Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Computer Networks 50 (2006) 317–331

Computer Networks

www.elsevier.com/locate/comnet

Fast network re-optimization schemes for MPLS and optical networks

Randeep Bhatia [*], Murali Kodialam, T.V. Lakshman

High Speed Networks Research Department in Bell Labs, Lucent Technologies, USA

Received 18 May 2004; received in revised form 5 February 2005; accepted 20 March 2005
Available online 13 July 2005

Responsible Editor: A.K. Somani

Abstract

This paper presents algorithms for re-optimizing network routing in connection-oriented networks such as Multi-Protocol Label Switched (MPLS) networks. The objective in re-optimization is to allow the network to carry more traffic without adding capacity. The need for re-optimization arises because of dynamic connection routing where connections, such as bandwidth guaranteed Label Switched Paths (LSPs) in MPLS networks, are routed as they arrive one-by-one to the network. Continual dynamic routing leads to network inefficiencies due to the limited information available for routing and due to simple path selection algorithms often used to satisfy connection set-up time constraints. We present a re-optimization scheme, where the re-optimizer constantly monitors the network to determine if re-optimization will lead to sufficient network efficiency benefits. When sufficient benefits can be obtained, the re-optimizer computes the least cost set of connections which must be re-routed to attain the necessary network efficiency and then computes the routes for the connections to be re-routed. We develop efficient re-optimization algorithms and demonstrate by simulations that several network performance metrics are significantly improved by re-optimization.
© 2005 Elsevier B.V. All rights reserved.

Keywords: MPLS; Network optimization; Linear programming; Approximation algorithms

1. Introduction

We consider efficient network routing in connection-oriented networks such as MPLS, Optical, and ATM networks. Efficient network routing with QoS guarantees has been well studied in both offline (where all connections to be routed are known ahead of time) and online (where connections arrive

[*] Corresponding author.
E-mail addresses: randeep@research.bell-labs.com (R. Bhatia), muralik@dnrc.bell-labs.com (M. Kodialam), lakshman@research.bell-labs.com (T.V. Lakshman).

1389-1286/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2005.03.004

to the network one-at-a-time) contexts [13,17,23]. However, the hybrid scenario where the network dynamically routes most connections in an online fashion and occasionally re-optimizes the network routing has not been studied extensively. This paper considers network re-optimization in dynamically routed connection-oriented networks.

Online connection routing is used in networks for quick set-up of connections. An example is the set-up of bandwidth guaranteed LSPs in MPLS networks [10]. Here, LSP set-up requests arrive to edge routers which route the LSP using a quick path selection scheme such as min-hop or shortest path with a priori fixed weights. The routing algorithm uses the OSPF link-state database to obtain topology and link-bandwidth usage (made possible by traffic engineering extensions to OSPF). The simple path selection scheme and the limited information available for online routing can cause network capacity to be inefficiently used. It is possible that LSP requests between certain ingress–egress pair are rejected due to lack of capacity whereas a more efficient network routing would allow successful LSP routing. Furthermore the paths for the arriving connections are computed according to the current state of the network, which includes the routing of the existing connections. As the network and traffic evolve, the routing of the existing connections may become sub-optimal. This evolution may include network topology changes such as the addition/deletion of new links and/or capacity and network demand changes resulting from customer churn and varying demand for different services. The objective of network re-optimization is to better adapt to network and traffic evolution and to offset the inefficiencies of online dynamic routing by occasional re-routing of LSPs.

We develop a re-optimization scheme where the re-optimizer monitors the network by keeping track of the aggregate routed traffic between the network ingresses and egresses. It also obtains the network topology by passive peering to the network routing protocol or by periodically obtaining the routing protocol's link-state data base from the routers (or switches) in the network. Knowing the currently routed aggregate traffic, the re-optimizer computes the potential benefit from re-optimization. This is done by comparing the amount of additional traffic that can be accommodated in the current network, if the traffic were to scale proportionally to the current load, to that which can be accommodated if the network were to be re-optimized. If this re-optimization benefit is considered sufficient (exceeds a set threshold), then the re-optimizer computes the set of connections which need to be re-routed so as to incur the minimum re-routing cost. Re-routing costs are determined using a re-routing penalty associated with every connection. The new routes for the set of connections to be re-routed is also computed by the re-optimizer. A key feature of our scheme is the ability to keep the network well balanced by ensuring sufficient available capacity between all ingress–egress pairs so that connections arriving to specific ingress–egress pairs are not all rejected. Note that since the re-optimization algorithms do not run on the network elements and re-optimization is infrequent, the re-optimization algorithms do not have to be restricted to very simple computations as is the case for online routing. Nevertheless, for large networks it is desirable to find computationally efficient re-optimization schemes.

Overall, our scheme has the following characteristics:

1. Use of limited network knowledge to compute re-optimization benefit.
2. Use of a network efficiency measure that ensures better network performance for online routing and the admission of more connections to the network.
3. Minimal re-routing to achieve high efficiency.
4. Computationally efficient.

2. Related work

Service providers use network re-optimization for increased utilization of their network infrastructures thereby deferring capital spending on new equipment. Some limited form of re-optimization is also built into network elements. For example in IP network routing is done over shortest paths and as the network evolves over time, the routing is adjusted to take advantage of the new shortest paths. In MPLS networks the same applies for connections that are routed using Constrained Shortest Path First algorithm (not explicitly routed). Some of the earlier work on re-optimization was done in the context of Digital Cross-Connect (DCS)-based networks [3,4]. Later, with the deployment of ATM as a networking technology, re-configuration of ATM networks was explored, leveraging ATM Virtual Paths [14,20–22,26]. More recently, re-configuration has been studied in the general context of optical networks [6–9,12,18,19, 24,25]. In some of this work [6,19,24] the underlying virtual topology of the optical network is re-configured to adapt to changing traffic patterns. The issues considered there are very different from those considered in our work since we assume that the underlying network topology is fixed and only the connection paths are to be re-optimized. In some of the other work re-configuration is done to improve the ability of the network to protect against multiple failures [18]. In [8] the authors study the problem of re-optimizing lightpaths in resilient mesh optical networks. In their model each demand is routed using a pair of disjoint primary and backup paths and sharing of backup resources is allowed. They consider a re-optimization algorithm that re-routes both primary and backup paths. In addition a partial re-optimization algorithm that re-routes only the backup paths is also considered. In our work we do not explicitly model the network provisions for supporting restoration under failures. However, our results are applicable to many link based local protection schemes where the network capacity is pre-partitioned a priori into primary and backup capacity and where routing of primary paths and bypass tunnels is done independent of each other. These simple pre-partitioning schemes have recently proved useful for supporting fast local restoration in MPLS and Optical networks [1,2,15]. For such resilient mesh networks our re-optimization scheme can be applied to optimize the primary paths without impacting the networks ability to guarantee local restoration for the provisioned connections.

