PATH SELECTION AND BANDWIDTH ALLOCATION IN MPLS NETWORKS

James E Burns and Teunis J Ott

Telcordia Technologies Inc, 445 South Street, Morristown NJ 07960-6438, USA

Anthony E Krzesinski and Karen E Müller

Dept. of Computer Science, University of Stellenbosch, 7600 Stellenbosch, South Africa

Abstract

Multi-protocol Label Switching extends the IP destination-based routing protocols to provide new and scalable routing capabilities in connectionless networks using relatively simple packet forwarding mechanisms. MPLS networks carry traffic on virtual connections called label switched paths. This paper considers path selection and bandwidth allocation in MPLS networks in order to optimize the network quality of service. The optimization is based upon the minimization of a non-linear objective function which under light load simplifies to OSPF routing with link metrics equal to the link propagation delays. The behavior under heavy load depends on the choice of certain parameters: It can essentially be made to minimize maximal expected utilization, or to maximize minimal expected weighted slacks (both over all links). Under certain circumstances it can be made to minimize the probability that a link has an instantaneous offered load larger than its transmission capacity. We present a model of an MPLS network and an algorithm which optimally distributes the traffic among a set of active paths and reserves a set of back-up paths for carrying the traffic of failed or congested paths. The algorithm is an improvement of the well-known flow deviation non-linear programming method. The algorithm is applied to compute optimal LSPs for a 100-node network carrying a single traffic class. A link carrying some 1,400 routes fails. The back-up paths are activated and we compare the performance of the path sets before and after the back-up paths are deployed.

Key words: Internet Protocol, label switched path, multi-protocol label switching, quality of service

[1] This work was supported by grants from the Telcordia Research Horizons Pro-

1 Introduction

The Internet is becoming the ideal platform to support all forms of modern communications including voice, data and multimedia transmissions. However, the standard IP routing protocols were developed on the basis of a connectionless model where routing decisions are based on simple metrics such as delay or hop count which leads to the selection of shortest path routes. Despite its ability to scale to very large networks, this approach provides only rudimentary Quality of Service (QoS) capabilities which cannot be used to provide scalable service level agreements for bandwidth intensive applications in modern networks.

Multi-protocol label switching (MPLS) [1] extends the IP destination-based routing protocols to provide new and scalable routing capabilities. MPLS routing/switching is achieved by forwarding IP packets along virtual connections called label switched paths (LSPs). LSPs are set up by a label distribution protocol which uses the information contained in layer 3 routing tables. The LSPs form a logical network that is layered on top of the physical network to provide connection oriented processing above the connectionless IP network.

This paper presents a model of flow optimization in MPLS networks. We address the following questions related to the optimal distribution of traffic flows among LSPs in an MPLS network:

- how is the set of paths computed,
- how is a subset of the computed paths selected to carry the offered traffic,
- how is the traffic distributed among the selected paths, and
- how useful are the remaining unselected paths as backups to be used in the event of traffic overload and/or path failure.

We formulate the problem of finding an optimal set of LSPs and optimally allocating bandwidths to these LSPs as a constrained non-linear programming problem (NLP) which minimizes an appropriate objective function. In qualitative terms the goal is to find a set of LSPs and a set of target bandwidths for these LSPs such that if the traffic forecasts are exact and all target bandwidths of LSPs are achieved, the system will carry all the offered traffic, no link is too heavily utilized, and the carried load is appropriately distributed.

Several previous studies (see [2–4] and the references therein) have formulated the bandwidth allocation problem in connectionless networks as an NLP using the $M/M/1$ formula as a penalty function to predict the queueing delay

gramme, the South African National Research Foundation, Siemens Telecommunications and Telkom SA Limited. The present address of TJ Ott is: Dept. of Computer Science, New Jersey Institute of Technology, Newark NJ 07102, USA.

on individual links, and a load balancing scheme is considered optimal if it minimizes the total delay over the network. However, the delay in an internet is limited by the drain time of buffers. Furthermore, TCP congestion avoidance and RED (Random Early Discard) schemes (see [5,6] and the references therein) make it possible to have a very high sustained utilization on a link with simultaneously only moderate packet loss and only moderate variability in buffer occupation. The use of an $M/M/1$ queueing delay in the penalty function is therefore highly suspect or even incorrect. The issue is not only that the $M/M/1$ formula is poor in predicting actual queueing delay, but that queueing delay is moderately insensitive to traffic intensity on a link. Mechanisms like weighted fair queueing (WFQ) and class-based WFQ will make the queueing delay even more independent of link utilizations.

Our approach to the NLP and its solution has several novel aspects. First, we present a penalty function that affords an appropriate representation of the actual quality of the network. Under light load our penalty function simplifies to OSPF routing with link metrics equal to the link propagation delays. Under heavy load the behavior depends on the choice of certain parameters: It can essentially be made to minimize maximal expected utilization, or to maximize minimal expected weighted slacks (both over all links). Under certain circumstances it can be made to minimize the probability that a link has an instantaneous offered load larger than its transmission capacity.

Second, we present an efficient technique to solve the NLP. We have adapted an existing solution technique, namely the flow deviation method [2–4] to minimize our objective function. Our implementation of the flow deviation algorithm differs from the standard method in that we identify a working set of LSPs and re-distribute bandwidth over these LSPs until it becomes advantageous to admit new LSPs to the working set. Appropriate numerical methods and data structures are used to achieve an efficient implementation of the NLP solver. The advantage of our flow deviation method is that, for the objective function we use, our flow deviation method is several times faster than the standard flow deviation method.

The flow deviation algorithm computes optimal flows in connectionless networks: the objective function is a sum of link penalty functions and the flow is optimally allocated among the links. Recently much attention has been given to adapting optimization algorithms, which were originally developed for circuit switched operations, so that they can be applied to logically fully connected networks which are layered on top of connectionless networks. For example, the capacity routing algorithm (see [7] and the references therein) discovers and capacitates optimal routes in multi-service connection-oriented networks. Here the objective function expresses an end-to-end service measure such as the call blocking probability. Bandwidth is optimally allocated among the discovered routes to form virtual path connections which are either shared among the traffic classes (service integration) or separate VPCs are allocated to each service class (service separation).

The rest of the paper is organized as follows. Sect. 2 presents a model of an MPLS network, definitions of feasible and optimal LSP bandwidth assignments, a description of the LSP design problem whose solution yields an optimal set of LSPs and optimal LSP bandwidth assignments, and a description of a penalty function which under light load simplifies to OSPF routing and depending on certain parameter choices under heavy load optimizes one of a range of performance criteria. Sect. 3 describes our implementation of the flow-deviation algorithm to solve the LSP design problem. Sect. 4 applies the flow deviation method to compute an optimal LSP set for a model of a 100-node network. The characteristics of the LSP set are investigated. Our conclusions are presented in Sect. 5.

2 The Model

Consider a communications network with N nodes and L links. Let $\mathcal{N} = \{1, 2, \ldots, N\}$ denote the set of nodes and let $\mathcal{L} = \{1, 2, \ldots, L\}$ denote the set of links. The nodes represent the routers in the MPLS-capable core of a network. Some nodes are connected by a link. The links are directed: each link has a starting node and an ending node which are routers from the set $\mathcal{N}$.

Each node $m \in \mathcal{N}$ is both an ingress router and an egress router. Each node is an ingress router because traffic from the non MPLS-capable part of the network enters the MPLS network at that point. Each node is an egress router because traffic to the non MPLS-capable part of the network exits from the MPLS network at that point.

