Load Balancing Routing of Fault Tolerant QoS-Guaranteed VPNs

Kartik Gopalan
Binghamton University
kartik@cs.binghamton.edu

Tzi-cker Chiueh
Stony Brook University and
Symantec Research Labs
chiueh@cs.sunysb.edu

Yow-Jian Lin
Telcordia Technologies
yjlin@research.telcordia.com

***Abstract*—As both end-to-end network reliability and performance becomes a growing concern for large distributed organizations, carriers face an increasing pressure to offer enhanced network services with higher Quality of Service (QoS). Such premier services are exemplified by wide-area virtual private networks (VPN) with QoS guarantees, or QVPNs, for which users can specify bandwidth, latency, and reliability requirements. From a carrier's standpoint, the primary challenge is to set up fault-tolerant routes for each QVPN request across its network with user-specified reliability and performance guarantees and, at the same time, maximize the total number of QVPNs that its network can support at any instant. We propose a *Fault Tolerant Load Balancing Routing* (*FTLBR*) algorithm to select fault-tolerant routes for QVPNs. *FTLBR* maintains *network-wide load balance* while selecting primary and backup routes for QVPNs and performs resource sharing along backup routes to achieve an overall high network resource utilization efficiency. *FTLBR* is able to avoid formation of bottleneck links during primary-backup route selection by employing a simple quantitative metric that effectively captures network-wide load balance. Simulation results show that *FTLBR* can support significantly higher number of QVPNs when compared with existing traffic engineering algorithms.**

I. Introduction

Increasing demands on both reliability and performance of network connectivity are driving carriers to offer Virtual Private Network connections with QoS guarantees (QVPNs). Carriers face a tremendous pressure to roll out fault-tolerant QVPNs with millisecond level failover, guaranteed bandwidth, and bounded end-to-end latency to support performance sensitive network traffic. Multi-Protocol Label Switching (MPLS) has emerged as a technology that can meet the reliability and QoS needs of QVPNs. Each QVPN can be mapped onto a primary and one or more backup Label Switched Paths (LSP) that collectively meet its reliability requirement, while reserving sufficient resources per LSP to fulfill its performance requirement. At the same time, the question of how to intelligently route each LSPs in order to maximize the total number of QVPNs on a given physical network remains an open problem. In this paper, we argue that the key to improving network utilization efficiency in the presence of fault-tolerance and QoS constraints is to carefully maintain network-wide load balance and prevent the formation of bottlenecks during route selection.

Typically, QVPN-like services are provided over a wide-area physical network infrastructure administered by a single network carrier. Service level agreement for each QVPN could include any combination of fault-tolerance, bandwidth, and delay bound specifications over one or more physical paths that extend from one point of presence in the carrier's network to another. A network resource provisioning and management system coordinates the set up and maintenance of QVPNs. This system utilizes periodic feedback on the current traffic demands to predict future link loads and selects routes for incoming QVPNs. Since each QVPN is expected to last for several days or even months at a stretch, the time scale of the feedback loop can also be in the same order of magnitude. As a consequence, the run-time measurement feedbacks are more likely to be indicative of future behaviors, and reactions to feedbacks can err on the conservative side without running the risk of being unresponsive.

Traditional hop-by-hop dynamic routing [1], [2] attempts to minimize the network resource usage by forwarding packets towards the destination along the shortest path. This approach tends to under-utilize the links that do not lie along the shortest routes. QoS routing schemes [3]–[13] attempt to find feasible routes for traffic flows with single or multiple QoS constraints. However, their primary focus is to determine a feasible route rather than to optimize for overall network resource usage efficiency. While a number of earlier approaches [3], [14]–[19] have addressed the routing problem from traffic engineering's standpoint, none of them have explicitly attempted to maintain network-wide load balance, presumably because the degree of network-wide load balance for a general mesh-like topology has never been accurately quantified. A quantitative metric is essential to measure and compare the effectiveness of network wide load balance achieved by different routing algorithms.