3. Outline and assumptions

For illustrative purposes, consider an MPLS network where bandwidth guaranteed LSPs are provisioned between edge routers. (We use the terms LSPs and connections synonymously in the rest of the paper.) Requests for new LSPs arrive to the network over time. When an LSP set-up request arrives to an edge router, if the network has enough capacity to accommodate the request, the request causes an LSP set-up using a signaling protocol such as RSVP-TE or CR-LDP. Otherwise the request is rejected. Requests for tear-down of existing LSPs also arrive over time leading to the network freeing up resources for the removed LSPs. Thus at any given time the network may have LSPs provisioned between certain pair of edge routers. However, the chosen network routing may not be efficient in the sense that the network may have insufficient capacity between certain ingress–egress pairs to accommodate new LSP requests, whereas a different allocation of routes to LSPs would permit it. The networks routing optimality may be restored by occasional re-optimization that re-routes some of the LSPs. Note that features such as the make-before-break feature in RSVP-TE [5] are usable for LSP re-routing. The paper presents an efficient scheme for re-optimization that can dramatically improve the performance of the network in terms of its ability to accommodate new LSPs.

The network is to be re-optimized to maximize its ability to accommodate the future demands. Even though demands (connection or LSP set-up requests) arrive in an online fashion, we will assume that the long term average traffic between ingress–egress pairs remain in proportion to what the network is currently carrying (i.e., in the absence of traffic forecasts, we take the current network traffic to be indicative of the long term to within a scaling factor). Here a demand for a source sink (ingress–egress) edge router pair represents the aggregate bandwidth of all the LSPs to be provisioned between the source sink pair.

We define a "Network Efficiency Measure" for measuring the instantaneous routing efficiency with respect to performance metrics of interest. Informally this measures the fraction or multiple of the traffic demand matrix that can be accommodated by the network. In general the network may be able to accommodate a bigger fraction (or multiple) of the demand between a given source sink pair but at the expense of other source sink pairs. However we are interested in a fair network efficiency measure that tries to maximize the minimum fraction of demands that can be accommodated for any given source sink pair. We use a network efficiency measure where we maximize the minimum fraction of demands accommodated for every source sink pair.

For a fixed demand matrix and for a particular network efficiency measure the re-optimization algorithm works as follows. Given the network with some currently provisioned LSPs we measure the efficiency of the network, assuming that the currently provisioned LSPs stay as they are. Next we compute the maximum possible gain in the efficiency of the network that can be obtained by re-routing the provisioned LSPs. If this gain in network efficiency is significant then we proceed with the re-routing of the LSPs. Finally among all the re-routings that result in the same improvement in the network efficiency the algorithm chooses the one which minimizes the cost of re-routing. The algorithm allows the operator to assign a cost-benefit measure down to the level of individual LSPs to guide the algorithm in picking the solution of minimum cost.

We now describe our scheme in more detail. We will start out by assuming the demand matrix is scaled from the currently provisioned traffic demands between source sink pairs and we will use the network efficiency measure to maximize the minimum fraction of demands accommodated for every source sink pair.

4. Key ideas for the re-optimization scheme

In this section, we give an informal description of the key ideas used for the re-optimization scheme. The next section presents a more formal mathematical description.

4.1. Illustrative example

Connection-oriented networks that use online routing tend to get unbalanced over time resulting in uneven load distribution. This may lead to some links getting congested that are "critical" for carrying some future demands. Re-optimization involves balancing the load on the links such that network congestion is alleviated. We illustrate this with a simple example. Fig. 1 shows a network of 9 nodes. Associated with each link is its capacity as shown in the figure. Nodes 1 and 3 are ingress routers and nodes 2 and 4 are egress routers. Demands for connections arrive between two pair of nodes $(1,2)$ and $(3,4)$. We assume that the network uses min-hop routing. For illustrative purposes we make the assumption that all connections require 1 Mbps of bandwidth. We also assume that connections last for a long time, thus a connection once provisioned stays provisioned in the network permanently.

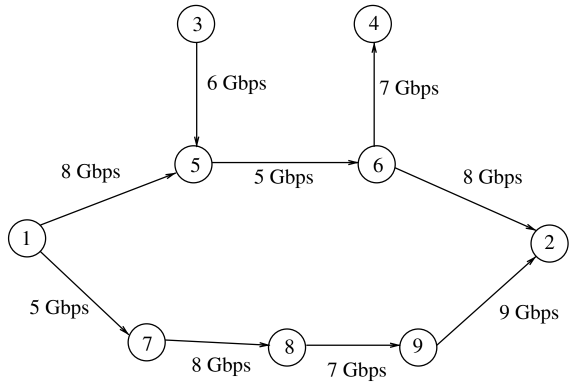

Fig. 1. Illustrative example.

Let the first demand be for a connection from node 1 to node 2 of bandwidth 1 Mbps. Note that this will be routed over the path $1, 5, 6, 2$ since this is the min-hop path between node pair $(1, 2)$. In fact all connections for node pair $(1, 2)$ will be routed over the path $1, 5, 6, 2$ until some link on this path runs out of available capacity. Subsequently additional connections for node pair $(1, 2)$ will be routed over path $1, 7, 8, 9, 2$. The connections for the node pair $(3, 4)$ have only one choice: they can only be routed over path $3, 5, 6, 4$. Note that link $(5, 6)$ is one of the critical links for the connections for node pair $(3, 4)$ and as long this link has available capacity all the newly arrived connections, for both node pairs, will be routed over this link. At the same time the other links such as those on path $1, 7, 8, 9, 2$ remain unloaded. Thus assuming a uniform mix of connections for the two node pairs, over time the network would get unbalanced.

Now consider the scenario where the network is to support 5000 connections for each of the node pairs. Let us say at some time t_1 2500 connections for each of the node pair $(3,4)$ and $(1,2)$ have arrived and been provisioned in the network. Note that connections for node pair $(3,4)$ are routed over path $3,5,6,4$ while those for node pair $(1,2)$ are routed over path $1,5,6,2$. At this point the link $(5,6)$ is loaded to its full capacity and no more connections can be accommodated in the network for node pair $(3,4)$. Note that this is not due to the network not having enough capacity to admit any more connections for node pair $(3,4)$. But it is due to the load imbalance which results in some capacity getting stranded in the network. This stranded capacity can be recovered by re-balancing the network. One possible re-optimization involves migrating the connections for node pair $(1,2)$ from the path $1,5,6,2$ to the path $1,7,8,9,2$, thus alleviating the congestion on the link $(5,6)$. Note that this creates enough capacity in the network to accommodate 2500 more connections for node pair $(3,4)$. Let this re-optimization be performed at time t_1 and then subsequently at some time t_2 let 1000 additional connections for each of the node pair $(3,4)$ and $(1,2)$ have arrived and been provisioned in the network. Note that the additional connections for node pair $(3,4)$ are routed over path $3,5,6,4$ while those for node pair $(1,2)$ are routed over path $1,5,6,2$. At this point the network can only admit 500 more connections for node pair $(3,4)$. However by doing another re-optimization where all of the $(1,2)$ connections are migrated from the path $1,5,6,2$ to the path $1,7,8,9,2$, we create enough capacity to accommodate 1500 additional connections for node pair $(3,4)$.

