 (26)	6198 (50)
D(0.1)	859 (45)	2360 (66)	5356 (49)	6005 (51)
D(0.01)	818 (48)	2266 (75)	5395 (42)	6040 (39)

age, while DYLBA(0.1) performs roughly the same. For an LSP's average lifetime longer than the previous one (λ/μ = 200), DYLBA performs better than MIRA.

The second two sets of experiments consider LSPs with higher capacity on average, i.e. the bandwidth demands are uniformly distributed between 1 and 12 units. In this case our algorithm performs always better than MIRA. This can be explained by considering that using LSPs with bigger bandwidth requests on average, the rerouting of a single path from a congested link rapidly decreases the overall network congestion. Furthermore, being the computational complexity of our algorithm proportional to the number of LSPs per congested link, the local search for the better LSP to reroute will last few iteration cycles.

Another interesting behaviour of our algorithm comes from the analysis of the blocked request for two different values of (λ/μ). Smaller values mean that an LSP will stay for a shortest time in the network on average, and viceversa while considering the larger value. The experiments demonstrate that MIRA algorithm performs better than DYLBA when the LSPs have short life in the network on average, while our algorithm can improve the rejection probability when the LSPs lives for long time in the network.

6. Conclusions

In this paper a new online algorithm to dynamically balance the load in an MPLS network has been presented. The feasibility of the proposed algorithm has been shown through an implementation in an MPLS network simulator [1]. Simulation results show that our algorithm performs better than MIRA in specific condition of network traffic, by reducing the LSP rejection probability and the average end-to-end delays.

An interesting direction to extend our work is to consider the use of our algorithm in the context of G-MPLS optical networks, where one can integrate resource information at both the IP and the optical layers.

Acknowledgments

We would like to thank Mikalai Sabel for the implementation of the C++ software needed for the experimental tests.

References

[1] G. Ahn and W. Chun. Design and Implementation of MPLS Networks Simulator (MNS). *http://flower.ce.cnu.ac.kr/˜fog1/mns/*.

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

[3] D. O. Awduche and B. Jabbari. Internet Traffic Engineering using Multi-Protocol Label Switching (MPLS). *Computer Networks*, (40):111–129, Sept. 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, Sept. 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*, Sept. 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, Nov. 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, Oct. 2000.

[9] K. Kar, M. Kodialam, and T. Lakshman. Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications. *IEEE Journal on Selected Areas in Communications*, 18(12):2566 –2579, Dec. 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] V. Project. Network Simulator - V.2.1b9. *http://www.isi.edu/nsnam/ns/*.

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

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