Let $d_{(m,n)}$ denote the predicted demand (offered load) of traffic that wants to enter the MPLS network at node m and wants to exit at node n. We assume that the demands $d_{(m,n)}$ and the link capacities b_i are such that a feasible solution exists. The definition of feasibility will be given shortly. If a feasible solution does not exist then systematic drop (discard) of traffic is necessary, and it is an interesting question what traffic needs to be dropped to minimize the damage. We consider only a single class of service.

2.1 Paths and Path Bandwidths

A path P is a sequence of links $L_1, L_2, \ldots, L_{H_P}$ where $H_P \geq 1$ is the *hop count* of the path P. In our terminology a route and a path and an LSP (label switched path) are synonymous. No path traverses the same link or the same node more than once. The algorithms that follow in later sections ensure that no paths contain cycles. Let $\mathcal{P}$ denote the set of all such non-cycling paths. Since any path P contains no cycles, the sequence of links traversed by a path P can be interpreted as a set denoted by $\mathcal{L}_P$. Let $\mathcal{P}^{(i)}$ denote the set of paths that utilize link i. Let $\mathcal{P}_{(m,n)}$ denote the set of paths from node m to node n with $m \neq n$.

2.2 Feasibility and Optimality

Each path P will be assigned a *target bandwidth* $B_P \geq 0$. The goal is to select these target bandwidths in an in some sense optimal way. Let $\boldsymbol{B} = (B_P)_{P \in \mathcal{P}}$ denote a set of target bandwidths. $\boldsymbol{B}$ is said to be feasible if the following two constraints hold:

(1) For each pair of nodes (m, n)

$$\sum_{P \in \mathcal{P}_{(m,n)}} B_P = d_{(m,n)} \tag{1}$$

so that if the traffic forecasts $d_{(m,n)}$ are exact, and if the target bandwidth is achieved for all paths, all of the offered traffic is carried.

(2) For each link i

$$\sum_{P \in \mathcal{P}^{(i)}} B_P \leq b_i \tag{2}$$

so that no link has an offered (target) load greater than its capacity.

We next choose a definition of optimality. Let

$$f_i = \sum_{P \in \mathcal{P}^{(i)}} B_P$$

denote the *target flow* on link i and let $\rho_i = f_i/b_i$ denote the *target utilization* of link i. Let $s_i = b_i - f_i$ denote the *target slack* on link i. Constraint (2) implies that all slacks must be non-negative.

Let $F_i(f_i)$ denote an objective function for link i when the link carries a flow f_i. The *LSP design problem* is specified in terms of the following constrained non-linear optimization problem: Find a set of feasible target bandwidths $\boldsymbol{B}_{\text{opt}}$ that minimizes the objective function

$$F(\boldsymbol{B}) = \sum_i F_i(f_i) \tag{3}$$

subject to the constraints (1) and (2) where the sum in Eq. (3) is over links i with $b_i > 0$. $\boldsymbol{B}_{\text{opt}}$ is said to provide an optimal solution to Eq. (3). Note that the optimal link flows f_i are almost certainly unique although the optimal bandwidths $\boldsymbol{B}$ are usually not: this matter is discussed in Appendix A.

2.3 The Objective Function

The link penalty functions $F_i(x)$ used in the LSP design problem have at least three roles. First, they must to a reasonable degree represent an intuition of what constitutes a "good" load balancing scheme. Second, they must be an efficient way of managing constraints, in particular the constraint that no link carries a load larger than, or even close to, its bandwidth. Third, the link penalty functions must make it possible to efficiently find an optimal solution to the LSP design problem.

The flow deviation algorithm requires that the link penalty functions $F_i(x)$ be increasing and convex on $[0, b_i)$ with $\lim_{x \uparrow b_i} F_i(x) = +\infty$. The latter requirement necessitates a minor change to the definition of feasibility: a solution is said to be feasible if $f_i < b_i$ (strict inequality) on all links i. It is also convenient to make a slightly stronger demand on the functions $F_i(x)$ and require that each $F_i(x)$ be "strongly monotone" on the interval $[0, b_i)$ so that the objective functions are non-negative on $[0, b_i)$ and their derivatives with respect to flow are positive on $(0, b_i)$.

Previous studies of the flow deviation algorithm [2–4] used

$$F_i(x) = M_i \frac{x}{b_i - x} \tag{4}$$

as a link penalty function. If we assume that the offered load to each link i is a Poisson process of packet arrivals and that packets have independent, identically distributed sizes with exponential distribution and average M_i, and that there is an infinite buffer, and that the resulting utilization of the link is x/b_i, then the objective function (4) is the product of the flow x and the average delay (waiting and service both included, but propagation delay excluded). With the $M/M/1$ assumptions above, the sum of the link penalty functions is a measure of the average total network delay.

However, in the modern Internet with TCP, and RED and all its variations, it is possible to have very highly utilized links (utilization practically one) and still low delay and low loss in the buffer: all delay is moved to the edge of the network. The same holds for example for ATM with ABR, in particular the ER version of ABR. Eq. (4) is probably no longer a suitable link penalty function. Given these concerns, we present a link penalty function with properties which make it suitable for use in an objective function whose minimization will yield routes and bandwidths that correspond closely to the optimal operation of a modern internet. Our choice of link penalty function is

$$F_i(x) = c_i x + \eta\sigma_i \left(\frac{\sigma_i}{b_i - x}\right)^\nu \tag{5}$$

where link i has a bandwidth $b_i \geq 0$, a weight factor $\sigma_i > 0$ with $\eta > 0$, $\nu > 1$ and $F_i(x) = \infty$ if $x \geq b_i$. The factor c_i is explained below. The function (5) is strongly monotone and the first derivative of the link penalty function is

$$F_i'(x) = \frac{d}{dx} F_i(x) = c_i + \eta\nu \left(\frac{\sigma_i}{b_i - x}\right)^{\nu+1}. \tag{6}$$

Let $\tau_i \geq 0$ denote the propagation delay on link i. Set $F_i'(0) = \tau_i$. Then $c_i = \tau_i - \eta\nu\left(\sigma_i/b_i\right)^{\nu+1}$. The properties of the link penalty function (5) under light and heavy load are discussed in the following section.

2.4 Behavior under Light and Heavy Load

With reference to the link penalty function (5) we choose η positive but small so that if a feasible solution exists for which all flows f_i are small and all link utilizations f_i/b_i are low – in which case the system is said to be uniformly lightly loaded – then the penalty function (5) will yield routes that are in agreement with OSPF routing where the propagation delays are the OSPF metrics of the links.

If the system is not uniformly lightly loaded then the penalty function enforces a distance from the barrier b_i. The parameter η determines when the barrier begins to dominate the initial linear behaviour of the penalty function. A larger value of η causes the penalty function to rise earlier when the flow approaches the barrier. The parameter ν determines the behaviour of the penalty function as it approaches the barrier. A larger value of ν makes the penalty function steeper when the flow approaches the barrier.

