COST279TD(02)041

Load Balancing by MPLS in Differentiated Services Networks

Riikka Susitaival, Jorma Virtamo, Samuli Aalto [1]

Helsinki University of Technology

Abstract

Multi Protocol Label Switching (MPLS) as a new technique assigns a short label to each packet at the ingress node of MPLS network and packets are forwarded according to these labels. The capability of MPLS of explicit routing as well as of splitting of the traffic on several paths allows load balancing. The report concentrates on two previously known approximations of the minimum-delay algorithm presented by Gallager. The first approximation defines the paths using the LP-optimization and after that allocates traffic using the NLP-optimization. The second approximation divides traffic into parts and routes them consecutively using Dijkstra's algorithm. Using these load balancing algorithms from the literature as a starting point, the main goal of this report is to develop optimization algorithms that differentiate classes in terms of mean delay. The differentiation is achieved by the use of both routing and WFQ-scheduling. Both optimal and approximative algorithms are developed for the joint optimization of the WFQ-weights and routing. The load balancing algorithms and the algorithms to differentiate classes are implemented and tested. As a result it is found that the use of the approximations simplifies the optimization problem but still provides results that are near to optimal. The algorithms that try to differentiate traffic classes by both routing and WFQ-scheduling provide the best performance.

1 Introduction

With Internet Protocol (IP), datagrams are forwarded on packet-by-packet basis. In the conventional IP routing, forwarding decisions are made independently in each router, based on the packet's header and precalculated routing tables. On the other hand, Asynchronous Transfer Mode (ATM) and Frame Relay (FR) are widely used connection oriented technologies that use virtual circuits, which are set up by signalling protocol beforehand. There are several technologies to run IP over ATM, but these techniques have remarkable scaling problems [1, 2].

MPLS (Multi Protocol Label Switching) is a flexible technology that enables new services in IP networks and makes routing more effective. It combines two different approaches, datagram and virtual circuit, as a compact technology. MPLS is based on short fixed length labels, which are assigned to each packet at the ingress node of the MPLS cloud. These labels are used to make forwarding decisions at each node. The assignments of a particular packet to a particular FEC (Forwarding Equivalence Class) are made only once. This simplifies and improves forwarding. The architecture of MPLS is defined in [3].

One of the most significant applications of MPLS is Traffic Engineering. Traffic Engineering (TE) concerns performance optimization of operational networks. It handles technological applications and scientific prin-

[1]E-mail: {riikka.susitaival, jorma.virtamo, samuli.aalto }@hut.fi

ciples to the measurement, modelling, characterization and control of the Internet traffic and applications to achieve specific performance objectives [4].

Traffic Engineering using MPLS provides mechanisms to route traffic that have equal starting point and destination along several paths. The capability to split traffic offers several advantages. Traffic can be routed successfully in the case of link failures using alternative paths, for example. However, the most important benefit of traffic splitting is the ability to balance load. The load balancing reduces congestion and therefore improves the performance of the network. Several load balancing algorithms have been developed for various networks, such as ATM networks. These methods could be adapted to MPLS networks also.

The IP routing paradigm offers services only on the best-effort basis. However, the demand to provide different services to different customers is evident. Quality of Service (QoS) is under a widespread discussion today. Two notable architectures, Integrated Services [5] and Differentiated Services [6], have been introduced in the literature. These architectures provide principles and goals to develop current IP networking.

MPLS and its Traffic Engineering capabilities could provide technical support to the implementation of QoS. The differentiation of services can be obtained by an alternative flow allocation that has the same principles as the load balancing methods. In order to make differentiation more effective, scheduling mechanisms, like WFQ-scheduling, can be utilized in the same context.

Our goal is to develop routing methods that optimize differentiation of experienced service of different classes in terms of their mean delays. We use load balancing methods as a starting point in the further development. These methods make an attempt to balance load in the network and thus achieve minimum mean delay.

The routing methods to be introduced are divided into two types. The first type tries to optimize only flow allocation so that differentiation is achieved. The traffic class that should achieve a small mean delay compared to the other classes is routed along the path that is not congested, for example. The second type of methods makes use of WFQ-weights. In each node, each service class has a WFQ-weight that determines, which proportion of the bandwidth is assigned to that service class. Also these methods optimize the flow allocation by minimizing the mean delay.

The rest of this report is organized as follows: In the second section we concentrate on the three previously known load balancing algorithms, which are the minimum-delay routing, the flow allocation algorithm using LP- and NLP-optimization and the heuristic approach to allocate traffic. In section 3 we introduce flow allocation methods that differentiate traffic classes by routing only. We present the flow allocation model that makes use WFQ-scheduling in section 4. We develop both optimal and approximative algorithms. In section 5 we present numerical results of all algorithms. Finally, section 6 makes some conclusions.

2 Load balancing algorithms

IP routing routes packets based on shortest path algorithms. It does not take into account the congestion of links when determining paths. However, using links that are congested can result in very long delays compared to an optimal flow allocation.

The possibility to use predetermined paths in routing allows one to allocate traffic effectively. Technologies where this approach is possible are ATM virtual circuits and MPLS, for example. Load balancing methods make an attempt to balance load in the network and therefore achieve better performance in terms of delay. The basic optimization problem minimizes the mean delay in the network. The use of the link delays of M/M/1-queues leads to a non-linear optimization problem (NLP). Many exact algorithms have been introduced to this optimization, the most famous one being Gallager's algorithm from year 1977 [7].

The flow allocation that minimizes the mean delay of network can be approximated in many ways. One coarse but simple approximation is to minimize the maximum link delay. This optimization function leads to a linear optimization program (LP), which is easier to solve than NLP-optimization.

In this section we introduce three flow allocation algorithms presented previously in the literature. The first algorithm is minimum-delay routing by Gallager [7]. The algorithm results in an optimal flow allocation. The second algorithm makes an attempt to reduce computation time of Gallager's routing method by solving paths by minimizing the maximum link load (LP-optimization) and after that by allocating traffic to paths using the mean delay as optimization function (NLP-optimization) [8]. Finally we introduce an algorithm that divides traffic into parts and allocates them using Dijkstra's algorithm [9].

2.1 Minimum-delay routing

Gallager defines in [7] an algorithm that minimizes the delay in a quasi-static network using distributed or centralized computation. The algorithm establishes the routing tables in the individual nodes based on the periodic information exchange between adjacent nodes. With successive updates of the routing tables, the average delay per message converges to the minimum average delay over all routing assignments. Traffic to each destination is guaranteed to be loop free at each iteration of the algorithm.