We present the first application of one such quantitative load balancing metric in selecting fault-tolerant routes for QVPNs with bandwidth and end-to-end delay guarantees. Our proposed algorithm, called *Fault Tolerant Load Balancing Routing* (*FTLBR*), accounts for the impact of selecting a disjoint primary-backup pair on the overall network wide load balance. *FTLBR* determines the importance of each link according to its current residual capacity and expected future load. These per-link importance values are then factored into a combined metric that measures the effective load balance across the network. The strategy in *FTLBR* is to promote the selection of those primary-backup route pairs which consist of less critical links from a load-balancing perspective. Additionally, *FTLBR* also performs *backup resource sharing* – a technique by which multiple QVPNs can share their reservations along backup routes to reduce the cost of providing fault-tolerance and to improve the overall resource usage efficiency. In [20], we introduced a link criticality metric which was used to perform load-balancing primary route selection and briefly outlined its variant that performs primary-backup route selection. In this paper, we describe in detail the *FTLBR* algorithm for primary-backup route selection, and present its comprehensive performance evaluation over three different network topologies. *FTLBR* routes incoming QVPNs without apriori knowledge of future request arrivals. Instead it utilizes current traffic demands from network measurements to predict future link importance and select network paths. Additionally, in order to minimize service disruptions in the absence of faults, *FTLBR* does not re-route operational QVPNs to accommodate new ones.

The rest of the paper is organized as follows. Section II discusses related research. Section III quantifies the metric for network wide load balance to measure the effectiveness of fault-tolerant QVPN routing schemes. Section IV describes the *FTLBR* algorithm. Section V presents the performance comparison of *FTLBR* with two other routing schemes. Section VI summarizes the research results.

II. Related Work

The Widest Shortest Path (WSP) algorithm [4] improves upon SP by selecting the feasible shortest route with maximum residual capacity at the bottleneck link. However, WSP still restricts itself to routes with minimum hops. Several variations [2], [6], [21] perform limited load balancing and congestion management but finally suffer the same fundamental problems of SP routing. QoS routing algorithms provide greater control in tuning network performance. Their primary objective is not to optimize for long-term traffic engineering considerations but to simply find a feasible short-term route that satisfies either single [3]–[6] or multiple [7]–[13] QoS constraints. On the other hand, Traffic Engineering (TE) solutions operate at the granularity of long-term traffic aggregates (QVPNs) and attempt to achieve network-wide resource management objectives. A class of TE schemes [14], [18], [22], [23] manipulate link weights in the standard interior gateway protocols to achieve load balance and minimize congestion for best-effort traffic, but do not support explicit QoS guarantees.

Explicit routing is another TE approach to support QVPN traffic in which the entire route for a QVPN is determined and set up apriori. Minimum Interference Routing Algorithm (MIRA) [15] is an explicit routing approach that addresses the QVPN route selection problem from a traffic engineering perspective. MIRA was one of the first algorithms for provisioning bandwidth guaranteed routes that uses the knowledge of possible ingress-egress node pairs to identify and avoid critical links in the network. The criticality of a link in MIRA depends upon the number of mincuts between the different ingress-egress pairs to which the link belongs. MIRA's notion of link criticality proves useful in avoiding links that impact the network carrying capacity between large number of ingress-egress pairs, but does not identify important links that may not be part of any mincuts. Profile-Based Routing (PBR) [19] is another approach that uses measured profile of past traffic to solve a multi-commodity network flow problem and set up advanced reservations for each traffic class. The advance reservations guide and limit the actual QVPN reservations during online provisioning. TeXCP [24] uses multiple paths to balance network loads and deliver demands from an ingress to an egress router, adaptively moving traffic from over-utilized to under-utilized paths. DEFT [25] permits routers to split traffic on multiple paths, with en exponential penalty on longer paths.

The key feature that distinguishes *FTLBR* from the earlier explicit routing approaches is that it quantifies the notion of network-wide load balance, which is maintained during every step of the route selection process to improve network resource utilization efficiency. The second unique feature of *FTLBR* algorithm is that it provides end-to-end delay guarantees for QVPNs in addition to bandwidth guarantees. None of the prior explicit TE routing schemes provide support for time-sensitive network traffic which is rapidly claiming a significant share of the overall Internet traffic. Finally, from reliability perspective, the *FTLBR* algorithm differs in its ability to provision backup routes to protect QVPNs against any single node or link failure along the primary route, and perform resource sharing along backup routes. This fault-tolerance requirement of QVPNs, in conjunction with bandwidth and delay guarantees, has not been explicitly considered in earlier approaches.