The example shows that periodic re-optimization of these connection-oriented networks helps recover the stranded capacity and keeps the network re-balanced. However re-optimization has a cost associated with it in terms of the disruption of provisioned connections possibly resulting in traffic hits. Thus re-optimization has to be carefully planned to maximize the benefits gained in terms of the recovered stranded capacity. To do this we need some quantification of how useful re-optimization is for a given state of the network. To this end we define a "network efficiency" measure. We compute the network efficiency before and after (a potential) re-optimization. If re-optimization can lead to a significant gain in the network efficiency we proceed with re-optimization otherwise we continue with the current state of the network.

In the example above the "network efficiency" of the network is computed at time t_1 as follows. Given the state of the network at time t_1 we ask the question what fraction of the demand matrix (5000 connections for each of the node pairs) can be accommodated by the network. Note the network has already accommodated 2500 connections for each node pair and the network cannot accommodate any more connections for the node pair $(3,4)$. Thus this fraction is $2500/5000 = 0.5$. Thus the network efficiency of the network at time t_1 is 0.5. However if we were to re-reroute all the connections provisioned on the path $1,5,6,2$ to the path $1,7,8,9,2$ then the network can accommodate a total of 5000 connections for each of the node pairs (assuming an optimal routing algorithm is used for routing the additional connections), with all the connections for node pair $(1,2)$ routed on the path $1,7,8,9,2$, and all the connections for node pair routed on the path $3,5,6,4$. Thus the network efficiency after re-optimization is $5000/5000 = 1.0$ a gain of 100%. Similarly at time t_2 the network can only admit a total of 4000 connections for node pair $(3,4)$, giving a network efficiency of $4000/5000 = 0.8$. Also note that the network can admit 5000 connections for each node pair after re-optimization giving a network efficiency of 1.0 subsequent to re-optimization. Thus re-optimization at time t_2 will result in a gain of 25%.

5. Basic re-optimization scheme

5.1. Network model

In the following we describe the network model that we use in the rest of the paper. We assume we are given a network with N routers (also called nodes) and M links between them. We denote the network by a graph $G = (V, E)$ where V is the set of nodes and E the set of links in the network. The network supports demands between n source sink router pairs (i, j) where each source and sink router is an edge router. These source sink pairs are numbered 1 to n and for the kth source sink pair (i, j) there is currently d_k amount of end to end demand provisioned in the network. This demand for a source sink pair is measured by the aggregate bandwidth allocated to all the LSPs between the source sink pair. We also denote this currently provisioned demand between source sink pair (i, j) by $d_{(i,j)}$. In addition we denote by $D_{(i,j)}$ or D_k the total desired demand to be supported for the kth source sink pair (i, j) as defined by a demand matrix D. For each link e in the network $c(e)$ and $b(e)$ denote its current residual capacity and its total capacity respectively.

In this section we assume that the demand matrix is defined by the currently provisioned demands between source sink pairs and we will use the network efficiency measure to maximize the minimum fraction of demands accommodated for every source sink pair. Thus $D_{(i,j)} = d_{(i,j)}$ and the network efficiency measure is the largest value for λ such that at least $\lambda d_{(i,j)}$ demand can be accommodated between source sink pair (i, j). These results extend to the more general demand matrix as shown in Section 8.

5.2. Solution approach

In this section we outline our approach to re-optimization. We first present our ideas for quantifying the usefulness of re-optimization for a given network. To this end we define a natural "network efficiency" measure. We compute the network efficiency before and after (a potential) re-optimization, which in conjunction with a network re-optimization benefit threshold help us determine the benefit of re-optimization. Having determined that re-optimization is beneficial, we determine a re-optimization solution that involves minimal amount of disruption to the network.

Any re-optimization solution for MPLS and Optical networks must be designed to have the most benefit in terms of reclaiming stranded capacity while having minimal impact on the network performance. For instance re-optimization involves moving connections which unless implemented carefully may cause disruptions in the services carried over the connections. Also re-optimization must not hinder with the network's ability to provide restoration guarantees to resilient connections. In addition re-optimization must not require splitting the traffic flow of services that must be routed on a single path to maintain the order of packet arrival at the destination node. Our re-optimization scheme strives to achieve these goals as outlined below.

Make-before-break for minimizing service disruptions. We propose to use the MPLS and Optical networks "make-before-break" mechanism to minimize disruption to network services due to traffic re-routing resulting from re-optimization. This involves first setting up the new path (resource reservation) before re-routing traffic on it and only tearing down the resources on the old path once the service is fully established on the new path. Note that the only disruption this may cause is packets arriving out of order for a small time period that it takes to switch over from one path to another.

Dealing with resilient connections. As mentioned earlier our re-optimization scheme does not explicitly take into the network provisions for supporting restoration under failures. However, our scheme is applicable to many link based local protection schemes used in MPLS and Optical networks for supporting resilient connections. In these schemes [1,2,15] the network capacity is pre-partitioned a priori into primary and backup capacity and routing of primary paths and bypass tunnels is done independent of each other. For such resilient mesh networks our re-optimization scheme can be independently applied to optimize the primary paths without impacting the networks ability to guarantee local restoration for the provisioned connections.

Traffic splitting. Our scheme is designed to ensure that the traffic flow of services that must be routed on a single path is not split after re-optimization. Although we describe our scheme for the case when all services are un-splittable, it can be easily modified to the case where not all services are of this type. Note that the re-optimization gains may be larger if we are allowed to split the service traffic over multiple routes. Our scheme first computes a solution without regard to the splitting constraint. Then it uses this solution as a basis for computing an un-split solution while trying not to deviate too much from the split solution.

5.3. Network efficiency measure before re-optimization

We now formally define the network efficiency measure for a network with currently provisioned demands $d_{(i,j)}$ between source sink pairs (i,j). Here we are interested in the network efficiency measure before re-routing. Thus we assume that the currently provisioned flows stay as they are and we want to compute the additional flow that can be routed between each source sink pair. Here we measure the efficiency of the network by the maximum value of $\lambda + 1$ where there exist a multi-commodity flow between every source sink pair such that the flow assigned in this multi-commodity flow to source sink pair (i,j) is $d_{(i,j)}\lambda$ and the total flow through any link e is at most $c(e)$. Thus if r is the efficiency then we can increase the flow between every source sink pair by a factor of $r-1$. Note also that $r \geqslant 1$ and the larger r is the more "efficient" the network is in admitting new connections between any source sink pair. Intuitively we are taking the currently provisioned demand matrix as a measure for the expected demand in the future to within a multiplicative factor such that we expect the demand ratios for future demands to follow the ratios in the current demand matrix.