The formulation of the problem according to Gallager is as follows: Let L be the set of links, $L = \{(i,k) :$ there exists a link from node i to node $k\}$. Let $r_i(j)$ be the expected traffic (bit/s) entering the network at node i and destined for node j, $t_i(j)$ be the total expected traffic at node i and destined for node j, $\phi_{ik}(j)$ be the fraction of the node flow $t_i(j)$ that is routed over link (i,k) and b_{ik} be the capacity of link (i,j). Since the node flow $t_i(j)$ at node i is the sum of input traffic and the traffic routed to i from other nodes,

$$t_i(j) = r_i(j) + \sum_l t_l(j)\phi_{li}(j), \quad \forall i,j. \tag{1}$$

Now let

$$f_{ik} = \sum_j t_i(j)\phi_{ik}(j) \tag{2}$$

denote the expected traffic on link (i,k). The formulation of the optimization problem that minimizes the mean delay in the whole network is as follows:

$$\begin{aligned}
&\text{Minimize}\\
&E[D] = \frac{1}{\Lambda}\sum_{(ik)} \frac{f_{ik}}{b_{ik} - f_{ik}} = \frac{1}{\Lambda}\sum_{(ik)} \frac{\sum_j t_i(j)\phi_{ik}(j)}{b_{ik} - \sum_j t_i(j)\phi_{ik}(j)},\\
&\text{subject to the constraints}\\
&f_{ik} = \sum_j t_i(j)\phi_{ik}(j) < b_{ik}, \qquad \text{for each link } (ik),\\
&t_i(j) = r_i(j) + \sum_l t_l(j)\phi_{li}(j), \qquad \text{for each } i,j,
\end{aligned} \tag{3}$$

where Λ is the total offered traffic of the network.

2.2 Flow allocation using two-step algorithm

The algorithm that calculates an optimal flow allocation in terms of mean delay may be approximated by using the approach presented in [8]. The approximative algorithm solves first the paths to be used by LP-optimization and, after that, allocates the traffic to these paths using NLP-optimization. The time consuming LP-problem is calculated infrequently off-line, but the NLP-problem is calculated continuously.

2.2.1 Linear programming formulation and solution

The formulation of the problem presented in [8] is as follows: A network consists of N nodes. A pair of nodes (m,n) can be connected by a directed link (m,n) with bandwidth equal to $b_{(m,n)}$. The number of links is denoted by L and the topology is denoted by T, which is a set of node-pairs. Let $A \in \mathbb{R}^{N \times L}$ be the matrix, for which $A(i,j) = -1$ if link j directs to node i, $A(i,j) = 1$ if link j leaves from node i and $A(i,j) = 0$ otherwise.

The traffic demands are given by the matrix $[d_{(i,j)}]$, where i is the ingress and j is the egress node. $R_{(i,j)} \in \mathbb{R}^N$ is a row vector for each ingress-egress pair (i,j) such that $R_{(i,j),k} = d_{(i,j)}$, if k is the ingress node, $R_{(i,j),k} = -d_{(i,j)}$, if k is the egress node and $R_{(i,j),k} = 0$ otherwise. Demands and capacities are assumed to be constant (or average traffic rates). Let $x_{(i,j),(m,n)}$ be the allocated traffic of ingress-egress pair (i,j) on link (m,n). Then the total traffic on the link (m,n) is

$$X_{(m,n)} = \sum_{(i,j)} x_{(i,j),(m,n)}. \tag{4}$$

The pair-based flow formulation that minimize the maximum link load is as follows:

$$\begin{aligned}
&\text{Minimize } \Big[-\epsilon Z + \sum_{(m,n)} w_{(m,n)} \sum_{(i,j)} x_{(i,j),(m,n)} \Big] \\
&\text{subject to the constraints} \\
&x_{(i,j),(m,n)} \geq 0; \quad Z \geq 0, \\
&\sum_{(i,j)} x_{(i,j),(m,n)} + C_{(m,n)} Z \leq b_{(m,n)}, \qquad \text{for each } (m,n) \text{ with } b_{(m,n)} > 0, \\
&A x_{(i,j)}^T = R_{(i,j)}^T, \quad \text{for each } (i,j) \qquad \text{at each node } n,
\end{aligned} \tag{5}$$

Free parameter Z in (5) describes the minimum value of the proportional unused capacity. The last equation in (5) states that, at every node n, incoming traffic of each ingress-egress pair must be equal to outgoing traffic [10].

2.2.2 Defining paths from the LP-solution

When the LP-problem (5) is solved and variables $x_{(i,j),(m,n)}$ found, we have to find paths for each ingress-egress pair. However, the solution is not unique.

The algorithm to define paths to one ingress-egress pair is as follows: We have original topology the T, which consists of the set of directed links. Because one ingress-egress pair uses only part of the whole topology, we define a new topology $T'_{(i,j)}$, which consists of the links for which $x_{(i,j),(m,n)}$ differs from zero. Topology $T'_{(i,j)}$ is loop-free, because if there were loops, they would be reduced and the original allocation would not be optimal. Let $L'_{(i,j)}$ be the number of links in topology $T'_{(i,j)}$. We search all paths to one ingress-egress pair (i,j) by a breadth-first-search algorithm, which is defined below.

1. Let $P_{(i,j)}$ be the empty set of paths.
2. Find all links (i,k) from topology $T'_{(i,j)}$. Compose for each link (i,k) a subset $P'_{(i,j)}$, which consists of nodes i and k. Add $P'_{(i,j)}s$ to set $P_{(i,j)}$.
3. If the last component of subset $P'_{(i,j)}$ is egress node j, subset $P'_{(i,j)}$ is complete.
4. Consider one subset $P'_{(i,j)}$. Find all links (k,l), where k is the last node of $P'_{(i,j)}$, from topology T'. Compose for each link (k,l) a subset $P''_{(i,j)}$, which consists of the nodes of set $P'_{(i,j)}$ and node l. Now replace set $P'_{(i,j)}$ by sets $P''_{(i,j)}$ in set $P_{(i,j)}$. Repeat this to all subsets $P'_{(i,j)}$, which are not complete.