Eq. (6) shows that if ν is large and for some link i the entity

$$\frac{s_i}{\sigma_i} = \frac{b_i - f_i}{\sigma_i}$$

becomes both small in the absolute sense, and also becomes the smallest among all the links, then the dominating objective of the NLP becomes to increase that entity. We call s_i/σ_i the *weighted slack* of link i. Thus if ν is large then the NLP maximizes the minimal weighted slack, at least as long as that minimal

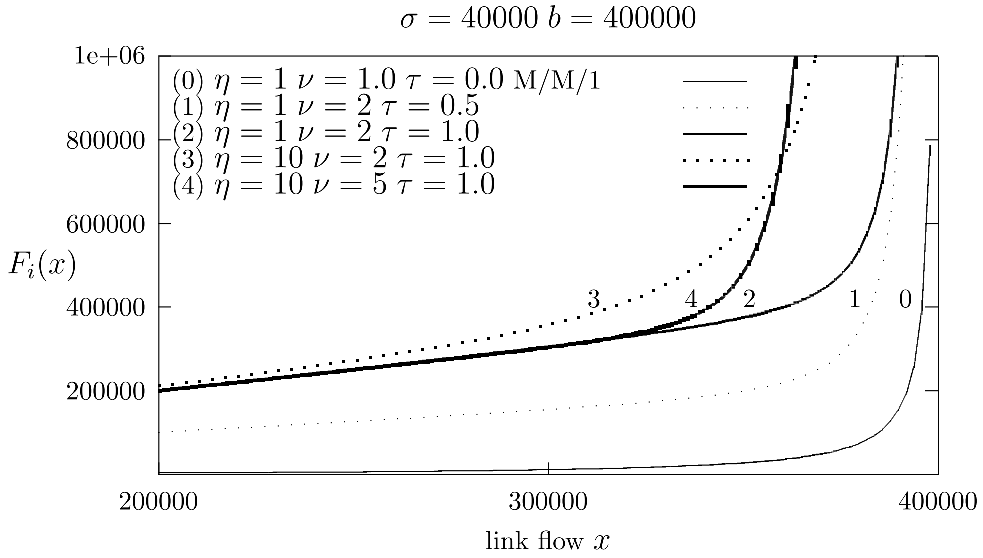

Fig. 1. Examples of the penalty function

weighted slack is small. The magnitude of the minimal weighted slack depends on ν. Even better: as long as ν is sufficiently large, the NLP attempts to perform a "lexicographic maximization" of all small weighted slacks: first it maximizes the smallest weighted slack, then the next smallest, and so on.

The choice of σ_i is of interest. For example, if we choose $\sigma_i = b_i$ then we minimize the maximal utilization, as long as that maximal utilization is large. An interesting situation also arises when all b_i (insofar positive) are large. In that case we can choose for σ_i an estimate of the standard deviation of the instantaneous offered load to link i in the situation where the target load is somewhat close to b_i. In that case the weighted slack is the "distance" from the target flow f_i to the bandwidth b_i, measured in units of standard deviations. Assuming a Central Limit Theorem, and assuming that the distance as defined above is at least several standard deviations, then by maximizing the minimal weighted slack we are also essentially minimizing the maximal probability that the offered load to a link is larger than its bandwidth.

Fig. 1 plots the penalty function (5) of a link i as a function of the link flow x. The link bandwidth $b_i = 400,000$ and the weight $\sigma_i = b_i/10 = 40,000$. These values are related to the parameters of a 50-node network model [8] where the average link capacity is 190,689 ±81,026 and the average flow carried on a link is 95,265 ±48,414.

With reference to Fig. 1 plot (0) shows the $M/M/1$ penalty function using related parameters. Plots (1) through (4) are for the penalty function (5). Plot (1) shows the effect of $\tau_i = 0.5$, $\eta = 1$ and $\nu = 2$. Plot (2) shows the effect of increasing τ_i from 0.5 to 1.0. The parameter η determines when the barrier b_i begins to dominate the initial linear behaviour of the penalty function. Plot

(3) shows that the penalty function begins to rise towards the barrier earlier when η is increased from 1 to 10. The parameter ν determines the behaviour of the penalty function as it approaches the barrier b_i: increasing ν increases the steepness of the rise. Plot (4) shows the effect of increasing ν from 2 to 5.

2.5 A Flow Optimization Model

Path identification and traffic distribution are the two most important processes involved in flow optimization. A flow optimization model may be based on either a reactive scheme where path identification and traffic distribution are computed simultaneously to achieve an optimal traffic flow allocation, or a pre-planned model where path identification and traffic distribution are performed separately.

We present a flow optimization model which consists of three parts:

Path identification. Path sets are computed for each source-destination pair using a reactive method.

Path selection. The path set is partitioned to yield a set of active paths to carry the offered traffic and a set of back-up paths for fault recovery or congestion avoidance.

Traffic distribution. The flow deviation model is used to move the offered traffic from higher cost paths to lower cost paths.

See [9] for a description of a flow optimization model based on a pre-planned path discovery model.

3 The Flow Deviation Algorithm

This section presents an implementation of the flow deviation algorithm [2–4] which minimizes a convex objective function and thus converges to a global optimum.

The algorithm executes in a loop where each iteration of the loop implements one step of the algorithm. During each step the algorithm computes the current set of shortest (least cost) paths from all sources to all destinations. An optimal amount of flow is diverted from the current set of LSPs to the shortest paths. Those shortest paths that are not already in the LSP set are added to the LSP set, the link costs are updated (the link costs have changed because the link flows have changed) and the next step of the algorithm is executed. The loop continues until flow re-distribution achieves no further reduction in the objective function. A small worked example of the operation of the flow deviation algorithm can be found in [3].

3.1 The Algorithm

In the MPLS context the flow deviation algorithm incrementally improves the set $\mathcal{P}$ of LSPs and improves the distribution of traffic over multiple paths in $\mathcal{P}$ from the same source to the same destination. Improving $\mathcal{P}$ mainly consists of adding paths that have, or are likely to have, lower cost than the existing paths from the same source to the same destination. Improving $\mathcal{P}$ may involve discarding paths P that are known not to have positive B_P in any optimal solution, or are not likely to have such a positive flow. Discarding non-promising paths is not necessary for convergence but significantly decreases the computational effort.

The algorithm executes in a loop. Each iteration of the loop implements one step which is identified by a step index k.

(1) Initialize: Set $k = 0$. For each link i set the link flow $f_i = 0$. Compute the least cost path $P = P_e(m, n)$ connecting each node pair (m, n). Set the target bandwidth $B_P = d_{(m,n)}$. If necessary call statement (6) to enforce a feasible solution. Initialize the path set $\mathcal{P} = \cup_{(m,n)} P_e(m, n)$.

Statements (2) through (8) below constitute the body of the loop.

(2) For each link i compute the link cost $C_i^L = F_i'(f_i)$. For each path P compute the route cost $C_P^R = \sum_{i \in \mathcal{L}_P} C_i^L$.
(3) Compute a feasible direction $\mathbf{\Delta} = (\Delta_P)_{P \in \mathcal{P}}$ and an improved path set $\mathcal{P}$. The calculation of an improved path set and a feasible direction is discussed in Sect. 3.2.
(4) Convergence test: If no feasible direction can be found then optimality has been achieved and the algorithm halts. This stopping rule is theoretically correct but of no practical value. A practical stopping rule is discussed in Sect. 3.4.
(5) Compute improved path flows $\boldsymbol{B}$: Compute a value of x such that $B_P := B_P + x\Delta_P$ yields a value $F(\boldsymbol{B})$ of the objective function which is a strict improvement over the value of the objective function computed in the previous step, and in the direction $\mathbf{\Delta}$ is optimal. This computation is called the "line search" for x. The calculation of x is discussed in Sect. 3.3. In qualitative terms: a very small positive x value always gives an improvement. We increase x either until the objective function stops decreasing, or until a path flow B_P goes to zero in which case the path P leaves the set $\mathcal{P}$.
(6) Enforce a feasible solution: If the target bandwidths $\boldsymbol{B}$ are not feasible

then for each link i set $b_i := \alpha b_i$ where $\alpha = \max_i(1.05 f_i / b_i)$. The solution $\boldsymbol{B}$ is now feasible. [2]

(7) Compute improved link flows: For each link i compute $f_i := f_i + x\delta_i$ where $\delta_i := \sum_{P \in \mathcal{P}^{(i)}} \Delta_P$.

(8) Loop statement: $k := k + 1$ and go to statement 2.