We take the total traffic demand between a source–sink router pair to be a single commodity. Let a path in the network be defined by a sequence of nodes $u_1, u_2, \ldots, u_a$ where each $(u_i, u_{i+1}) \in E$ is a link and two nodes u_i and u_j are the same if and only if $i = j$. Let $\mathscr{P}$ denote the set of all possible paths in the network. We denote by $f^k(P)$ the flow for commodity k for the kth source sink pair on path $P \in \mathscr{P}$. Note that if path P does not start or end at the source and sink nodes of commodity k then $f^k(P) = 0$. Then r can be obtained by solving the following multi-commodity concurrent flow problem. Specifically if $\lambda = \lambda^*$ is the optimal solution to the following multi-commodity problem then $r = \lambda^* + 1$. Here the variable P ranges over all paths in the network (i.e., the set $\mathscr{P}$)

$$\max \lambda \tag{1}$$

$$\sum_{k=1}^{n} \sum_{P:e\in P} f^k(P) \leqslant c(e) \quad \forall e \in E, \tag{2}$$

$$\sum_{P} f^k(P) = \lambda d_k \quad \forall k, \tag{3}$$

$$f^k(P) \geqslant 0 \quad \forall P\, \forall k. \tag{4}$$

5.4. Network re-optimization benefit measure

This measures the maximum network efficiency that can be obtained by re-routing the existing demands in the network.

The network re-optimization benefit measure is thus the maximum value of λ where there exist a multi-commodity flow between every source sink pair such that the flow assigned in this multi-commodity flow to source sink pair (i,j) is $d_{(i,j)}\lambda$ and the total flow through any link e is at most $b(e)$. Thus if b is the network re-optimization benefit then note that $b/r \geqslant 1$ and that by re-routing the existing demands the efficiency of the network can be increased to b.

Note that b can be obtained by solving the following multi-commodity concurrent flow problem. Specifically if $\lambda = \lambda^*$ is the optimal solution to the following multi-commodity problem then $b = \lambda^*$. Here the variable P ranges over all paths in the network (i.e., the set $\mathscr{P}$)

$$\max \lambda \tag{5}$$

$$\sum_{k=1}^{n} \sum_{P:e\in P} f^k(P) \leqslant b(e) \quad \forall e \in E, \tag{6}$$

$$\sum_{P} f^k(P) = \lambda d_k \quad \forall k, \tag{7}$$

$$f^k(P) \geqslant 0 \quad \forall P\, \forall k. \tag{8}$$

5.5. Network re-optimization benefit threshold

A threshold $\alpha > 1$ is used to determine if network re-optimization is beneficial. In other words we say that it is beneficial to do network optimization if the ratio of the network benefit measure b and the network efficiency measure r exceeds α. Our goal would be to do re-routing so that after re-routing the efficiency of the new network is $r\alpha$.

5.6. Minimum cost re-routing

As mentioned in Section 1 our goal is not just to improve the network efficiency but to do this with re-routing of minimum cost. Note that these are contradictory goals since minimum cost re-routing would imply least network efficiency and vice versa. However in order to achieve a balance we strive to increase the efficiency of the network to $r\alpha$ yet at the same time find the minimum cost re-routing that achieves this network efficiency. Here we outline a scheme for just this.

The algorithm allows the operator to assign a cost-benefit measure down to the level of individual LSPs to guide it in picking the re-routing solution of minimum cost. Let $x^k(P)$ denote the amount of commodity k currently provisioned on path $P \in \mathscr{P}$. Let LSP_k denote the set $\{(P,k): x^k(P) > 0\}$. We assume that LSP_k is some operator specified splitting of the flow for commodity k in the existing network. We will assume an ordering of the elements of LSP_k such that the ith element is denoted by tuple $(k, f_k(i))$ and would correspond to flow for commodity k on path $P_{f_k(i)}$, for some function f_k. For convenience we will write f_k as f whenever the dependence on k is obvious from the context. Thus we will denote the ith tuple as $(k, f(i))$. Let LSP denote the union of the n sets LSP_k. We assume that the operator has associated a cost $c^k(P)$ for the flow of commodity k that is not routed on path P in the re-routing. Let $(k, f(i)) \in LSP_k$. Let P denote the path $P_{f(i)}$. Thus commodity k has $x_P^k > 0$ flow provisioned on path P. Let after re-routing only a fraction f of this flow stay on path P. Then a cost of $(1-f)c^k(P)$ is incurred for the commodity k for this path in the re-routing solution.

We now present a scheme for determining the re-routing solution of minimal cost that achieves the required re-optimization benefit. This scheme is based on solving a budgeted version of the maximum concurrent flow problem. First we split each commodity k into $|LSP_k| + 1$ commodities such that the ith commodity among the first $|LSP_k|$ commodity corresponds to the ith tuple $(k, f(i)) \in LSP_k$. We set the demand for the ith commodity among the first $|LSP_k|$ commodity, which we denote by (k, i), to $x^k(P_{f(i)})/(r\alpha)$. We set the demand for the $|LSP_k| + 1$th commodity, denoted by $(k, |LSP_k| + 1)$, to

$$d_k - \sum_{(k,i)} x^k(P_{f(i)})/(r\alpha) = d_k(1 - 1/(r\alpha)).$$

For sake of simplicity we will use the notation P_i for $P_f(i)$ in the following description.

The ith commodity denoted by (k, i), for $i \leqslant |LSP_k|$ can be routed on any path, however for any flow not routed on path P_i a cost per unit flow of $c^k(P_i)/x^k(P_i)$ is incurred. We solve a maximum concurrent multi-commodity flow problem for these commodities subject to the constraint that the total cost incurred is some budget B. Our aim is to find the smallest value for B such that the solution to this maximum concurrent flow problem is $\lambda = r\alpha$. Putting this together we get the following budgeted maximum concurrent flow problem:

$$\max \lambda \tag{9}$$

$$\sum_{k=1}^{n} \sum_{i=1}^{|LSP_k|+1} \sum_{P:e\in P} f^{(k,i)}(P) \leqslant b(e) \quad \forall e \in E, \tag{10}$$

$$\sum_{P} f^{(k,i)}(P) = \lambda d_{(k,i)} \quad \forall k \;\; 1 \leqslant i \leqslant |LSP_k| + 1, \tag{11}$$

$$f^{(k,i)}(P) \geqslant 0 \quad \forall P \; \forall k \;\; 1 \leqslant i \leqslant |LSP_k| + 1, \tag{12}$$

$$\sum_{k=1}^{n} \sum_{i=1}^{|LSP_k|} \frac{c^k(P_i)}{x^k(P_i)} \sum_{P \neq P_i} f^{(k,i)}(P) \leqslant B. \tag{13}$$

Here $f^{(k,i)}(P)$ denotes the flow for commodity (k, i) on path P and $d_{(k,i)}$ denotes the demand for commodity (k, i). Note that here $c^k(P_i)f$ is the cost incurred for routing a fraction $1-f$ of the currently provisioned flow on path P_i for commodity k.

Claim 1. *There exists a choice of B for which the optimal solution to this LP* $\lambda^* = r\alpha$.