5. If there is some subset $P'_{(i,j)}$, which is not complete, go back to step 3, else stop the algorithm.

As a result of the algorithm, we have a set of possible paths $P_{(i,j)}$ for one ingress-egress pair. Let $K_{(i,j)}$ be the number of paths and $Q_{(i,j)} \in \mathbb{R}^{L'_{(i,j)} \times K_{(i,j)}}$ be the matrix, where

$$Q_{(i,j),(l,k)} = \begin{cases} 1, & \text{if path } k \text{ uses link } l \text{ of topology } T'_{(i,j)}, \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

for each ingress-egress pair (i,j). Let $Y_{(i,j),k}$ be the traffic allocation in path k. $Y_{(i,j),k}$s can be solved from matrix equation

$$Q_{(i,j)} Y_{(i,j)}^T = {x'}_{(i,j)}^T, \quad \text{for each } (i,j), \tag{7}$$

where $x'_{(i,j)} \in \mathbb{R}^{L'_{(i,j)}}$ is the flow vector for each ingress-egress pair (i,j). The system defined by matrix equation (7) can be over- or underdetermined, because the number of paths differs from the number of links. So it is useful to determine the pseudoinverse of $Q_{(i,j)}$ when solving the equations. Finally, if an element of the solution matrix $Y_{(i,j),k}$ differs from zero, ingress-egress pair (i,j) uses path k, else not. So reducing these paths from path set $P_{(i,j)}$ we get the actual path set.

2.2.3 Non-linear programming formulation and solution

Now the set of paths is already solved with the linear programming and therefore the size of problem is reduced. The objective is to find an optimal flow allocation to these paths. Let $K = \sum_{(i,j)} K_{(i,j)}$ be the total number of paths in the network, $P_{(i,j),k}$ be the path k of node-pair (i,j), and $\phi_{(i,j),k}$ be the fraction of $d_{(i,j)}$ allocated to path $P_{(i,j),k}$.

The structure of path k is defined by $\delta_{(m,n),(i,j),k}$ as follows:

$$\delta_{(m,n),(i,j),k} = \begin{cases} 1, & \text{if path } p_{(i,j),k} \text{ uses link } (m,n), \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Note that the traffic allocation $Y_{(i,j),k}$ presented in section 2.2.2 is a special case of traffic allocation $d_{(i,j)}\phi_{(i,j),k}$. Our objective is to minimize the mean delay of the total network. So the optimization problem is as follows:

$$\begin{aligned} &\text{Minimize} \\ &\frac{1}{\Lambda} \sum_{(m,n)} \frac{\sum\limits_{(i,j),k} \delta_{(m,n),(i,j),k} d_{(i,j)} \phi_{(i,j),k}}{b_{(m,n)} - \sum\limits_{(i,j),k} \delta_{(m,n),(i,j),k} d_{(i,j)} \phi_{(i,j),k}}, \\ &\text{subject to the conditions} \\ &\sum_{(i,j),k} \delta_{(m,n),(i,j),k} d_{(i,j)} \phi_{(i,j),k} < b_{(m,n)}, && \text{for each } (m,n), \\ &\phi_{(i,j),k} \geq 0, && \text{for each } (i,j), k, \\ &\sum_{k=1}^{K_{(i,j)}} \phi_{(i,j),k} = 1, && \text{for each } (i,j), \end{aligned} \tag{9}$$

where Λ is the total offered traffic of the network.

In paper [11] Carpenter et al. present three enhancements to the routing algorithm introduced in [8]. The first enhancement is the capability to select additional paths to the solution in advance, which increases the robustness of the routing. The second enhancement is the ability to choose only a single path to one ingress-egress node-pair. The third enhancement is the capability to use admission control when it is impossible to carry all offered traffic.

2.3 Heuristic approach

The cost of achieving the optimal routing can be divided into two different parts, the first part is the classification cost in ingress node, which depends on the granularity and the second part is the forwarding cost. The traffic granularity refers to the level of traffic aggregation [9]. The finer the level of the granularity the finer the traffic partitioning to different paths.

The heuristics presented in [9] to allocate traffic into the network using a particular level of granularity is very simple. Depending on the level of granularity, traffic demands from ingress to egress node are divided into streams (e.g. using some hashing function). Streams are added one at a time onto route which minimizes delay.

Let $g \in \mathbb{N}$ be the level of granularity. Traffic demands from ingress node i to egress node j are given by a matrix $[d_{(i,j)}]$. We compose a list D consisting of ingress-egress pairs and their traffic intensities. The elements in the list D are sorted in (i) ascending order in terms of their traffic demand, (ii) descending order in terms of their traffic demand, (iii) descending order in terms of their shortest mean delay from ingress node to egress node calculated in the empty network.

We route each element of the list D sequentially one at the time. The algorithm to route element l is as follows:

1. Calculate link costs $c^{(l)}_{(m,n)}$ using the mean delay of an $M/M/1$-queue:

$$c^{(l)}_{(m,n)} = \frac{1}{b_{(m,n)} - (a^{(l)}_{(m,n)} + \frac{d^{(l)}}{g})}, \tag{10}$$

   where $[a^{(l)}_{(m,n)}]$ is traffic matrix that contains all traffic that is already routed to link (m, n) (in the first phase $a^{(1)}_{(m,n)} = 0$ for each (m, n)).

2. Calculate the shortest path using Dijkstra's algorithm. Use $c^{(l)}_{(m,n)}$ as link cost.

3. Add traffic flow $d^{(l)}/g$ to these components of matrix $[a^{(l)}_{(m,n)}]$, which are included in the solution of Dijkstra's algorithm.

4. Repeat phases 1-3 g times, so that all traffic of the ingress-egress pair is routed.

The procedure above is repeated to each ingress-egress pair. Finally, the mean delay in the whole network can be calculated using

$$C(\gamma) = \frac{S}{\Lambda} \sum_{(m,n)} \frac{a_{(m,n)}}{b_{(m,n)} - a_{(m,n)}}, \tag{11}$$

where S is the average packet size and Λ is the total offered traffic. The total implementation of the heuristics needs the level of granularity times the number of ingress-egress pairs calculations of Dijkstra's algorithm.

3 Methods to achieve differentiation by routing

In this section we present flow allocation methods, which try to differentiate traffic classes by routing only. The traffic classes with a higher priority are routed along the paths that are not congested, for example. We differentiate classes by minimizing the sum of weighted mean delays, by fixing the ratio of mean delays and using a heuristic approach. The heuristic approach forces the lower priority classes to avoid the paths used by the higher priority classes.