3.2 Choosing a Feasible Direction

A feasible direction is a map $\boldsymbol{\Delta} = (\Delta_P)_{P \in \mathcal{P}}$ with the following properties:

- the traffic demand $d_{(m,n)}$ offered to each node pair (m, n) is constant therefore $\sum_{P \in \mathcal{P}_{(m,n)}} \Delta_P = 0$,
- an empty path cannot have its bandwidth allocation lowered so that if $B_P = 0$ then $\Delta_P \geq 0$,
- a feasible direction will lower the network cost so that $\sum_{P \in \mathcal{P}} \Delta_P C_P^R < 0$.

We present two methods for computing a feasible direction. The first method, the so-called global method, may add paths to the set $\mathcal{P}$. The second method, the so-called local method, does not add paths to the set $\mathcal{P}$: in fact it is likely to remove paths from $\mathcal{P}$.

3.2.1 The Global Method

Given a feasible solution $\boldsymbol{B}$ and the current link costs C_i^L, compute the shortest path $P_e(m, n)$ connecting each source-destination (S-D) pair (m, n). There may be several such paths in which case a tie-breaking mechanism is needed. This path may already be in the set of known paths $\mathcal{P}_{(m,n)}$ and have a positive flow $B_{P_e(m,n)} > 0$. If the path is not in $\mathcal{P}_{(m,n)}$ then it is added to $\mathcal{P}_{(m,n)}$. For each $P \in \mathcal{P}_{(m,n)}$ compute

$$\Delta_P = \begin{cases} -B_P & P \in \mathcal{P}_{(m,n)} \setminus P_e(m,n) \\ d_{(m,n)} - B_P & P = P_e(m,n). \end{cases}$$

3.2.2 The Local Method

Given a feasible solution $\boldsymbol{B}$ and the current link costs C_i^L and the route costs C_P^R, choose a subset $\mathcal{R}_{(m,n)}$ from $\mathcal{P}_{(m,n)}$ for each S-D pair (m, n) as follows: all routes $P \in \mathcal{P}_{(m,n)}$ with $B_P > 0$ are in $\mathcal{R}_{(m,n)}$; optionally some or all routes

[2] When the flow deviation algorithm terminates then $\alpha = 1$ else the solution $\boldsymbol{B}$ is not feasible.

$P \in \mathcal{P}_{(m,n)}$ that have the minimal value of C_P^R for all $P \in \mathcal{P}_{(m,n)}$ may be included, even those with $B_P = 0$; no other paths are included in $\mathcal{R}_{(m,n)}$.

Let $\mathcal{R} = \cup_{(m,n)} \mathcal{R}_{(m,n)}$ denote the set of *active* paths. $\mathcal{R}$ includes the paths P that have $B_P > 0$ and in addition $\mathcal{R}$ may contain some paths P that, because of their low current cost compared with other active paths from the same source to the same destination, are likely to be assigned a positive B_P.

Let $R_{(m,n)} = |\mathcal{R}_{(m,n)}|$ denote the number of active paths that connect node m to node n. Let

$$C^S_{(m,n)} = \sum_{P \in \mathcal{R}_{(m,n)}} C_P^R.$$

For each $P \in \mathcal{R}_{(m,n)}$ compute

$$\Delta_P = C^S_{(m,n)} - R_{(m,n)} C_P^R. \tag{7}$$

Thus if there are for example two routes from node m to node n then Eq. (7) will decrease the flow on the more costly route and increase the flow on the cheaper route (at the same rate), and the rates of change are proportional to the difference in route costs. The ideal situation would be that where all node-pairs (m,n) with two routes will reach their cross-over point where costs become equal at about the same time (for about the same value of x).

Reducing the size of the path set $\mathcal{P}$ substantially improves the performance of the flow deviation algorithm. The next section describes a method to quickly remove many paths from $\mathcal{P}$ that are unlikely to belong to the final optimal set of paths.

The Local Method: Removing Inferior Paths

Given a feasible direction $\boldsymbol{\Delta}$, compute for each S-D pair (m,n)

$$x^R_{max}(m,n) = \min_{P \in \mathcal{R}_{(m,n)}: \Delta_P < 0} \quad \frac{B_P}{|\Delta_P|} \tag{8}$$

where $x^R_{max}(m,n) = +\infty$ if $\Delta_P = 0$ for all $P \in \mathcal{R}_{(m,n)}$. In the line search, if x grows to $x^R_{max}(m,n)$ then the flows on one or more of the routes in $\mathcal{R}_{(m,n)}$ will decrease to zero, and that route would be expelled from $\mathcal{P}$. This mechanism with high probability expels at most one route per iteration. We have a mechanism to improve this:

Choose a parameter $\widehat{x} \geq 0$. For those S-D pairs (m,n) with $x^R_{max}(m,n) < \widehat{x}$

we re-scale

$$\Delta_P := \Delta_P \frac{x^R_{max}(m,n)}{\widehat{x}} \tag{9}$$

for all $P \in \mathcal{R}_{(m,n)}$. Compute

$$x^R_{max} = \min_{(m,n)} x^R_{max}(m,n)$$

using the values of $x^R_{max}(m,n)$ computed in Eq. (8) with the re-scaled $\mathbf{\Delta}$ values as computed in Eq. (9). Now $x^R_{max} \geq \widehat{x}$ and if Eq. (9) is applied at least once then $x^R_{max} = \widehat{x}$. The result now is that no path in $\mathcal{R}$ loses all its flow until x increases to x^R_{max}, and for $x = x^R_{max}$ a potentially large number of paths all lose all their flow.

We proceed as follows: initialize $\widehat{x} = 0$ so no re-scaling occurs after Eq. (7) has computed a feasible direction $\mathbf{\Delta}$. If Eqs. (11) and (12) below determine $x = x_{max} = x^R_{max}$, (i.e. the optimal x is one that causes elimination of at least one route), calculate

$$\widehat{x} = 2x^R_{max}$$

for use in the next iteration of the flow deviation algorithm. Else (if the optimal value of x does not cause elimination of any route) we set $\widehat{x} = 0$ for the next iteration of the flow deviation algorithm. Once the correct set of paths has been found, $\widehat{x}$ is likely to remain at zero.

3.2.3 A Mixed Method

It is likely that in the optimal solution each S-D pair (m,n) will be connected by a small number of paths $P \in \mathcal{P}_{(m,n)}$ and these paths will all have the same route costs. It is to be expected that after a while the shortest path algorithm will keep returning paths from that small set. Once the algorithm is in this situation the use of the local method seems preferable.

The local and global methods can be combined as follows. The flow deviation algorithm initially iterates using the global method until the shortest path algorithm finds no new paths.

The flow deviation then alternates between the local and the global methods as follows: an iteration of the global method is followed by k iterations of the local method ($k \geq 0$) and then another iteration of the global method. If that new iteration of the global method finds a new path, it is followed by zero iterations of the local method. Otherwise, it is followed by $k+1$ iterations of the local method.

The algorithm thus alternates between the global and local methods until the stopping rule in Sect. 3.4 below is triggered.

3.3 The Line Search

In this section we compute a value of x which yields an improved solution

$$B_P(x) = B_P + x\Delta_P \tag{10}$$

for all $P \in \mathcal{P}$. Define

$$x^L_{max} = \min_{i:\delta_i>0} \quad \frac{s_i}{\delta_i}$$

where $x^L_{max} = +\infty$ if $\delta_i \leq 0$ for all i. If x grows to $x^L_{max} < \infty$ then the slack on one or more links will equal zero. Thus we have the constraint $x < x^L_{max}$. Next define

$$x^R_{max} = \min_{P:\Delta_P<0} \quad \frac{B_P}{|\Delta_P|}.$$

It is impossible that $\Delta_P \geq 0$ for all P. If x grows to x^R_{max} then the flows on one or more routes in $\mathcal{P}$ will decrease to zero. Thus we have the constraint $x \leq x^R_{max}$.