Proof. Let $\lambda = g(B)$ denote the optimal solution to the budgeted LP for a given value B of the budget constraint. Note $g(B)$ is monotonically non-decreasing in B. In addition $g(B)$ is a continuous function of B, since from a flow scaling argument it follows that for $B' \leqslant B$, $g(B') \geqslant g(B)B'/B$. Note that if B is set to the value

$$\sum_{k=1}^{n} \sum_{i=1}^{|LSP_k|} c^k(P_i)$$

then $\lambda = r\alpha$ is a feasible solution to the budgeted LP. Thus for this choice of B we have $g(B) \geqslant r\alpha$. Also if B is set to 0 then $\lambda = r\alpha$ is not a feasible solution to the budgeted LP, since then even without re-routing we will have a network efficiency of $r\alpha$. Thus for this choice of B we have $g(B) < r\alpha$. Thus by continuity of $g(B)$ there exists a B for which $g(B) = r\alpha$. $\square$

Corollary 2. *The choice of B for which* $\lambda^* = r\alpha$ *can be found to within a factor of* $(1 + \epsilon)$ *by doing a binary search for a additional running time factor of* $\log_{1+\epsilon}\sum_{k=1}^{n}\sum_{i=1}^{|LSP_k|}c^k(P_i)$.

5.7. LSP re-routing

Note that so far we have looked at the flows for a commodity at the granularity of the paths which define the set LSP_k. In general for each path P such that $(P, k) \in LSP_k$ we may have multiple LSPs (connections) for commodity k provisioned on it, and our goal is to find a re-routing for these individual connections so as to maximize the total profit. We use the solution of the linear program given by the constraints (9)–(13) as a starting point for computing a re-routing solution for the individual connections.

The overall algorithm for the re-routing of the connections uses at most four phases. The first phase involves solving the multi-commodity flow problems as described in previous sections. In the second phase we find a set of connections that do not need to be re-routed, since they are currently routed over a path on which sufficient flow is routed by the multi-commodity solution. The connections are selected in a priority order which can be modified by the operator. As connections are assigned to path the flows on the paths are updated to reflect the capacity that is used up by the assigned connections. The third phase is a re-routing phase where connections unassigned in phase two are assigned to paths on which positive flow was routed in the multi-commodity solution (and as updated by phase two). Here again the algorithm selects the connections to be re-routed in an order defined by their priorities and it tries to assign selected connections to the first path in which they can fit i.e., the path has enough flow associated with it, for the source sink pair of the connection, to accommodate the connection. As connections are assigned to path the flows on the paths are updated to reflect the capacity that is used up by the assigned connections. The fourth phase is required if some connections remain unassigned after the three phases. In our simulations this phase was rarely invoked. This phase involves re-routing the leftover connections by using constrained shortest path first (CSPF) over the residual graph resulting from phase three.

Note that the re-routing scheme described above requires the knowledge of the currently provisioned connections on each path for each commodity. This knowledge is available on the edge routers corresponding to the source sink pair for each of the commodity k. The edge router for commodity k just needs to know the flows f_P^k for each $(P, k) \in LSP_k$. Therefore a natural place to implement such a scheme would be in the edge routers. This means that the route server need not have information about all the connections that are provisioned in the network and thus it can gather all the information it needs to solve the three linear programs by passively peering with OSPF-TE and/or looking into the network mib elements.

5.8. Overall scheme

We now describe the overall re-optimization algorithm.

```
Phase I
    $r$ = solution of LP defined by Eqs. (1)–(4)
    $b$ = solution of LP defined by Eqs. (5)–(8)
    **If** $\frac{b}{r} < \alpha$ **then** Stop; /* Re-optimization is not useful */
    $B^*$ = value of $B$ in LP defined by Eqs. (9)–(13) for which $\lambda = r\alpha$
    $f^{(k,i)}(P)$ = solution of LP defined by Eqs. (9)–(13) for $B = B^*$, $\forall k, i, P$
    $f^k(P) = \sum_i f^{(k,i)}(P)\ \forall k, P$

Phase II /* Find connections that are not to be re-routed */
  For $k = 1$ to $n$
    For connection $s_i = s_1, s_2, \ldots$ of kth source sink pair
      Let $P$ be the path for $s_i$.
      If Bandwidth $b_i$ for connection $s_i$ is $\leqslant f^k(P)$.
        $s_i$ stays routed on $P$.
        $f^k(P) = f^k(P) - b_i$.
      EndIf
    end For
  end For

Phase III /* Re-route remaining connections */
  For $k = 1$ to $n$
    For connection $s_i = s_1, s_2, \ldots$ of kth source sink pair
      For paths $P_j = P_1, P_2, \ldots$ between kth source sink pair
        If Bandwidth $b_i$ for connection $s_i$ is $\leqslant f^k(P_j)$.
          Route $s_i$ on $P_j$.
          $f^k(P_j) = f^k(P_j) - b_i$.
        EndIf
      end For
    end For
  end For

Phase IV /* CSPF to re-route un-assigned connections */
  Let $s_1, s_2, \ldots, s_r$ be the set of un-assigned connections
  For $i = 1$ to $r$
    Use CSPF to route connection $s_i$ over the residual network
    Update residual network
  end For
```

6. Efficient implementation

Note that our scheme relies heavily on solving linear programs for certain multi-commodity flow problems. Use of linear program solvers is computationally prohibitive if the schemes are to run on devices with limited computational capabilities such as edge routers. We therefore seek fast and efficient algorithms for solving these linear programs while trading off optimality of the found solutions. We use the machinery developed by Garg et al. [11,16] for this purpose.

Note that the results in [11,16] can be directly applied to solve linear program given by Eqs. (1)–(4) and to solve linear program given by Eqs. (5)–(8) to obtain the network efficiency r to within any specified error ϵ of the optimal value and to obtain the network re-computation benefit measure b to within any specified error ϵ. The techniques presented in [11,16] rely only on shortest path computations on a suitable network and hence are computationally efficient. More formally the running time is given by:

Claim 3 [16]. *There is a Fully Polynomial Time Approximation Scheme for solving the maximum concurrent multi-commodity flow problem that runs in time* $O(\epsilon^{-2}M^2\log^{O(1)}M)$ *for a connected network with M edges.*

Our budgeted version of the maximum concurrent flow problem as defined by Eqs. (9)–(13) differs from the same problem defined in [11], where costs are associated with flows over edges and not with paths for the flows. Hence the results in [11] are not directly applicable. However we can modify the results in [11] to show:

Claim 4. *There is a Fully Polynomial Time Approximation Scheme for solving the budgeted version of the maximum concurrent multi-commodity flow problem that finds a* $(1-\epsilon)^{-3}$*-approximation and runs in time* $O(2n\log n + M)C_2T_{sp}$, *where* $C_2 = O(\frac{1}{\epsilon}\log_{1+\epsilon}\frac{M}{1-\epsilon})$ *and* $T_{sp} = O(N\log N + M)$ *is the running time of a shortest path algorithm such as Dijkstra. Here n is the number of source sink pairs, N and M are the number of nodes and edges respectively in the network.*

Note that to find the smallest value of the budget for which a network efficiency of $r\alpha$ is achieved we need an additional running time factor of $\log_{1+\epsilon}\sum_{k=1}^{n}\sum_{i=1}^{|LSP_k|}c^k(P_i)$, as established in Corollary 2.

The running time of Phase II and III of the algorithm is dependent on the number of tuples (P, k) for which $f^k(P) > 0$, and the number of currently provisioned connections in the network. The former is at most $O(2n \log n + M)C_2$, where C_2 is as defined in Claim 4, since there are these many iterations of the algorithm for solving the budgeted maximum concurrent flow problem each potentially routing the flow on a distinct path.