3.1 Optimization using cost weights

We consider the situation where the total traffic is divided into traffic classes, into the gold, silver and bronze classes, for example. The gold class has the highest priority and the bronze class the lowest priority. Each traffic class l has its own traffic matrix $[d_{l,(i,j)}]$, where i is the ingress node and j is the egress node. $R_{l,(i,j)} \in \mathbb{R}^N$ is an array for each class l and ingress-egress pair (i,j), where $R_{l,(i,j),k} = d_{l,(i,j)}$, if k is the ingress node, $R_{l,(i,j),k} = -d_{l,(i,j)}$, if k is the egress node and $R_{l,(i,j),k} = 0$ otherwise. The total traffic offered by class l is denoted by Λ_l. Let $x_{l,(i,j),(m,n)}$ be the allocated traffic of ingress-egress pair (i,j) of class l on link (m,n). So the total traffic of class l on link (m,n) is

$$X_{l,(m,n)} = \sum_{(i,j)} x_{l,(i,j),(m,n)}, \quad \text{for each } l, (m,n). \tag{12}$$

Let w_l be the cost weight associated to traffic class l. Additional notation used is the same as in section 2.2.

The purpose is to divide traffic into paths so that classes with higher priority, like gold class, achieve smaller mean delay than other classes. So the cost of delay is highest in the gold class and so on. The optimization problem, where we minimize the sum of the weighted mean delays of the classes, is as follows:

$$\begin{aligned} &\text{Minimize} \\ &\sum_l w_l E[D_l] = \sum_{(m,n)} \frac{\sum_l \frac{w_l X_{l,(m,n)}}{\Lambda_l}}{b_{(m,n)} - \sum_l X_{l,(m,n)}}, \\ &\text{subject to the constraints} \\ &\sum_l X_{l,(m,n)} < b_{(m,n)}, && \text{for each } (m,n), \\ &A x^T_{l,(i,j)} = R^T_{l,(i,j)}, && \text{for each } l, (i,j), \end{aligned} \tag{13}$$

where traffic allocations $x_{l,(i,j),(m,n)}$ are decision variables.

When the cost weights of different classes differ enough, the optimization function tries to minimize the mean delays of high priority classes at the expense of lower priority classes. The high priority class can be routed through the shortest path and the other classes through some other path in order to provide enough capacity to the high priority class. As a result, the routing obtained by the optimization function above differs from the routing obtained by the load balancing, because in the load balancing approach the delays in the nodes are balanced to be equal and the differentiation could occur only if the paths are of different length.

3.2 Optimization with a fixed mean delay ratio

Now we fix the ratio of the mean delays of various classes to some value in order to differentiate classes. For example, in the case of two classes (gold and silver), the ratio of the mean delays between the silver and the gold class could be fixed to parameter q:

$$\frac{E[D_{l_2}]}{E[D_{l_1}]} = \frac{\dfrac{1}{\Lambda_{l_2}} \displaystyle\sum_{(m,n)} \frac{w_{l_2} X_{l_2,(m,n)}}{b_{(m,n)} - \sum_{l} X_{l,(m,n)}}}{\dfrac{1}{\Lambda_{l_1}} \displaystyle\sum_{(m,n)} \frac{w_{l_1} X_{l_1,(m,n)}}{b_{(m,n)} - \sum_{l} X_{l,(m,n)}}} = q. \tag{14}$$

After that the optimization can be done by minimizing the mean delay of either class or the sum of the mean delays like in optimization problem (13).

Compared to the optimization in the previous section, this approach does not include cost weights and the actual ratio of the mean delays is known before the optimization. It is useful to constraint the ratio of the mean delays to some small domain in order to make the optimization procedure easier.

3.3 Heuristics

There exists a demand to provide also simple routing methods that can be implemented without heavy optimization. The approach that routes traffic to the network near optimally but still obtains the differentiation in terms of mean delay tries to adapt the heuristic approach presented in section 2.3.

The heuristic approach in the case of two classes (gold and silver) is as follows:

1. The gold class is routed using heuristics introduced in 2.3.
2. The allocated traffic of the gold class (denoted by $a_{(m,n)}$) is multiplied by $1 + \Delta$.
3. The silver class is routed using heuristics introduced in 2.3.

The idea of the heuristics is that the links that are used by the gold class look more congested than they actually are. So the routing scheme tries to balance load by routing the silver class by some other way. That is, the artificial congestion forces the silver class to avoid links used by the gold class and therefore the gold class should achieve more bandwidth. The choice of the parameter Δ depends on how much there is traffic offered compared to the capacity of the network.

4 Methods to achieve differentiation in mean delay using WFQ-scheduling

The possibilities to provide differentiated services using routing only are limited. To achieve certain ratio of mean delays may lead up to disadvantageous routing because the low priority class is routed along long and congested paths in order to artificially obtain the desired ratio of the mean delay.

Weighted Fair Queueing as a packet scheduling mechanism divides bandwidth among the parallel queues. Each queue achieves a guaranteed bandwidth, which depends on the WFQ-weight of that queue and the link capacity. We try to find optimal routing that differentiates the quality of service by including the WFQ-weighs to the optimization function as free parameters.

Because WFQ-scheduling is a work-conserving discipline, the bandwidth that is guaranteed for a class in our model can be viewed as the lower bound. Actually, the bandwidth available to a class can be much greater as the other classes may not always use the bandwidth reserved for them.

The bandwidth of each link is shared according to WFQ-weights. We approximate the behavior of WFQ-scheduling as follows: Let $\gamma_{l,(i,j)}$ be the WFQ-weight that determines the proportion of total bandwidth that is given to class l. The bandwidth $b_{l,(m,n)}$ of class l on link (m,n) is thus

$$b_{l,(m,n)} = \gamma_{l,(m,n)} b_{(m,n)}, \quad \text{for each } l, (m,n). \tag{15}$$

The sum of the WFQ-weights of the classes must equal to one. As a result, we have changed over from the WFQ-scheduling system to the system of parallel independent queues with link capacities described above.