Earlier research in the area of network fault-management can be classified as *reactive vs. proactive* and *local vs. global* schemes. *Reactive* techniques [26] react to a network-fault only when a fault occurs, such as by re-allocating unreserved network resources among the affected QVPNs. While reactive schemes do not need to set aside resources in advance to handle failures, time to recover from a failure could be long especially under heavy loads. Additionally some QVPNs affected by an actual failure may need to be terminated or degraded due to lack of adequate unreserved resources. *Proactive* techniques [27]–[29] pre-allocate network resources in order to provide guarantees and quick recovery from network faults, albeit at the expense of resource usage efficiency. *Local* schemes, such as [30], recover from failures by allocating resources in the neighborhood of the fault whereas *Global* schemes [28] perform more efficient resource allocation in an end-to-end manner. The *FTLBR* algorithm for disjoint primary-backup route selection falls under the category of *proactive and global fault management*. Backup routes, that are disjoint from primary routes, are provisioned at QVPN setup time in an end-to-end manner.

Earliest form of backup resource sharing was proposed in the RAFT approach [28], followed by [31], [32] that deal with bandwidth guarantees alone. More aggressive variations of the backup resource sharing [33], [34] overbook backup reservations for QVPNs to achieve different levels of fault-tolerance, though at the risk that some affected flows might be rejected after a failure. In contrast, *FTLBR*'s application of backup resource sharing provides guaranteed recovery from single failures in a network in the presence of both bandwidth and delay guarantees.

The large body of optical networks research has addressed problems such as lightpath restoration upon failures [35], [36], survivable routing of lightpaths [37] and routing lightpaths to minimize disconnected source-destination pairs [38]. These goals are different from the goal of selecting primary-backup QVPN routes with performance guarantees.

III. Network-Wide Load Balancing Metric

The key intuition behind maximizing the number of admitted QVPNs is to select those routes which best balance the load across different parts of the network and keep the critical network links available for future QVPN requests. This implies that, to the extent possible, the routing algorithm should desist from using links that are critical in carrying traffic between a number of source-destination pairs, so as to prevent the formation of bottleneck links. But how can one determine the importance of a link to the network load balance in the absence of exact knowledge of future QVPN requests? To achieve this goal, this section describes a network-wide load balancing metric, which we first introduced in [20]. The importance of a link in *FTLBR* can be measured by the future *expected load* on each link, i.e. the expected amount of traffic between different source-destination pairs that is carried over this link. A link that is expected to carry a higher amount of traffic between different source-destination nodes is considered more critical than one that is expected to carry less. More formally, let $f_l(s,d)$ be the fraction of all network routes between the source-destination pair (s,d) that pass through link l. Assume that we have knowledge of the expected bandwidth demand $E(s,d)$ between each source-destination pair (s,d) based on measured daily traffic profiles or service-level agreements. The total *expected load* E_l on link l is defined as the fractional sum of expected demands on the link from all possible source-destination pairs in the network. Assuming that the load is equally distributed over each possible route between s and d, then E_l can be expressed as follows.

$$E_l = \sum_{(s,d)} f_l(s,d)E(s,d) \tag{1}$$

The equal-load distribution assumption above may not always hold and we discuss the impact of this assumption in Section IV. One concern in calculating E_l for each link is that the total number of routes between any source-destination pair can be potentially very large. However, typically only a small subset of network nodes are designated as ingress-egress points, which significantly reduces the computation overhead for E_l. Furthermore, our approach can select primary-backup routes from among k-shortest disjoint route pairs, where k is a configurable and small parameter. Thus we can avoid considering overly long routes that may consume excessive resources. This makes it computationally feasible to compute E_l. In addition, $f_l(s,d)$ is completely determined by topology of the network and changes only when the topology changes. Similarly, $E(s,d)$ changes relatively infrequently, such as on a daily basis. Thus the values of $f_l(s,d)$ and E_l can be periodically pre-computed offline and reused multiple times in an online route selection phase.