Finally the running time of Phase IV is dependent on the number of un-assigned connections which is bounded by the number of currently provisioned connections in the network.

7. Simulation

In this section we present our simulation results for the basic re-optimization scheme. For our simulations we used a network of 20 nodes which is shown in Fig. 2. All links in the network are bidirectional. This network has 6 ingress routers: 1, 3, 4, 5, 10, 11 and 6 egress routers: 6, 7, 8, 9, 12, 18. The network has 36 source sink node pairs one for each ingress egress router pairs. For solving the multi-commodity flow programs we use the efficient algorithms described in Section 6 with ϵ set to 0.01.

We set out with two main goals for our simulations. First we want to study the benefit of re-optimization for varying traffic demands. For this we set the bandwidth of all the links at 15 Gbps and

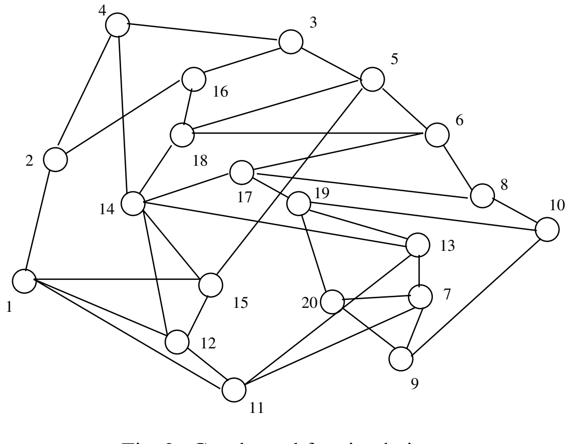

Fig. 2. Graph used for simulations.

we try out different randomly generated traffic demand matrices. Next we are interested in assessing the sensitivity of the re-optimization scheme to the link bandwidths and to network topology. For this we fix a demand matrix and we try out different bandwidths for the links ranging from 5 Gbps to 50 Gbps. Next we fix a demand matrix and fix all the link bandwidths at 15 Gbps but try out different topologies obtained by deleting nodes from the network.

For the first set of simulations our set-up is as follows. We generate a demand matrix at random for the source sink pairs. Then we load the network with connections that arrive for node pairs at rate proportional to the demand for the node pairs in the demand matrix. Each connection is an LSP of bandwidth 1 Mbps. We route these connections using min-hop routing. As the network is getting loaded we continuously monitor the network efficiency. When the network efficiency of the loaded network falls below approximately 10% of the network efficiency of a network obtained by re-routing the provisioned connections we invoke re-optimization. We then measure the re-optimization gain in terms of the load that can be handled by the re-optimized network versus the load that can be handled by the network if no re-optimization was performed. We measure this gain in terms of two quantities. First we compute the maximum number of additional connections that the network can accommodate for each source sink pair just before re-optimization and we compare it to the same quantity immediately after doing the re-optimization. Second after re-optimization we load the network with 25% additional connections (proportional to the demand for the node pairs in the demand matrix). We count the number of connections that are rejected for each source sink pair. We then do the same for the network without performing re-optimization and compare the two quantities. We also do this before and after comparison after loading the network with 50% additional connections. Finally we also compute the network efficiency for different loading of the network for a particular demand matrix both with and without re-optimization. In our simulations the number of connections provisioned in the network ranged from 150000 to

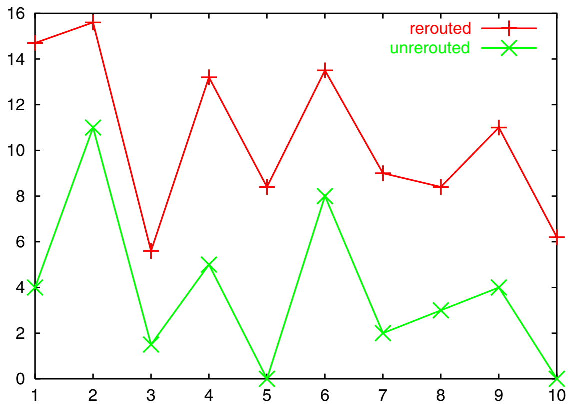

Fig. 3. Minimum of maximum number of connections that can be accommodated for all source sink pairs.

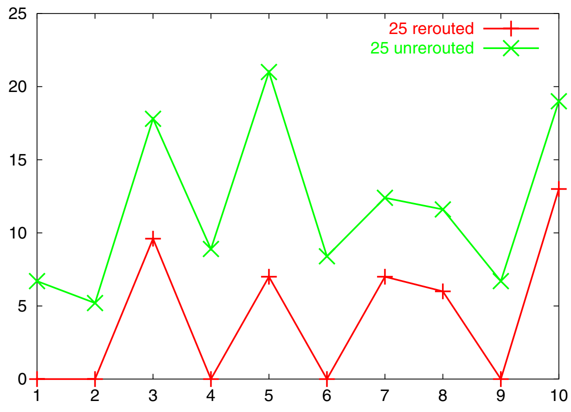

Fig. 4. Max percentage of connections rejected for all source sink pairs with 25% additional loading.

250 000. Our simulations were performed with 10 different demand matrices.

Fig. 3 shows the minimum over all source sink pair of the maximum number of additional connections that can be accommodated by the network for each source sink pair both before re-routing and after re-routing. The number shown in the *Y*-axis is this quantity expressed in units of 1000 connections. The values are plotted for 10 different demand matrices. Note that we could have plotted instead of the minimum the average value for all source sink pairs of the maximum number of additional connections. However in general only a handful of the source sink pairs are blocked due to the un-even load distribution. Thus the average values tend to be dominated by the majority of non-blocked source sink pairs and hence are not much different. By plotting the minimum value instead we are able to see how the load balancing is able to free up the blocked source sink pairs.

Fig. 4 shows the maximum percentage of demands that cannot be accommodated by the network for each source sink pair, when the network is loaded with 25% additional connections from the point where re-optimization is found to be useful. The figure compares these percentages for the case when re-routing is performed with the case when the connections are not re-routed. The number shown in the *Y*-axis is the maximum percentage of demands and is plotted for 10 different matrices. Note that here too we could have plotted instead of the maximum percentages the average percentages for all source sink pairs. However the same reasoning of there being very few blocked source sink pairs justifies the use of minimum percentage for illustrating the gains of re-optimization. Fig. 5 shows the same for the case when the network is loaded with 50% additional connections from the point where re-optimization is found to be useful.