Set $x_{max} = \min(x^L_{max}, x^R_{max})$. With an abuse of notation, we wish to find a value of $x \in [0, x_{max}]$ which minimizes

$$F(x) = \sum_i F_i(f_i + x\delta_i).$$

Setting $y = f_i + x\delta_i$ and taking derivatives we obtain

$$F^{(k)}(x) = \left(\frac{d}{dx}\right)^{(k)} F(x) = \sum_i (\delta_i)^k \left(\frac{d}{dy}\right)^{(k)} F_i(y).$$

Since the functions $F_i(\cdot)$ are strongly monotone, even derivatives of $F(x)$ are positive and odd derivatives of $F(x)$ are strictly increasing. If

$$x^R_{max} < x^L_{max} \tag{11}$$

then $x_{max} = x^R_{max}$. In that case, compute

$$F'(x_{max}) = F^{(1)}(x_{max}) \tag{12}$$

If Eq. (11) holds and $F'(x_{max}) \leq 0$ then x_{max} is the optimal value for x. In this case, in updating the feasible solution, one or more routes have their flow reduced to zero and these routes may be removed from $\mathcal{P}$.

If Eq. (11) does not hold, or if it holds but $F'(x_{max}) > 0$ then we need to find the value of $x \in [0, x_{max})$ where $F'(x) = 0$. Because the even derivatives of $F(x)$ are positive we can use the Newton-Raphson method [10] to find the value of x.

3.4 The Stopping Rule

The algorithm requires a stopping rule to determine the iteration k when the algorithm has converged. We can stop when either $|F'(x_{k+1}) - F'(x_k)|$ or $|x_{k+1} - x_k|$ has been close to zero for some time in which case further iterations will yield no improvement in the solution. We can use a combination of these two criteria. Because even derivatives of $F(\cdot)$ are positive, the sequence (x_k) will become monotone decreasing or increasing. It may be safe not to stop until the sequence has been monotone for some time and one or both of the other conditions above is satisfied.

3.5 Comparing the Local and Global Methods

The global method. The global method computes the shortest paths between all S-D pairs each time a new feasible direction is calculated. The shortest path calculation has complexity $\mathrm{O}(N^3)$ where N is the number of nodes in the network. After each shortest path calculation the global method needs to check if the shortest paths are already in the set $\mathcal{P}$.

The choice of direction can lead to convergence problems. For many S-D pairs there are, in the optimal solution, several paths with equal costs and each path carries a significant flow. When close to the optimal solution, one of these paths will have the least cost. The global method will move flow from the slightly more costly path to the slightly less costly paths. With high likelihood, in the next iteration the previous slightly more costly path has become the slightly less costly, and the direction of the transfer of flow is reversed: the algorithm oscillates.

The local method. The local method for choosing a feasible direction has computational complexity $\mathrm{O}(R_{(m,n)})$ per S-D pair (m, n).

Paths which are likely not to carry a flow in an optimal solution are removed from the set of active flows by driving their flows to zero. If we had not used the refinement introduced in Sect. 3.2.2 then with near certainty the local method will remove one path per iteration or worse: not every iteration need eliminate one such flow. If there were a large number of such flows this could, in the absence of the above refinement, cause a major slow-down in the algorithm.

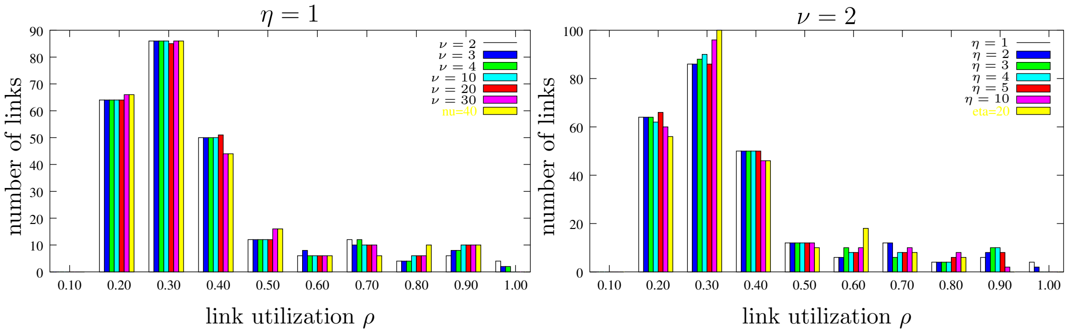

Fig. 2. Distribution of the link utilization ρ

3.6 Implementation Issues

Each time the global method is invoked it calculates the shortest paths connecting all S-D pairs and checks whether the current set of shortest paths is already in $\mathcal{P}$. These calculations are computationally expensive. The shortest paths are computed using Floyd's algorithm – see [11] and the references therein for a discussion of the relative merits of several well-known shortest path algorithms. Each path P is stored in a table which is accessed via a hash index computed over the link set $\mathcal{L}_P$.

4 An Application

This section presents a numerical study of MPLS path selection in a model of a 100-node network with 244 uni-directional links and 1 traffic class. The links are capacitated with 6,515,881 units of bandwidth. A total of 250,000 units of flow are offered to the 9,900 S-D pairs. A description of the model with link capacities and offered traffics can be found at the URL http://www.cs.sun.ac.za/projects/COE/models.zip.

4.1 The Choice of Penalty Function Parameters

Fig. 2 compares the link utilization distributions computed by the global flow deviation method for several values of the penalty function (5) parameters η and ν. Given that the link utilization distribution (for the 100-node model) is relatively insensitive to the values of η and ν, the results presented in the rest of this section are for $\eta = 1$, $\nu = 2$, $\tau_i = 1$ and $\sigma_i = b_i/10$.

4.2 The Kleinrock versus the Bertsekas-Gallager Methods of Flow Deviation

Two variants of the standard flow deviation algorithm have appeared in the literature. Kleinrock's implementation [4] uses a line search to compute the optimal amount of flow to move, and flow is moved for all S-D pairs at once. Bertsekas and Gallager [2] developed a variant of the flow deviation algorithm which avoids a computationally expensive line search and instead estimates the amount of flow to be moved – the flow is moved for one S-D pair at a time. The main advantage of the Bertsekas-Gallager algorithm is that it is computationally less expensive than Kleinrock's algorithm and it computes better values for small moves near the optimal point where Kleinrock's algorithm can oscillate. Kershenbaum [3] developed a technique for scaling the link capacities to enforce feasible solutions: we use this technique since it is effective and simpler than the search for feasible solutions presented in [4].

Fig. 3 investigates the commonality among the active path sets found by the Kleinrock (K) and the Bertsekas-Gallager (B-G) flow deviation algorithms. Consider a path P that is present in both the K and the B-G path sets. Let F_P and F'_P denote the flow on path P when path set is computed using the K and the B-G flow deviation algorithms respectively. If $|F_P - F'_P| < 0.05$ then the path P is said to be in *strong agreement* among the K and B-G path sets, else the path P is said to be in *weak agreement*. Fig. 3 shows that 73% of the paths are in strong agreement: the flows on these paths are the same to within 5% in the K and B-G solutions. 12% of the paths are in weak agreement. Kleinrock's algorithm finds 1,800 paths that do not occur in the B-G path set, but these paths carry a trivial flow. Likewise 6 paths in the B-G path set do not occur in the K path set, but these paths also carry a trivial flow. The additional routes with minimal flow discovered by Kleinrock's method can be useful as back-up routes.