Let C_l be the total bandwidth capacity of a link and R_l be its residual capacity at any time. To maintain a network-wide load balance, it is essential to first identify a quantitative measure for the degree of network-wide load balance at a given instant. Towards this end, we begin by defining the dynamic cost of each link $e_l = E_l/R_l$ as the expected load per unit of available capacity at link l. Thus, a link with small residual capacity R_l or large expected load E_l is considered more expensive for use in routing QVPNs.

Next, in order to quantify *network-wide load balance*, we must account for the both the *magnitude* of individual link costs and also the *variations* among them. Assume a simple network with two links that have expected loads E_1 and E_2. The term E_l/C_l represents the minimum value of the link cost when the residual capacity is maximum at $R_l = C_l$. For ideal load balance, residual link capacities R_1 and R_2 should evolve towards zero such that (e_1, e_2) indeed stays along the line connecting $(0,0)$ and $(E_1/C_1, E_2/C_2)$. At the same time, the amount of resources consumed at each link along the route should be minimized, because balancing the load at the expense of excessive resource consumption defeats the overall objective of maximizing resource usage efficiency and thus should be avoided. Therefore, it may not always be possible to follow the ideal trajectory and achieve perfect network-wide load balance. Consequently, we attempt to minimize the geometrical distance between the current cost vector (e_1, e_2) and the initial cost vector $(E_1/C_1, E_2/C_2)$ when making a routing decision. This approach ensures that (e_1, e_2) is as close as possible to the ideal load-balance trajectory, while at the same time minimizing the amount of resources consumed along the selected route. Therefore, we define the extent of load balance in a network G as the square of geometrical distance between the current link cost and the initial (idle-state) link cost in an n-dimensional space, where n is the number of links in the network. More formally:

$$B(G) = \sum_{l\in G} \left(e_l - \frac{E_l}{C_l} \right)^2 \tag{2}$$

A smaller value of $B(G)$ translates into a higher network-wide resource usage efficiency.

IV. Primary-Backup Route Selection

In this section we present the *Fault Tolerant Load Balancing Routing* (*FTLBR*) algorithm to select a primary route X_N and a disjoint backup route Y_N for a new QVPN request F_N between a source node s and a destination node d. Both primary and backup routes must be selected so as to maintain network-wide load balance. This goal in turn translates to minimizing the network cost metric $B(G)$. Both the routes X_N and Y_N must satisfy the long-term bandwidth requirement ρ_N and the end-to-end delay requirement D_N. In addition, the backup route Y_N is there to take over the QVPN traffic in case at most one network element (a link or a node) fails along the primary route X_N. The backup route Y_N would provide identical QoS guarantees as the primary route X_N on bandwidth ρ_N and/or end-to-end delay bound D_N. The basic requirement for being able to tolerate any single-element failure is that the primary and backup routes of a QVPN must be completely disjoint with respect to all intermediate network elements. This is

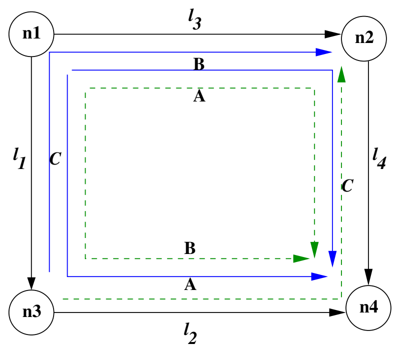

Fig. 1. Resource sharing among the backup paths for three QVPNs A, B, and C. Primary routes are in bold lines and backup routes are in dashed lines.

$Primary(l_2)$	$\{A\}$
$Backup(l_2, n_1)$	$\{C\}$
$Backup(l_2, n_2)$	$\{B\}$
$Backup(l_2, n_3)$	$\{\}$
$Backup(l_2, n_4)$	$\{\}$
$Backup(l_2, l_1)$	$\{C\}$
$Backup(l_2, l_2)$	$\{\}$
$Backup(l_2, l_3)$	$\{B, C\}$
$Backup(l_2, l_4)$	$\{B\}$

Fig. 2. One primary set and eight backup sets at link l_2 in Figure 1.

to ensure complete switch-over to backup route in case of any failure along the primary route. If any network element was to be shared between primary and backup routes, and the shared element happened to fail, then both primary and backup routes would be disabled simultaneously. Failure of source and destination nodes of a QVPN is not covered by the single-failure model since such a failure is fatal to the QVPN by definition. In rest of this section, we first describe the concept of resource sharing along backup routes, then describe *FTLBR* for QVPNs that require bandwidth guarantees alone, and finally QVPNs that require both bandwidth and delay guarantees.