Finally Fig. 6 shows the variation in network efficiency as the network is loaded with connections, both when re-optimization is performed and when the network is not re-optimized. For this simulation we fixed a single demand matrix and then we loaded the network with connections that

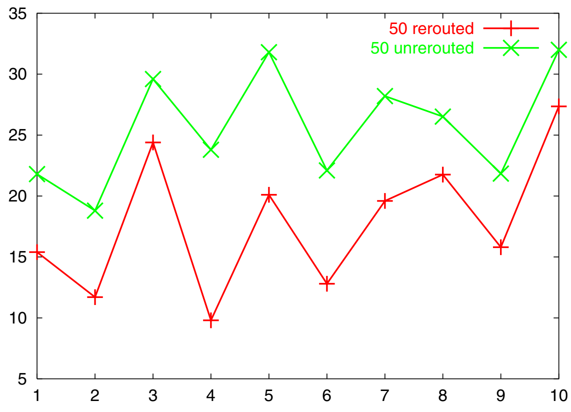

Fig. 5. Max percentage of connections rejected for all source sink pairs with 50% additional loading.

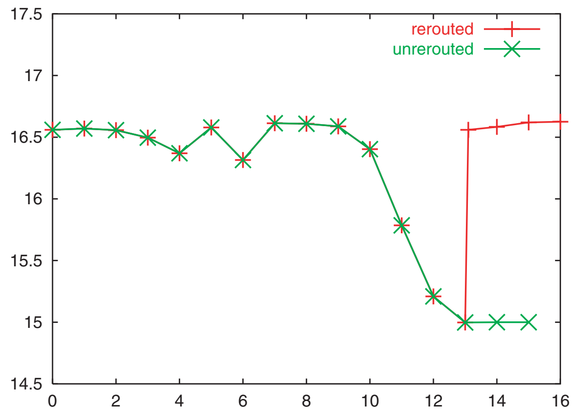

Fig. 6. Normalized network efficiency before and after re-routing.

arrive for node pairs at rate proportional to the demand for the node pairs in the demand matrix. This helps us calibrate the network efficiency in terms of a loading factor of the demand matrix. The *X*-axis represents the various loading of the network (in terms of the demand matrix) and the *Y*-axis plots the network efficiency in terms of a loading factor of the demand matrix. At a loading of 13 times the demand matrix re-optimization is useful since here the network efficiency of the un-optimized network is less than 15 while by re-optimizing the network efficiency is raised to 16.6. Also note that if re-optimization is not performed the network starts rejecting connections at a loading of approximately 15 times the demand matrix. However if re-optimization is performed the first time the network starts rejecting connections is at a loading of 16.5 times the demand matrix.

Our second set of simulations assess the sensitivity of the re-optimization scheme to variation in link bandwidths and network topology for a fixed demand matrix. We first vary the link bandwidths, by randomly selecting individual link bandwidths in the range 5–50 Gbps. We do this eight times and the results are shown in Fig. 7. For providing a reference with the previous set of simulations we set all the link bandwidths at 15 Gbps for the first experiment. The *X*-axis represents the eight different experiments and the *Y*-axis shows the network efficiency with and without re-optimization. The network efficiency is measured

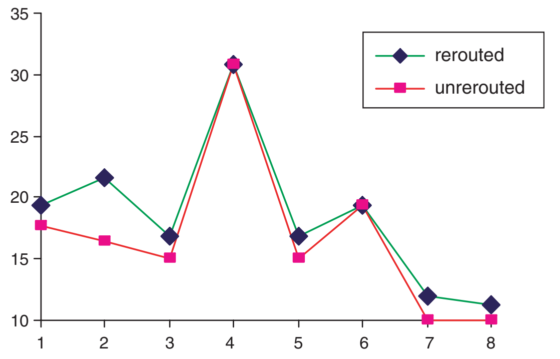

Fig. 7. Network efficiency before and after re-routing for changing link bandwidths.

after the network is loaded with the connections corresponding to the fixed demand matrix. For instance for the second experiment the network efficiency without re-optimization is 16.4 but jumps to 21.6 after re-optimization. Next we set all the link bandwidths at 15 Gbps but varied the network topology by deleting nodes from the network. The result is shown in Fig. 8. We deleted the nodes in the order 19, 20, 16, 15, 2, 7, 1, 17. Again for providing a reference with the previous set of simulations the first experiment is carried out with the full network topology. After deleting each node (in the order mentioned above) we measured the network efficiency with and without re-optimization. Thus there are nine experiments and these

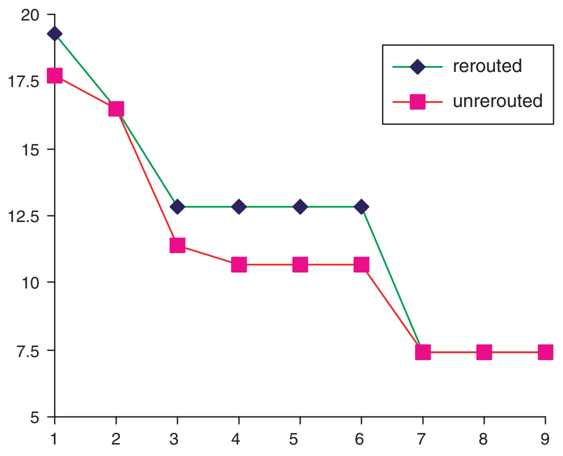

Fig. 8. Network efficiency before and after re-routing for changing topology.

are represented on the X-axis. So for example the third experiment is done after deleting nodes 19 and 20 from the network. The Y-axis shows the network efficiency with and without re-optimization. For instance after deleting nodes 19 and 20 from the network the network efficiency without re-optimization is 11.4 but jumps to 12.8 after re-optimization.

8. Generalizations of the re-optimization scheme

In this section we show how to extend our basic re-optimization scheme to support any general matrix not necessarily the one defined by the currently provisioned demands between source sink pairs.

Here we assume that the demand between the kth source sink edge router pair (i, j) is given by D_k or $D_{(i,j)}$ which is not necessarily equal to d_k or $d_{(i,j)}$. Although it must be the case that $D_k \geqslant d_k$. Note that in this case the only modification to all the linear programs, except the one defined by Eqs. (1)–(4) is the replacement of d_k by D_k everywhere.

Note that the linear program defined by Eqs. (1)–(4) is for computing the network efficiency measure before re-optimization. Let us say we start out by guessing a value for r the network efficiency measure. Let us replace the RHS of Eq. (3) by $D_k - d_k/r$. If we now solve the new linear program and the solution happens to be $\lambda = r$ then note that in this solution exactly $r(D_k - d_k/r) + d_k = rD_k$ amount of flow is routed for source sink pair k including the existing flow. Thus implying that the network efficiency is indeed r. In general our guess for r may be incorrect but we can establish the following.

Claim 5. *Let r be a guess for the network efficiency measure before re-optimization. Then the solution to the modified linear program $\lambda \geqslant r$ if and only if the network efficiency before re-optimization $r^* \geqslant r$.*

Proof. Note that since the amount of flow routed for commodity k is at least d_k we only need to consider values of r for which $rD_k \geqslant d_k$. Or in other words $D_k - d_k/r \geqslant 0$. Note that the total amount of flow routed for commodity k is given by $\lambda(D_k - d_k/r) + d_k = rD_k + (D_k - d_k/r)(\lambda - r)$. which is greater than or equal to rD_k if and only if $\lambda \geqslant r$ Thus $r^* \geqslant r$ if and only if $\lambda \geqslant r$. □

The result in Claim 5 suggests that we can compute r^* by doing a binary search. We only need to search for the value in the range $[1, b]$ where b is the network re-optimization benefit measure as given by the linear program defined by Eqs. (5)–(8).