We conclude that, for the 100-node model, the K and B-G flow deviation algorithms discover equivalent path sets once the trivial routes are discarded from the K path set. In the remainder of this section will therefore use the Bertsekas-Gallager algorithm as the basis for our global and mixed flow deviation algorithms. Note again that the optimal paths sets are usually not unique: this matter is discussed in Appendix A.

4.3 The Effect of the Penalty Function

We next investigate whether the global flow deviation algorithm using the penalty function (5) succeeds in computing a flow distribution which maximizes the minimum slacks of the network links. The calculation of a set of

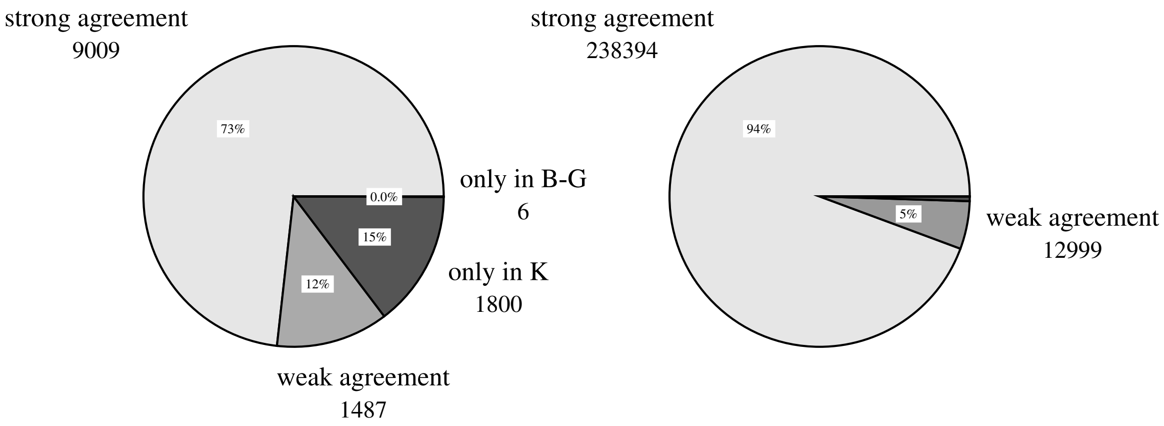

Fig. 3. Commonality among the (a) paths and (b) path flows computed by the Kleinrock and the Bertsekas-Gallager flow deviation algorithms

link flows which maximizes the minimum slacks was formulated as a linear programming (LP) problem and the optimal routes and route flows were extracted from the optimal link flows [12].

Fig. 4 investigates the commonality among the active path sets found by the flow deviation (FD) and the LP methods. 51% of the paths are in strong agreement: the flows on these paths are the same to within 5% in the FD and LP solutions. 18% of the paths are in weak agreement. 25% of the FD paths do not occur in the LP path set, but these paths carry only 7% of the flow in the FD model. 6% of the LP paths do not occur in the FD path set, but these paths carry only 4% of the flow in the LP model. 89 % of the flow is distributed among paths that are present in both the LP and the FD path sets, and 80% of the flow is carried on paths that are in strong agreement.

Fig. 5(a) compares the link utilization distributions computed by the LP and FD methods. The two distributions are in good agreement though the LP method yields a few more under- and over-utilized links.

From our comparison of the LP and FD link and path flows we conclude that for the 100-node model and for the parameter values being used, the LP and FD path sets are largely equivalent: the flow deviation algorithm using the penalty function (5) has succeeded in computing a flow distribution which maximizes the minimum slacks of the network links.

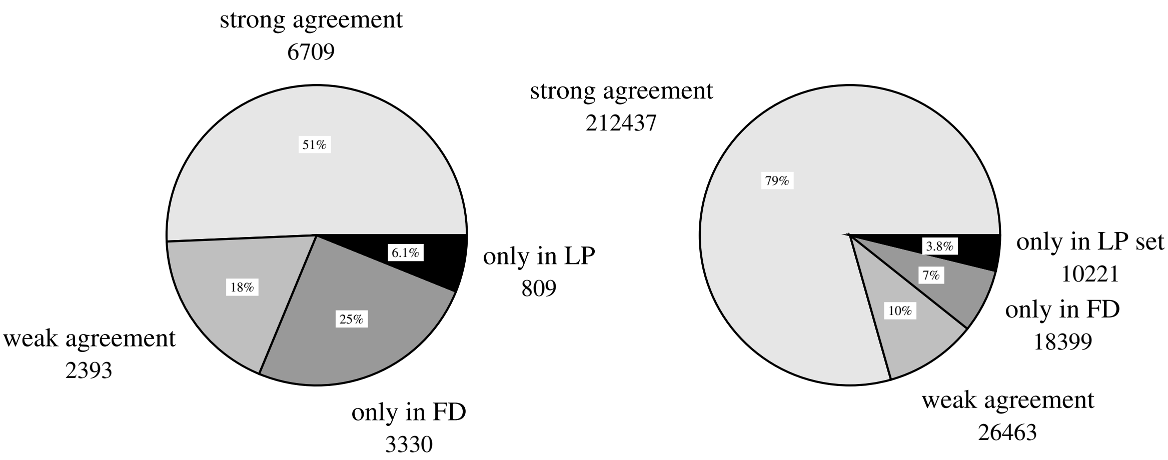

Fig. 4. Commonality among the (a) paths and (b) path flows computed by the FD and LP methods

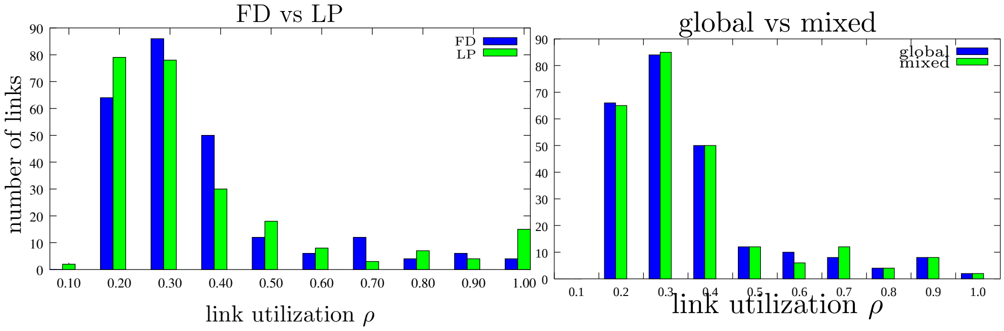

Fig. 5. Distribution of the link utilization ρ computed by (a) the global FD and the LP methods (b) the global and the mixed FD methods

4.4 The Global versus Mixed Flow Deviation Methods

We next compare the qualities of the active path sets as computed by the global and the mixed flow deviation methods. The global method requires some 20 seconds of CPU time on a Pentium III 700 MHz processor to solve the LSP design problem for the 100-node model; the mixed method requires some 10 seconds. Fig. 5(b) shows that the global and mixed methods yield nearly the same link utilization distributions.

In the remainder of this section, the length of a path denotes the hop count of a path. The *normalized length* of a path is the length of that path minus the length of the shortest path connecting the S-D pair of that path.

path multiplicity	S-D pairs	S-D routes	flow	%	flow/ route	flow/ S-D
global method						
1	9307	9307	237690	95.0	25.5	25.5
2	584	1168	12193	4.8	10.4	20.8
3	9	27	103	0.0	3.8	11.5
mixed method						
1	9141	9141	236466	94.5	25.8	25.8
2	738	1476	13457	5.3	9.1	18.2
3	20	60	220	0.0	3.6	11.0
4	1	4	8	0.0	2.0	8.3

Table 1
Path multiplicity

The global method finds an optimal LSP set containing 10,502 routes. The average normalized route length is 0.39 and the average LSP bandwidth is 23.8. The mixed method finds an optimal LSP set containing 10,681 routes. The average normalized route length is 0.38 and the average LSP bandwidth is 23.4. The LSP sets have several attractive features. The LSPs overwhelmingly coincide with the shortest routes connecting the S-D pairs. Most S-D pairs are connected by one or two LSPs. Some 95% of the flow is assigned to the shortest LSPs.