A. Resource Sharing along Backup Paths

A simple model for supporting fault-tolerant routes is to reserve exclusive resources for each QVPN along both primary and backup paths. The backup reservations remain dormant till a failure occurs along the primary path. With such an approach, given that failures might occur infrequently in typical networks, about half the network resources along backup routes can remain tied up and unutilized most of the time. In order to maximize utilization of resources along backup routes, we now describe the notion of *backup resource sharing*. Our goal is to exploit the fact that QVPNs whose primary routes are completely independent of each other can share their resource reservations on the common links along their backup routes. This concept was first proposed in the RAFT approach [28] in the context of flows that require bandwidth guarantees alone. Our approach to backup resource sharing is similar but can be applied to the context of both bandwidth constrained as well as delay-bandwidth constrained QVPNs. We apply the resource sharing concept within the comprehensive admission control and resource allocation framework of the *FTLBR* algorithm.

QVPNs A and B *intersect* with each other at a network element e if both their primary routes pass through element e. Whenever a network element e fails, we need to activate the backup routes for all the QVPNs that intersect at element e. The IDs of all QVPNs whose primary routes pass through a link l are maintained in one *primary set* $Primary(l)$ per link. Further, there are $(m+n)$ *backup sets* per link of QVPN backup reservation, where each set corresponds to one network element; m is the number of links and n is the number of nodes in the entire network. Each backup set at link l is represented by $Backup(l, e)$, $1 \leq e \leq (m+n)$, where each backup set corresponds to one network element e. The backup set $Backup(l, e)$ contains IDs of those QVPNs whose backup routes traverse link l and whose primary routes intersect at the network element e. In other words, $Backup(l, e)$ represents the set of QVPNs whose backup reservations at link l need to be activated in the event that network element e fails.

Figure 1 and Figure 2 illustrates the concept of primary and backup sets. The network consists of 4 nodes and three QVPNs: A (from $n1$ to $n4$), B (from $n1$ to $n4$), and C (from $n3$ to $n2$). The primary route for A (solid line) traverses the route $n1$–$n3$–$n4$ and its backup route (dashed line) traverses a disjoint route $n1$–$n2$–$n4$. Similarly for B and C. Figure 2 gives the primary set and backup sets at link l_2. Link l_2 carries primary route for A and backup routes for B and C. Thus the set $Primary(l_2)$ equals $\{A\}$, $Backup(l_2, n_1)$ equals $\{C\}$, $Backup(l_2, l_3)$ equals $\{B, C\}$ (since primary routes of B and C intersect at l_3), and so on.

Flows whose primary routes do not intersect with each other at any network element do not appear together in any backup set. For instance A and B do not appear together in any backup set. A QVPN whose primary route traverses h links (and hence $h-1$ intermediate nodes) from source to destination, will be part of $(2h-1)$ backup sets at every link along its backup route. For instance, B is part of 3 backup sets at link l_2 since its primary route is 2 hops long and contains 3 network elements (2 links and 1 node) that can potentially fail.

During normal operations, each link scheduler operates with the reservations for QVPNs in its primary set $Primary(l)$. Whenever a network element e fails, its corresponding backup sets $Backup(l, e)$ are activated at all the links l and the respective link schedulers start operating with reservations in primary set $Primary(l)$ plus those in backup set $Backup(l, e)$.

Backup resource sharing affects the manner in which residual link capacities are calculated. If we represent the primary reservation for a QVPN F_i at link j by $\rho_{i,j}$ and its backup reservation at link k by $\beta_{i,k}$, then the residual capacity R_l at link l is calculated as follows.

$$R_l = C_l - \sum_{i \in Primary(l)} \rho_{i,l} - \max_{\forall e} \left\{ \sum_{i \in Backup(l,e)} \beta_{i,l} \right\} \quad (3)$$

In other words, the residual capacity R_l is obtained by deducting the sum of primary reservations and the maximum of the sum of backup reservations in each backup set from the total link capacity C_l. In contrast, without backup resource sharing, the final term would be the sum of backup reservations of *all* flows at link l. Here we exploit the assumption that at most one backup set is active on a link at any time.