4.1 Optimization using cost weights

In this section we obtain the differentiation between classes by using the cost weights as in (13). The gold class gets the greatest cost weight and so on. The joint optimization of flow allocation and WFQ-weights where the sum of weighted mean delays is minimized is presented in (16). The optimization is referred to as "straightforward".

$$\begin{aligned}
&\text{Minimize} \\
&\sum_l w_l E[D_l] = \sum_l \frac{w_l}{\Lambda_l} \sum_{(m,n)} \frac{X_{l,(m,n)}}{\gamma_{l,(m,n)} b_{(m,n)} - X_{l,(m,n)}}, \\
&\text{subject to the constraints} \\
&X_{l,(m,n)} < \gamma_{l,(m,n)} b_{(m,n)}, && \text{for each } l, (m,n), \\
&A x^T_{l,(i,j)} = R^T_{l,(i,j)}, && \text{for each } l, (i,j), \\
&0 < \gamma_{l,(m,n)} < 1, && \text{for each } l, (m,n), \\
&\sum_l \gamma_{l,(i,j)} = 1, && \text{for each } (i,j),
\end{aligned} \tag{16}$$

where traffic allocations $x_{l,(i,j),(m,n)}$ and WFQ-weights $\gamma_{l,(m,n)}$ are decision variables.

4.1.1 Two-step algorithms

The optimization problem presented in (16) is quite heavy and time-consuming. If the allocated traffic exceeds the bandwidth, the value of the objective function goes up to infinity. The constraints that prevent the allocated traffic to exceed the bandwidth of a particular link have an effect on the value of objective function.

We introduce near optimal algorithms that make the size of the problem smaller and the calculation easier. The first two algorithms first allocate the traffic into the network and after that optimize WFQ-weights. The structure of both algorithms are as follows:

1. Allocate the traffic into the network without WFQ-weights so that the sum of mean delays is minimized. The formulation of the optimization algorithm is presented in section 3.1.
2. Fix the traffic allocation obtained in the first step.
3. Determine WFQ-weights using optimization problem (16). Now the number of free variables equals the number of WFQ-weights, i. e., the number of links in the network multiplied by the number of classes minus one.

The cost weights of the optimization function are selected twice. The difference between the first and second algorithm is that in the first two-step algorithm the cost weights of the first step are equal to one (referred to as "two-step, version 1") and in the second two-step algorithm the cost weights of the first and second steps are equal (referred to as "two-step, version 2").

The first two-step algorithm makes only use of WFQ-scheduling when trying to differentiate classes. It is reasonable since the flow allocation is optimal and artificial differences in mean delays do not appear in the first step. The second two-step algorithm instead utilizes both routing and WFQ-scheduling when differentiating classes and can therefore be closer to the optimal.

The third approximative algorithm makes use of the two-step algorithm presented in section 2.2. The paths are first calculated using the linear optimization that minimizes the maximum link load. Then the traffic is allocated to the paths using the non-linear optimization. The algorithm (referred to as "QoS-LP-NLP") is as follows:

1. Calculate the traffic allocation that minimizes the maximum link load using algorithm presented in section 2.2.1. Consider the traffic of all classes as one aggregate.
2. Calculate the paths corresponding to the traffic allocation of the previous step using the algorithm presented in section 2.2.2.
3. Allocate traffic to the paths obtained in step 2 and define WFQ-weights using the non-linear optimization that minimizes the mean delay of the network (compare to optimization problem (9)).

4.2 Fixing the link delay ratio

In the routing with WFQ-weights, if the ratio of mean delays is fixed like in the algorithm presented in section 3.2, the optimization problem is demanding. One possibility is to fix the mean delay ratios at the link level. If the lengths of paths of different classes do not differ significantly, this approach should result in the same mean delay ratio in the whole network.

We consider the case with two classes, gold and silver. We fix the ratio of link delays to parameter q, that is

$$\frac{E[D_{l_2,(m,n))}]}{E[D_{l_1,(m,n)}]} = \frac{\gamma_{l_1,(m,n)} b_{(m,n)} - X_{l_1,(m,n)}}{(1-\gamma_{l_1,(m,n)})b_{(m,n)} - X_{l_2,(m,n)}} = q, \quad \text{for each } (m,n). \tag{17}$$

The WFQ-weights $\gamma_{l_1,(m,n)}$ can be solved from equation (17) as a function of the traffic of both classes,

$$\gamma_{l_1,(m,n)}(X_{l_1,(m,n)}, X_{l_2,(m,n)}) = \frac{b_{(m,n)} + qX_{l_1,(m,n)} - X_{l_2,(m,n)}}{b_{(m,n)}(q+1)}, \quad \text{for each } (m,n). \tag{18}$$

The optimization problem of the flow allocation with the fixed ratio of mean delays is almost similar to optimization problem (16). The difference is that the WFQ-weights are not free parameters in the first approach. The optimization problem is as follows:

$$\begin{aligned}
&\text{Minimize}\\
&\sum_l w_l E[D_l] = \sum_l \frac{w_l}{\Lambda_l} \sum_{(m,n)} \frac{X_{l,(m,n)}}{\gamma_{l,(m,n)} b_{(m,n)} - X_{l,(m,n)}},\\
&\text{subject to the constraints}\\
&\gamma_{l_1,(m,n)} = \frac{b_{(m,n)} + qX_{l_1,(m,n)} - X_{l_2,(m,n)}}{b_{(m,n)}(q+1)}, && \text{for each } (m,n),\\
&X_{l,(m,n)} < \gamma_{l,(m,n)} b_{(m,n)}, && \text{for each } (m,n),\\
&Ax^T_{l,(i,j)} = R^T_{l,(i,j)}, && \text{for each } l,(i,j),\\
&0 < \gamma_{l,(m,n)} < 1, && \text{for each } l,(m,n),\\
&\sum_l \gamma_{l,(i,j)} = 1, && \text{for each } (i,j).
\end{aligned} \tag{19}$$

If we want to differentiate the classes by fixing the link delays only, the cost weights of the optimization function are equal to one (referred to as "fixed link delays"). We can also optimize the sum of the weighted mean delays (referred to as "fixed link delays and weights"). The problem is now how to determine the cost weights in relation to the ratio of link delays.

5 Numerical results

We have implemented three different load balancing algorithms, minimum-delay using non-linear optimization (referred to as "minimum-delay"), minimum-delay using two-level procedure (linear and non-linear optimization, referred to as "LP-NLP") and minimum-delay using heuristic approach (referred to as "heuristics"). We have tested the heuristic algorithm using different level of granularity. All three methods have been compared.

The routing methods that try to differentiate the mean delay between traffic classes, are tested also. In order to make optimizations simpler, we study only the case of two classes, gold and silver. The traffic matrices of both classes are equal, so that the comparison between the traffic classes is easier.