9. Concluding remarks

In this paper we presented our work on network re-optimization. Network re-optimization introduces new constraints such as minimizing the re-routing cost, working with limited network information (in comparison to offline routing), and using network efficiency measures different from those used for offline routing. We presented efficient network re-optimization algorithms that use limited aggregate information to continually monitor the network for re-optimization opportunities. The algorithm needs more detailed connection specific information only when connection re-routing is to be done. The algorithms are computationally efficient and can be implemented in network management systems for connection-oriented data (MPLS, ATM) networks or optical networks.

References

[1] M. Alicherry, R. Bhatia, Pre-provisioning networks to support fast restoration with minimum over-build, in: IEEE Infocom, 2004.

[2] M. Alicherry, R. Bhatia, Y.C. Wan, Designing networks with existing traffic to support fast restoration, in: 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX), 2004.

[3] D. Anding, Service offerings utilizing digital cross-connect systems a bell south perspective, in: International Conference on Communications, Philadelphia, PA, June 1998, pp. 336–339.

[4] G. Ash, K. Chan, J.F. Labourdette, Analysis and design of fully shared networks, in: 14th International Teletraffic Congress, Antibes Juan-Les-Pins, France, June 1994, pp. 1311–1320.

[5] A. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, G. Swallow, RSVP-TE: extensions to RSVP for LSP Tunnels, IETF RFC 3209, December 2001.

[6] D. Banerjee, B. Mukherjee, Wavelength-routed optical networks: linear formulation, resource budgeting tradeoffs, and a reconfiguration study, IEEE/ACM Transactions on Networking 8 (5) (2000) 598–607.

[7] I. Baldine, G.N. Rouskas, Dynamic reconfiguration policies for WDM networks, in: IEEE Infocom, New York, NY, March 1999, pp. 313–320.

[8] E. Bouillet, J.F. Labourdette, R. Ramamurthy, S. Chaudhuri, Lightpath re-optimization in mesh optical networks, IEEE/ACM Transactions on Networking 13 (2) (2005) 437–447.

[9] S. Datta, Z. Bogdanowicz, Route re-optimization in optical shared mesh networks, in: IASTED International Conference on Wireless and Optical Communications, Alta., Canada, July 2002.

[10] B. Davie, Y. Rekhter, MPLS Technology and Applications, Morgan Kaufman Publishers, 2000.

[11] N. Garg, J. Könemann, Faster and simpler algorithms for multi-commodity flow and other fractional packing problems, in: Proceedings of the 39th Annual Symposium on Foundations of Computer Science, 1998, pp. 300–309.

[12] A. Gencata, B. Mukherjee, Virtual-topology adaptation for WDM mesh networks under dynamic traffic, in: IEEE Infocom, New York, NY, June 2002, vol. 1, pp. 48–56.

[13] R. Guerin, D. Williams, A. Orda, QoS Routing Mechanisms and OSPF Extensions, in: Proceedings of IEEE Globecom, 1997.

[14] A. Herschtal, M. Herzberg, Dynamic capacity allocation and optimal rearrangement for reconfigurable networks, in: IEEE Conference on Global Communications (GLOBECOM), Singapore, SGP, 1995, pp. 946–951.

[15] M. Kodialam, T.V. Lakshman, S. Sengupta, A simple traffic independent scheme for enabling restoration oblivious routing of resilient connections, in: IEEE Infocom 2004.

[16] G. Karakostas, Faster approximation schemes for fractional multicommodity flow problems, in: Proceedings of the Thirteenth Annual ACM–SIAM Symposium on Discrete Algorithms, 2002, pp. 166–173.

[17] K. Kar, M. Kodialam, T.V. Lakshman, Minimum interference routing of bandwidth guaranteed tunnels with applications to MPLS traffic engineering, IEEE Journal on Selected Areas in Communications (December) (2000) (Special Issue on Quality of Service in the Internet).

[18] S. Kim, S.S. Lumetta, Evaluation of protection reconfiguration for multiple failures in optical networks, in: Proceedings of the Optical Fiber Communication Conference, Atlanta, GA, March 2003.

[19] J.F.P. Labourdette, G.W. Hart, A.S. Acampora, Branch-exchange sequences for reconfiguration of lightwave networks, IEEE Transactions on Communications 42 (10) (1994) 2822–2832.

[20] D. Medhi, Multi-hour, multi-traffic class network design for virtual path-based dynamically reconfigurable wide-area ATM networks, IEEE/ACM Transactions on Networking 3 (6) (1995) 809–818.

[21] J.A.S. Monteiro, M. Gerla, Topological reconfiguration of atm networks, in: IEEE Infocom, San Francisco, CA, June 1990, pp. 207–214.

[22] T. Noh, D. Vaman, X. Gu, Reconfiguration for service and self-healing in atm networks based on virtual paths, Computer Networks 29 (16) (1997) 1857–1867.

[23] S. Plotkin, Competitive routing of virtual circuits in ATM networks, IEEE Journal of Selected Areas in Communications (1995) 1128–1136 (Special Issue on Advances in the Fundamentals of Networking).

[24] G.N. Rouskas, M. Ammar, Dynamic reconfiguration in multihop WDM networks, Journal of High Speed Networks 4 (3) (1995) 221–238.

[25] M. Sridharan, A.K. Somani, M.V. Salapaka, Approaches for capacity and revenue optimization in survivable WDM networks, Journal of High Speed Networks 10 (2) (2001) 109–125.

[26] Y. Xiong, L. Mason, Restoration strategies and spare capacity requirements in self-healing atm networks, IEEE/ACM Transactions on Networking 7 (1) (1999) 98–110.

Randeep Bhatia received the Ph.D. degree in Computer Science from University of Maryland, the M.S. degree in Mathematics and Computer Science from University of Illinois at Chicago and the B.Tech. degree in Computer Science and Engineering from Indian Institute of Technology, Delhi. He is currently with the High Speed Networks Research Department at Bell Labs, Lucent technologies, working on network design, traffic engineering and scheduling algorithms. His current research interests are in the area of Qos for emerging multimedia services in next generation networks.

Murali Kodialam received the Ph.D. degree in operations research from Massachusetts Institute of Technology in 1991 and has been with Bell Labs since. He is currently with the High Speed Networks Research Department, working on resource allocation in networks and performance analysis of communication systems. His current interests are in the areas of new algorithms for robust routing and real-time traffic estimation in IP networks.

T.V. Lakshman received the Master's degree in Physics from the Indian Institute of Science, Bangalore, India, and the Ph.D. degree in Computer Science from the University of Maryland, College Park. He is currently the Director of the High Speed Networks Research Department at Bell Labs, Lucent technologies. His research interests and contributions span a spectrum of networking topics including switch architectures, traffic management, network design, high-speed packet filtering, and TCP performance. He has received several best paper awards from ACM and IEEE, and was an editor of the IEEE/ACM Transactions on Networking from 1996 to 2002. He is a Fellow of the IEEE.