B. FTLBR *With Bandwidth Constraints*

Assume that we are provisioning a QVPN F_N that requires a long-term bandwidth guarantee of ρ_N. Selection of a primary-backup route pair (X_N, Y_N) for F_N causes a decrease in the residual capacity of all the links $l \in X_N \cup Y_N$ from R_l to R'_l. The difference $R_l - R'_l$ would be ρ_N for links along the primary route and would range from 0 to ρ_N for links along the backup route, depending upon the degree of backup resource sharing. With smaller residual link capacities, the value of metric $B(G)$ increases. The goal of *FTLBR* is to find the route pair (X_N, Y_N) that produces the least increase in $B(G)$.

FTLBR first eliminate links for which $(R_l - R'_l)$ is negative. Next, it finds the shortest path-pair [39] (as opposed to shortest path if one were selecting the primary route alone), that increases $B(G)$ by the smallest margin. Specifically, *FTLBR* selects the path-pair (X_N, Y_N) with minimum value of $\sum_{l \in (X_N \cup Y_N)} w_l$, where w_l is given by Equation 4 below. (For ease of exposition, we overload the terminology by treating routes X_N and Y_N as *sets* of links along the primary and backup paths respectively.) In computing the shortest path-pair, the per-link weight w_l is defined as the increment in link l's component in $B(G)$ after F_N is admitted.

$$w_l = \left(\frac{E_l}{R'_l} - \frac{E_l}{C_l}\right)^2 - \left(\frac{E_l}{R_l} - \frac{E_l}{C_l}\right)^2 \tag{4}$$

C. FTLBR *With Bandwidth and Delay Constraints*

The shortest path-pair routing algorithms can no longer be applied directly in the case of a QVPN F_N that requires an end-to-end delay guarantee of D_N in addition to the bandwidth guarantee of ρ_N. Presence of end-to-end delay constraints complicates the route selection process since different bandwidth reservations for F_N at a link l can result in different delay budgets. Specifically, the larger the amount of reserved bandwidth, the smaller is the bound on maximum queuing delay experienced by F_N's packets at the link (we assume a rate-based link scheduler such as WFQ [40] to service outgoing packets at every link). Consequently, a given end-to-end delay budget D_N can be partitioned in a number of different ways among the individual links of a route [13], [41], [42]. However, all delay partitioning schemes require advance knowledge of the complete set of links along the candidate routes, whereas the classical shortest path-pair algorithms incrementally construct the route one link at a time. As a result, the traditional shortest path-pair algorithm, which was used for bandwidth constrained *FTLBR*, cannot be used to solve the bandwidth-delay constrained *FTLBR* .

An alternative solution could be to explicitly consider all possible candidate route pairs between the source and destination, perform end-to-end delay partitioning along each candidate, and select the route pair that best maintains the load balance. However, this solution is impractical because the number of route-pairs tends to grow exponentially with increase in network size and connectivity. In order to narrow down the set of candidate route-pairs to a manageable size, *FTLBR* examines only those pairs that have small number of hops. The intuition behind this heuristic strategy is that shorter routes are more likely to consume fewer network resources than longer routes, and consequently are more likely to minimize $B(G)$.

FTLBR for primary-backup route selection has two phases. The first *offline* phase is executed only once to pre-compute $k = k_1 \times k_2$ shortest primary-backup route pairs for all source-destination pair in the network. This is done by first discovering k_1-shortest candidate primary routes from the network graph G using a standard algorithm such as [11]. Next for each candidate primary route X, *FTLBR* discovers k_2 candidate backup routes from the residual network graph that excludes the network elements (links and nodes) along X. The offline phase also pre-computes the expected load E_l for each link based on the computed candidate route pairs.