The formulations of the optimization problems were written using a General Algebraic Modelling System (GAMS), which is a high-level modelling language for mathematical programming problems [12]. It is specially useful for complex and large scale linear and non-linear optimization. GAMS-program calls a solver, which can be specified in the program. We have used solver module Minos 5 in our optimizations. The heuristics described in section 2.3 is implemented and tested using Mathematica.

The algorithms are tested in a known test-network, which consists of 10 nodes, 58 links and 72 ingress-egress pairs. The link capacities and the traffic demands of the ingress-egress pairs are available at web-page
http://brookfield.ans.net/omp/random-test-cases.html. In the case of the two traffic classes the traffic demands are half of the original demands.

5.1 Load balancing routing

We have used granularity levels 1, 2, 4, 8, 16 and 32, which correspond to the lengths of the prefixes 0, 1, 2, 3, 4 and 5, when studying the effect of the level of granularity on the mean delay. The results in Figure 1 are quite similar to the results of paper [9]. The level of granularity has a great impact on the mean delay. On the other hand, the use of a finer granularity than 8 does not decrease mean delay significantly. The policy

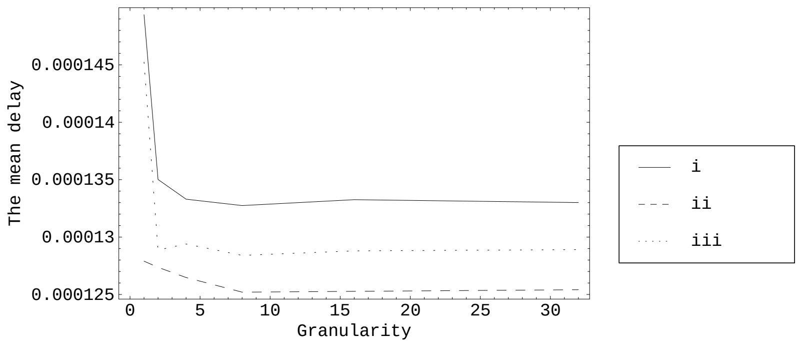


Figure 1: The mean delay as a function of granularity when streams are routed in (i) increasing order in terms of the traffic load, (ii) decreasing order in terms of the traffic load, and (iii) decreasing order in terms of the mean delay

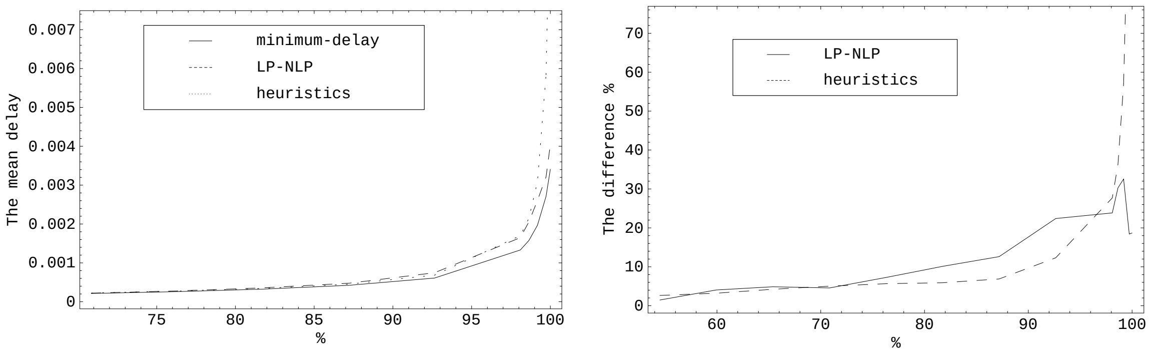


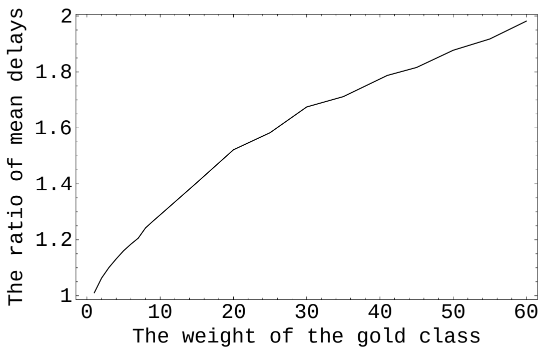
Figure 2: The mean delay and the relative deviation of the mean delay from the optimal as a function of the traffic load

to route first an ingress-egress pair that has the greatest traffic intensity seems to result in the smallest mean delay. The explanation to this is that the ingress-egress pairs with the greatest traffic intensity are the hardest problems to route and there is more free capacity and therefore more routing alternatives in the network in the early phase of the algorithm.

The mean delay and the relative deviation of the mean delay from the optimal as a function of the traffic load of the three routing methods is presented in Figure 2. With the heuristic approach, we use the granularity level 32, except in the cases of heavy load (the traffic load is over 95%) when the used granularity level is 128. We can see that the mean delays of different methods do not differ significantly. Only when the traffic load is near to the total capacity of the network is the performance of the minimum-delay routing notable. The mean delay using LP-NLP-routing and heuristics are quite similar, until the load is heavy, when LP-NLP-optimization provides a better result.

Table 1: The computation times of different optimizations in seconds

Optimization	Generation time	Execution Time
LP-optimization	0.27	0.27
NLP-optimization	0.02	0.02
minimum-delay	3.3	3.3



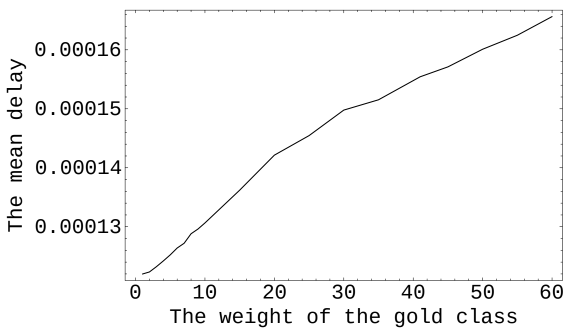


Figure 3: The ratio of mean delays and the mean delay as a function of the weight of the gold class

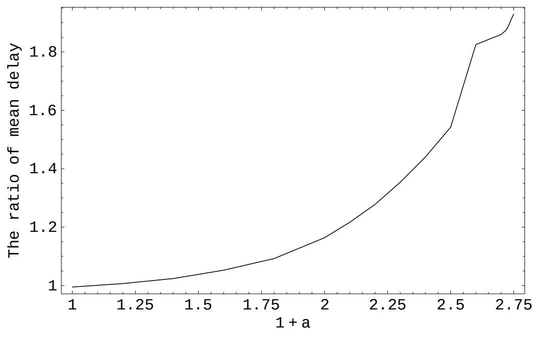


Figure 4: The mean delay as a function of $(1 + \Delta)$

Table 1 shows also that the computation time of the LP-NLP-optimization is ten times shorter than the computation time of the minimum-delay optimization as was to be expected as the original idea in [8] was to make the computation lighter. So the performance of the LP-NLP-approach is very good if the total traffic demand does not exceed 70% of the maximum load.