Tab. 1 shows the flow assigned to n-path connections where $n = 1, 2, 3, 4$ (a S-D pair is said to have an n-path connection or a path multiplicity of n if the pair is connected by n LSPs.) For example the second row of Tab. 1 shows that 584 S-D pairs are connected by two routes: the 1,168 routes carry 12,193 units of flow which is 4.8% of the total flow carried by the network. Each of these routes carries on average 10.4 units of flow. Each two-path connection carries on average 20.8 units of flow. The global method yields an average path multiplicity of 1.06. The mixed method yields an average path multiplicity of 1.08.

Fig. 6 investigates the commonality among the active path sets found by the global and the mixed flow deviation methods. Fig. 6 shows that 84% of the paths are in strong agreement: the flows on these paths are the same to within 5% in the global and the mixed solutions. 9% of the paths are in weak agreement. 3% of the global paths do not occur in the mixed path set, but these paths carry a trivial flow. 4% of the mixed paths do not occur in the global path set, but these paths carry a trivial flow. 99 % of the flow is distributed among paths that are present in both the global and the mixed path sets, and 95% of the flow is carried on paths that are in strong agreement.

Tab. 1, Fig. 5(b) and Fig. 6 confirm that the LSP sets computed by the global and mixed flow deviation methods are nearly equivalent. Given the good agreement between the solutions for the 100-node model as computed by

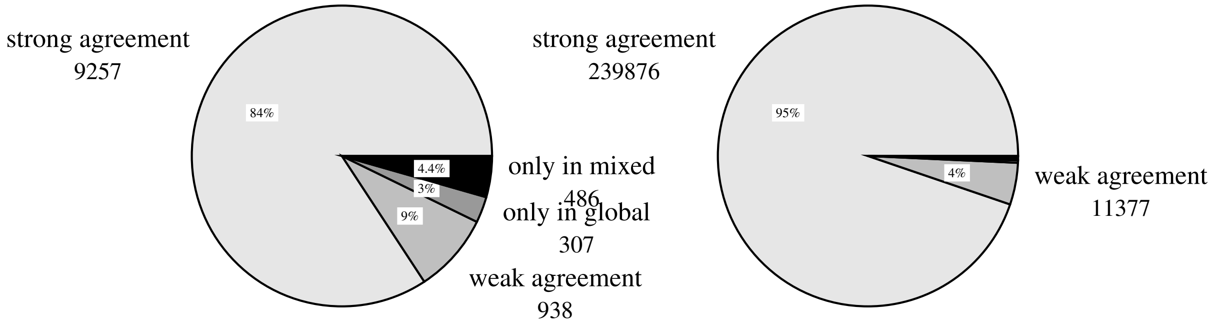

Fig. 6. Commonality among the (a) paths and (b) flows computed by the global and the mixed flow deviation methods

the global and mixed methods, the results presented in the rest of this section are computed by the global method since this method, though less efficient than the mixed method, computes optimal as opposed to near-optimal path sets.

4.5 Back-up Paths

The fault-tolerance capability of a routing scheme increases with the number of paths available for re-routing and with the quality of these paths. Consider an S-D pair connected by several paths. These paths are said to be pair-wise link disjoint if they have no links in common. Link disjoint paths afford better fault-tolerance since they do not share a common point of failure.

The flow deviation algorithm computes an active path set for the 100-node model consisting of 10,502 routes carrying 249,987 units of flow. The algorithm identified 8,609 routes to which flow was not assigned: these routes are designated as back-up paths. These paths provide back-up for 50% of the S-D pairs which offer 33% of the traffic to the network. 12% of the back-up paths are pair-wise link disjoint.

A back-up path should be provided for each S-D pair. In [9] we describe the K shortest path (KSP) method for computing a set of link-disjoint paths connecting each S-D pair. A set $\mathcal{Q}$ of back-up paths that covers all the S-D pairs is found by taking the union of the set $\mathcal{P}_{\text{back-up}}$ of back-up paths discovered by the flow deviation algorithm and the set $\mathcal{P}_{\text{KSP}}$ of KSP paths with the set $\mathcal{P}_{\text{active}}$ of active paths removed. Thus

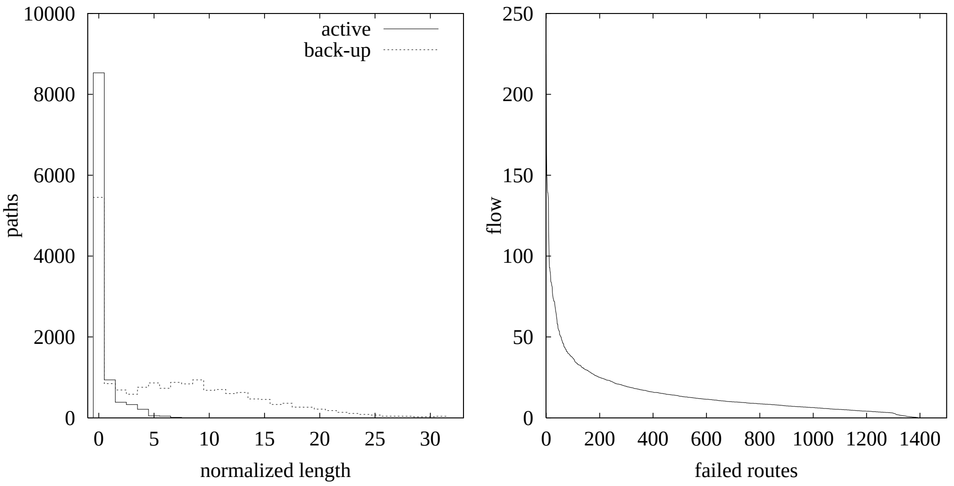

Fig. 7. (a) Normalized lengths of active and back-up paths (b) flows on failed routes

$$\mathcal{Q} = \mathcal{P}_{\text{back}-\text{up}} \cup (\mathcal{P}_{\text{KSP}} \setminus \mathcal{P}_{\text{active}}).$$

The combined path sets yield a total of $|\mathcal{Q}| = 18,349$ back-up paths.

Fig. 7(a) presents the distributions of the normalized lengths of the active paths and the back-up paths. Most of the active paths are of normalized length 0 and are thus the shortest paths between their respective S-D pairs. This implies that the active paths make efficient use of the link bandwidth. Although many of the back-up paths have a normalized length 0, some of the back-up paths have a large normalized length and do not make efficient use of the link bandwidth.

The most active link – though not the most heavily utilized link – in the 100-node network carries 1,393 routes (13% of the network routes) and 21,228 units of flow (8% of the network flow). The failure of this link is modelled by setting the cost of this link to a very large positive number. Fig. 7(b) presents the distribution of the flow on the failed routes. The flow deviation algorithm is then executed with the back-up path set $\mathcal{Q}$ as input. The flow deviation algorithm can assign flow to the back-up paths. However, the algorithm may determine that some back-up paths are not optimal and in this case the algorithm will find and use new paths.