The *online phase* of *FTLBR* is illustrated in Algorithm 1 and is repeatedly executed whenever a new QVPN request arrives. As input to the online phase, we supply the set of route pairs (X, Y) and E_l values that are pre-computed during the offline phase. For every candidate route pair (X, Y) between s and d that can satisfy the delay-bandwidth requirements (D_N, ρ_N), *FTLBR* partitions the end-to-end delay D_N among the links of both X and Y using one of the delay partitioning algorithms [13], [41], [42]. Delay partitioning assigns a bandwidth reservation value $\rho_{Nl} \geq \rho_N$ for each link $l \in (X \cup Y)$ that guarantees a per-link delay budget D_{Nl} such that $\sum_{l \in X} D_{Nl} \leq D_N$ and $\sum_{l \in Y} D_{Nl} \leq D_N$. Description of specific end-to-end delay partitioning algorithms is beyond the scope of this paper. These are described and compared in greater detail in [41]. Next, *FTLBR* calculates the projected network-wide cost $B(G)$ that would result if the route pair (X, Y) is assigned to F_N. Request F_N is rejected if either no route pair (X, Y) can satisfy F_N's delay-bandwidth requirements or the minimum projected value of $B(G)$ is greater than a cost threshold α. Latter case indicates that admitting F_N would take some parts of the network to a highly resource depleted or imbalanced state that may not be conducive for admitting future QVPN requests. Otherwise, F_N is admitted and assigned to the route pair (X, Y) which causes the least increase in the value of $B(G)$.

D. *Updating Expected Load Values*

In both *FTLBR* algorithms above, every time bandwidth is reserved along a route pair for a new QVPN F_N, the expected load E_l of each link in the network needs to be updated. This is because original E_l values were computed under the assumption of equal traffic distribution among all candidate routes, which may no longer hold once a new QVPN is admitted. For each link l, its E_l value is updated as follows.

$$\begin{aligned} E_l = E_l - f_l(s,d)\rho_N + \rho_{Nl} \quad & \forall l \in X_N \\ E_l = E_l - f_l(s,d)\rho_N + (R_l - R'_l) \quad & \forall l \in Y_N \\ E_l = E_l - f_l(s,d)\rho_N \quad & \forall l \notin (X_N \cup Y_N) \end{aligned} \tag{5}$$

In other words, we subtract the fractional bandwidth demand value $f_l(s,d)\rho_N$ that was expected to pass through each link, and, for links that lie along the selected route pair (X_N, Y_N) we add in the new actual reservation. For links along the primary route X_N, the actual reservation is ρ_{Nl}. In the case of

Algorithm 1 The FTLBR *algorithm* selects both the primary and backup routes for a QVPN F_N between source s and destination d. F_N requires an end-to-end delay bound of D_N and bandwidth of ρ_N.

```
1: Input: 1. New QVPN specification $(s, d, D_N, \rho_N)$
2:          2. Each link's expected load $E_l$, residual capacity $R_l$
3:             and total capacity $C_l$.
4:          3. Set $S$ of $k$ candidate primary-backup route pairs $(X, Y)$.
5: Output : Primary route $X_N$ and backup route $Y_N$
6:
7: for all links $l$ do
8:     $e_l = \frac{E_l}{R_l}$
9: end for
10: $B_{min} = \infty$;
11: for each route pair $(X, Y) \in S$ that satisfies $(D_N, \rho_N)$ do
12:     Partition $D_N$ independently along both $X$ and $Y$.
13:     Recompute resulting residual link capacities $R'_l$.
14:     Recompute the link costs $e_l = \frac{E_l}{R'_l}$.
15:     Recompute the network-wide metric $B(G)$.
16:     if $B(G) < B_{min}$ then
17:         $B_{min} = B(G)$;$X_N = X$; $Y_N = Y$.
18:     end if
19: end for
20: if $(B_{min} > \alpha)$ then
21:     Reject $F_N$
22: else
23:     Select $(X_N, Y_N)$ as primary-backup route pair for $F_N$.
24: end if
```

bandwidth constrained *FTLBR*, $\rho_{Nl} = \rho_N$ for all links $l \in X_N$. On the other hand, in the case of bandwidth-delay constrained *FTLBR* , we partition the end-to-end delay into per-link delay constraints and, as a result, each link l along X_N and Y_N can potentially have a different bandwidth reservation ρ_{Nl}, where $\rho_{Nl} \geq \rho_N$. For links along the backup route Y_N, the actual reservation $(R_l - R'_l)$ in both cases ranges between 0 and ρ_{Nl}, depending upon the degree of backup resource sharing at l. R_l and R'_l are computed according to Equation 3, respectively before and after the making reservation for F_N. The basic idea is to update all E_l values to reflect each link's current expected load as QVPNs are actually routed. In the end, after setting up all feasible routes, the most utilized links will have largest (updated) E_l.