5.2 Methods to achieve differentiation using routing

The first algorithm in section 3 makes an attempt to differentiate the classes by optimizing the sum of weighted mean delays. As a result, the ratio of the mean delay of the silver class to the mean delay of the gold class and the growth of mean delay as a function of the cost weight of the gold class is presented Figure 3. We can see that the ratio of mean delays increases when the cost weight of the gold class increases and the cost weight of the gold class should be great in order to achieve differentiation. In the case where the traffic demand matrices of each class are equal, the flow allocation that differentiates the mean delay differs from the flow allocation of the optimal routing. So the mean delay increases when the cost weight of the gold class increases.

The heuristics routes first the gold class and multiplies the allocated traffic by some factor in order to make the silver class to avoid the links used by gold class. The ratio of mean delay as a function of multiplier $(1+\Delta)$ is presented in Figure 4. The maximum ratio that can be obtained using this approach is approximately 2, which may be too small for the required differentiation.

We take the mean delay of the total network as a function of the ratio of mean delay as a performance indicator. The increase in mean delay describes the cost of achieving a certain level of differentiation. All three methods are compared in Figure 5. The figure shows that the first and second optimizations generate the same result. However, the benefit of optimization function (14) is that the ratio of mean delays is known

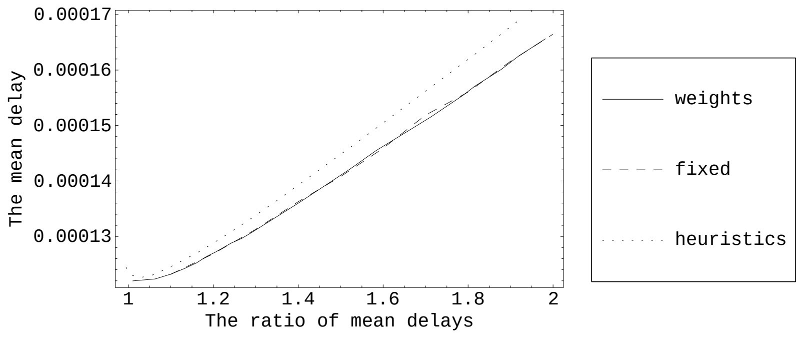


Figure 5: The mean delay as a function of the traffic load

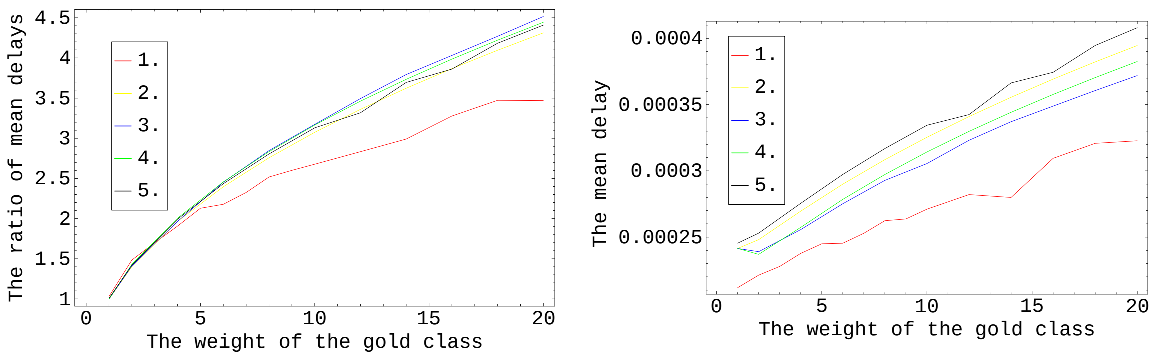


Figure 6: The ratio of mean delays and the mean delay as a function of the weight of the gold class

a priori, while the cost weights of optimization function (13) must be determined.

5.3 Methods to achieve differentiation by WFQ-scheduling

5.3.1 Using cost weights

We have implemented the optimization method that minimizes the weighted sum of mean delays straightforwardly and the optimization methods that divide the problem into two steps (introduced in section 4.1.1).

A near optimal routing can also be achieved using an iterative approach (referred to as "two-step, iterative"). The flow allocation and the WFQ-weights are optimized alternatingly. In the following optimizations we use the minimum-delay routing as the starting point ("two-step, version 1"). The number of iterations is ten, which means that ten flow allocations and ten WFQ-weight determinations are done in the optimization.

The ratio of the mean delay of the silver class to the mean delay of the gold class and the mean delay of all five algorithms are presented in Figure 6. The numbering of curves in the previous and following figures is explained in Table 2. All methods except the straightforward routing behave equally. The irregularities in the curve of the straightforward routing are perhaps a consequence of numerical errors in the optimization procedure. As the result, the cost weight to achieve a certain ratio of mean delay is more than ten times smaller in the optimization that makes use of WFQ-weights than in the optimization without WFQ-weights.

Table 2: The numbering of the algorithms used

	Algorithm
1.	straightforward
2.	two-step, version 1
3.	two-step, version 2
4.	two-step, 10 iterations
5.	QoS-LP-NLP
6.	fixed link delays
7.	fixed link delays and weights

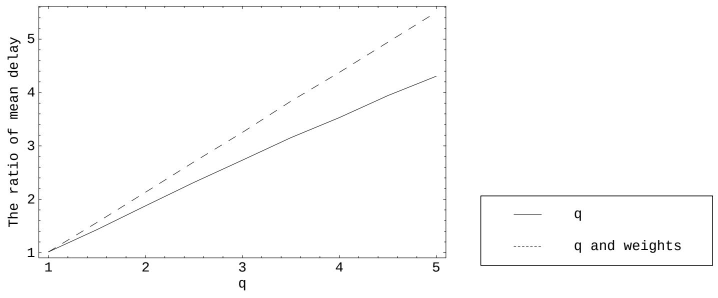


Figure 7: The ratio of mean delays as a function of q

5.3.2 Fixing the ratio of link delays

In order to simplify optimization and to provide differentiation also at the ingress-egress pair level, we have implemented optimizations that fix the ratio of link delays to some parameter q ("fixed link delays"). It is not guaranteed that the ratio of mean delays of different classes on the network level is the same as the ratio of link delays. The optimization problem was presented in section 4.2

We combine also cost weight w_{l_1} to the optimization function with the fixed ratio of link delays ("fixed link delays and weights"). In this approach the problem is how to determine both the parameter q and the cost weight w_{l_1}. We have implemented only one case, where the cost weight is three times greater than link delay ratio q.