Fig. 8 compares the performance of the path sets before and after the link failed. Fig. 8(a) shows that 61% of the routes that were present before the link failed remain in use after the link failed. 2,361 routes were dropped: the dropped routes include the 1,393 failed routes that passed through the failed link as well as other (discarded) routes that are no longer used once the flow was optimally diverted to avoid the failed link. 2,722 new routes were used to carry the diverted flow: 1,296 of these routes are back-up routes and 1,453 are

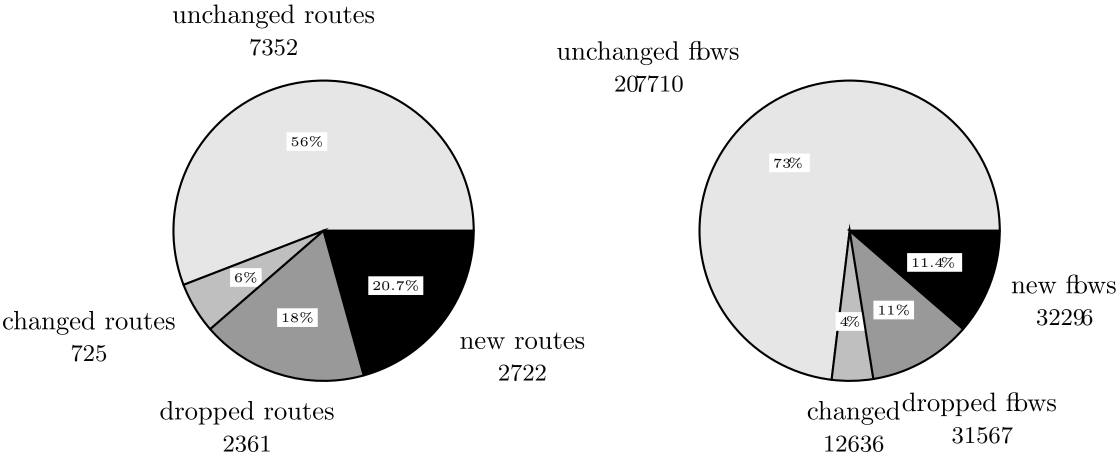

Fig. 8. Commonality among the (a) paths and (b) flows after a heavily utilized link has failed and the back-up paths are deployed

new routes. Many of the back-up routes were therefore not used.

Fig. 8(b) shows that 73% of the route flows are in strong agreement before and after the link failed: the flows on these routes are, to within 5%, undisturbed by the link failure. 4% of the routes are in weak agreement: their flows have changed by more than 5%. 31,567 units of flow were moved from the dropped routes: 21,228 units of flow were moved from the failed routes and 10,339 from the discarded routes. 32,296 units of flow were moved to back-up paths and to newly discovered paths. Of this amount, 17,400 units of flow were assigned to the back-up routes and 14,896 units of flow were assigned to new routes.

5 Conclusion

This paper considers the problem of optimal path selection in MPLS networks. The problem is formulated as the minimization of a non-linear objective function which under light load simplifies to OSPF routing with link metrics equal to the link propagation delays, and under heavy load minimizes the probability that a transmission link has an instantaneous offered load larger than its bandwidth. We present an efficient algorithm based on the flow deviation method to find the optimal paths and to assign optimal bandwidths to these paths. The algorithm also discovers a set of back-up paths for carrying the traffic of failed or congested paths. The algorithm is applied to compute optimal LSPs for a 100-node network carrying a single traffic class. We show that the flow deviation algorithm, using the given objective function, computes a flow distribution that is consistent with the goal of maximizing the minimum slacks on the network links. We investigate several variants of the flow devi-

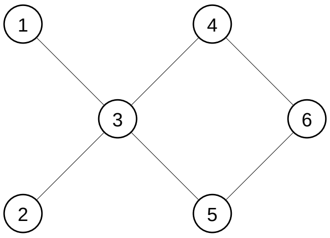

Fig. A.1. The “Fish” network

ation algorithm and show that they compute near identical flows. Finally we investigate the utility of the back-up paths. A heavily utilized link carrying some 1,400 routes fails: the back-up paths are activated and we compare the performance of the path sets before and after the back-up paths are deployed.

A Path Sets and Route Degeneracy

The optimal solution $\boldsymbol{B}$ computed by the flow deviation algorithm is not unique. For example consider the network [1] presented in Fig. A.1 where traffic is offered from nodes 1 and 2 to node 6: the traffic demands are $d_{(1,6)} = 0.5$ and $d_{(2,6)} = 1.5$. All links have capacity $b_i = 2$ and have the same propagation delay and the same weight factor. The optimal link flows are

$$\begin{array}{ll} f_{(1,3)} = 0.5 & f_{(2,3)} = 1.5 \\ f_{(3,4)} = 1.0 & f_{(3,5)} = 1.0 \\ f_{(4,6)} = 1.0 & f_{(5,6)} = 1.0 \end{array}$$

Let f_P denote the flow on path P. We can assign any flow z where $0 \leq x \leq 0.5$ to path $(1,3,4,6)$ whereupon the flows assigned to other routes are

$$\begin{array}{ll} f_{(1,3,4,6)} = z & f_{(1,3,5,6)} = 0.5 - z \\ f_{(2,3,4,6)} = 1.0 - z & f_{(2,3,5,6)} = 0.5 + z. \end{array}$$

It is probably an advantage for a S-D pair to have two paths rather than one path. Having four paths rather than three is probably a disadvantage. Operational requirements may prefer a particular value of z. Thus $z = 0$ and $z = 0.5$ will reduce the number of paths from four to three. The flow deviation algorithm yields $z = 0.25$ which assigns two paths from each of nodes 1 and 2 to node 6 with equal bandwidth. From the point of view of robustness under traffic forecast error, this may be the preferred solution.

Given the link flows, we need methods to compute not only a set of paths and a set of path flows consistent with the link flows, but we also need criteria to determine which set of paths and path flows are superior, and we need mechanisms to find optimal (according to those criteria) path sets and path flows.

References

[1] Consult http://www.ietf.org for MPLS RFC and Draft Documents such as RFC 3031 (MPLS Architecture), RFC 3036 (LDP Specifications), RFC 2702 (Requirements for Traffic Engineering over MPLS).

[2] D. Bertsekas and R. Gallager, *Data Networks Second Edition*, Prentice-Hall International Inc., 1992.

[3] A. Kershenbaum, *Telecommunication Design Algorithms*, McGraw-Hill, 1993.

[4] L. Kleinrock, *Queueing System Vol. 2: Computer Applications*, John Wiley & Sons, New York, 1976.

[5] S. Floyd and V. Jacobson, Random early detection gateways for congestion avoidance, *IEEE/ACM Transactions on Networking*, 1993.

[6] T.J. Ott, T.V. Lakshman and L.H. Wong, SRED: Stabilized RED, *Proceedings of IEEE INFOCOM'99*, pp. 1346 – 1355, 1999.

[7] Traffic Engineering and QoS Methods for IP-, ATM- and TDM-Based Multiservice Networks, draft-ietf-tewg-qos-routing-04.txt. Obtainable from http://www.ietf.org/internet-drafts/draft-ietf-tewg-qos-routing-04.txt

[8] C. Villamizar, http://brookfield.ans.net/omp/random-test-cases.html.

[9] A.B. Bagula and A.E. Krzesinski. Traffic Engineering label Switched Paths in IP Networks using a Pre-planned Flow Optimization Model. *Proceedings 9th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'2001)* (August 2001) Cincinnati, USA, pp 70–77.

[10] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery, *Numerical Recipes in C Second Edition*, Cambridge University Press, 1992.

[11] R. Bhandari, *Survivable Networks: Algorithms for Diverse Routing*, Kluwer Academic Publishers.

[12] T. Carpenter, K.R. Krishnan and D. Shallcross. Enhancements to Traffic Engineering for Multi Protocol Label Switching. *Proceedings ITC17* (Sept 2001) Salvador, Brazil.