In Figure 7 we present the relation between the ratio of link delays and the ratio of mean delays of different classes. The ratio of mean delays seems to be smaller than the ratio of link delays. The explanation is that the routing algorithm tries to balance traffic load by routing classes that achieve more bandwidth through long routes. The routing that uses cost weights also seems to provide a greater range for the ratio of mean delays than the routing that uses only parameter q. The reason is that the routing with cost weights utilizes also routing when trying to differentiate the classes.

5.3.3 Summary of algorithms with the WFQ-weights

Finally, we compare all the methods. The performance metric is the same as in section 5.2, the mean delay of total network as a function of the ratio of mean delays. The results are presented in Figure 8.

The straightforward optimization seems to have the smallest mean delay. The difference to other algorithms is significant when the ratio of mean delays is small. When the ratio is greater, the performance of the two-step algorithm that utilizes both routing and WFQ-weights ("two-step, version 2") is near to optimal.

As a conclusion, the differentiation methods that make use of both routing and WFQ-scheduling ("straight-

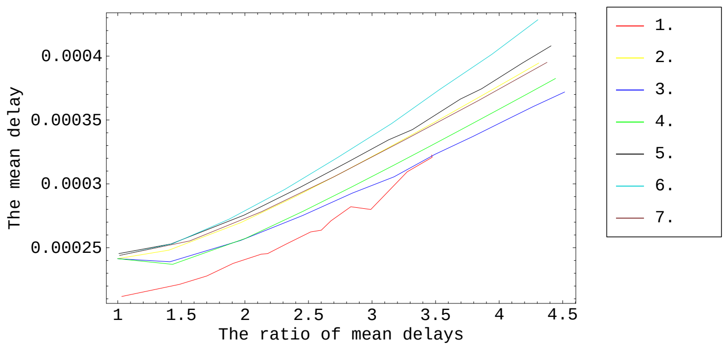


Figure 8: The mean delay of the network of as a function of the ratio of mean delays

forward", "two-step, version 2", "two-step, iterative", "fixed link delays and weights") perform better than the differentiation methods that make use of only WFQ-scheduling ("two-step, version 1", "QoS-LP-NLP" and "fixed link delays").

6 Conclusion

MPLS is a flexible new technique the key benefit of which is the support for Traffic Engineering that enables the use of explicit routes, for example. Explicit routing gives the capability to route traffic along paths that differ from those selected by IP routing. Traffic can also be split to several paths and load can thus be balanced.

Several load balancing algorithms have been introduced in the literature. We concentrated on three methods that try to minimize the mean delay of the network. As a result, we notice that the computation time of the two-step algorithm is ten times smaller than the computation time of the optimal routing. However, the mean delays of the algorithms differ significantly only if the load is near to the maximum.

We have used load balancing algorithms as a starting point when developing routing methods that try to differentiate traffic classes in terms of the mean delay. The performances of the algorithm that uses cost weights and the algorithm that fixes the ratio of mean delays are equal. The advantage of the latter algorithm is that the cost weights have not to be known in advance. The performance of the heuristic approach is a little poorer than that of the other two algorithms.

We have presented a model where the bandwidth of each link is shared among the traffic classes according to the WFQ-weights. The optimization problem is to minimize the mean delay. We have done this by minimizing the weighted sum of mean delays and using two-step approaches. We notice that the use of the algorithm that makes use of both routing and WFQ-scheduling gives the best result. However, the computation time of the two-step algorithm that uses both linear and nonlinear optimization is only one twentieth of that of the computation time of the other algorithms.

The bandwidth guaranteed by WFQ-scheduling to each class is the theoretical minimum. As a result, the actual ratio of mean delays may differ from the result obtained by the optimizations. It would be interesting to know whether the actual ratio of mean delays is greater or smaller. A simulation study of WFQ-scheduling may provide some answers. Adaptive algorithms used in the situation where the traffic matrices of the classes are unknown would be useful.

References

[1] A. Viswanathan, N. Feldman, Z. Wang and R. Callon, Evolution of Multi-Protocol Label Swithing, IEEE Communication Magazine, pages 165-173, May 1998.

[2] G. Armitage, MPLS: The Magic Behind the Myths, IEEE Communications Magazine, January 2000.

[3] E. Rosen, A. Viswanathan and R. Callon, Multiprotocol Label Switching Architecture, IETF RFC3031, January 2001.

[4] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell and J. McManus, Requirements for Traffic Engineering over MPLS, IETF RFC 2702, September 1999.

[5] R. Braden, D Clark and S. Shenker, Integrated Services in the Internet Architecture: an Overview, IETF RFC 1633, June 1994.

[6] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, An Architecture for Differentiated Services, IETF RFC 2475, December 1998.

[7] R.G. Gallager, A Minimum Delay Routing Algorithm Using Distributed Computation, IEEE Transactions on Communication, Vol. COM-25, Nr 1, pages 73-85, January 1977.

[8] T. Ott, T. Bogovic, T. Carpenter, K. R. Krishnan and D. Shallcross, Algorithms for Flow Allocation for Multi Protocol Label Switching, Telcordia Technical Memorandum TM-26027, 2001.

[9] A. Sridharan, S. Bhattacharyya, R. Guérin, J. Jetcheva and N. Taft, On The Impact of Aggregation on The Performance of Traffic Aware Routing, Technical report, University of Pennsylvania, July 2000.

[10] J. Castro and N. Nabona, Computational tests of a nonlinear multicommodity network flow code with linear side contraints through primal partitioning, DR 94/05, Statistics ans Operations Research Dept., Universitat Politéctica de Catalunya, Barcelona.

[11] T. Carpenter, K.R. Krishnan and D. Shallcross, Enhancements to Traffic Engineering for Multi Protocol Label Switching, Proceedings ITC17, Brazil, 2001.

[12] http://www.gams.com.