V. Performance Evaluation

This section compares *FTLBR* against two earlier traffic engineering based approaches that were originally proposed for bandwidth constrained primary route selection – the Widest Shortest Path (WSP) [4] based algorithm and the Minimum Interference Routing Algorithm (MIRA) [15]. The original WSP algorithm selects a primary route which has maximum residual bottleneck link capacity from among all the feasible routes having minimum length. A *feasible* route is one that satisfies the QoS constraints of the QVPN being admitted. The primary-backup route selection version, that we call FTWSP, selects the feasible route pair with minimum-length primary path that has maximum bottleneck link capacity. If multiple route pairs have minimum-length primary routes with equal maximum bottleneck capacity, then the same selection criteria is applied along the dimension of backup routes. For bandwidth-delay constrained route selection, FTWSP applies load balancing criteria of WSP in the context of k-shortest paths framework i.e., from among k candidate route pairs, the minimum length candidate with maximum bottleneck link capacity is chosen. In MIRA, the weight of a link is given by the number of source-destination pairs whose mincuts include the link l. The intuition is to defer loading those links which are part of a large number of mincuts, and thus are critical for a large number of source-destination pairs. For bandwidth constrained route selection, the primary-backup variant of MIRA (which we call FTMIRA) selects the route pair with minimum sum of link weights. For bandwidth-delay constrained route selection, FTMIRA applies the load balancing criteria of MIRA in the context of k-shortest paths framework by examining a set of k candidate route pairs and selecting the one with minimum sum of link weights.

Simulations are performed using three different network topologies. The first topology is the North American IP backbone topology of Sprint[1] which consists of 36 nodes and 69 links. The second topology is the AT&T CERFnet[2] nationwide backbone infrastructure with 41 nodes and 64 links. The third topology is a regular 5x5 Grid topology with 25 nodes and 40 links. Link capacities range from 45 Mbps to 200 Mbps and sources-destination pairs are selected uniformly. Bandwidth demand profile $E(s, d)$, is uniformly random between 45 Mbps and 100 Mbps. New QVPNs request a bandwidth guarantee of 100Kbps average rate, and when required, an end-to-end delay guarantee of 60ms, with 5Kbits burst size. QVPN reservations that are provisioned once stay in the network forever. QVPNs are admitted to the network till the network reaches a state where no more QVPNs can be admitted between any ingress-egress node pair. Intra-path delay partitioning is performed using the Load-based Slack Sharing algorithm [41] and Weighted Fair Queuing's (WFQ) [40] bandwidth-delay relationship is used.

A. Number of Admitted QVPNs

The most direct performance measure is the number of QVPNs admitted by each algorithm under identical topology and workloads. Figure 3 plots the number of QVPNs admitted with bandwidth constraints alone by different algorithms. Figure 4 plots the same for QVPNs with bandwidth-delay constraints. In each run, a different random number seed controls the combination of link capacities used. *FTLBR* consistently admits more QVPNs than WSP and MIRA algorithms since its routing decisions are guided explicitly by a network-wide load balancing metric, whereas WSP and MIRA attempt limited load balancing through indirect metrics. MIRA considers the ingress-egress information to calculate link criticality. In cases where MIRA admits fewer QVPNs, the links along the mincut between ingress-egress pairs do not faithfully represent the most critical links in the network, thus defeating its criticality metric. This behavior has also been observed in [19]. In the presence of bandwidth-delay constraints, *FTLBR* shows a wider margin of performance improvement than with bandwidth constraints alone. This is due to the fact that tight delay constraints of F_N require bandwidth reservation ρ_{Nl} at each

[1]http://www.sprintworldwide.com/english/maps/northamerica.pdf
[2]http://www.cerf.net/about/images/bbone-large.jpg

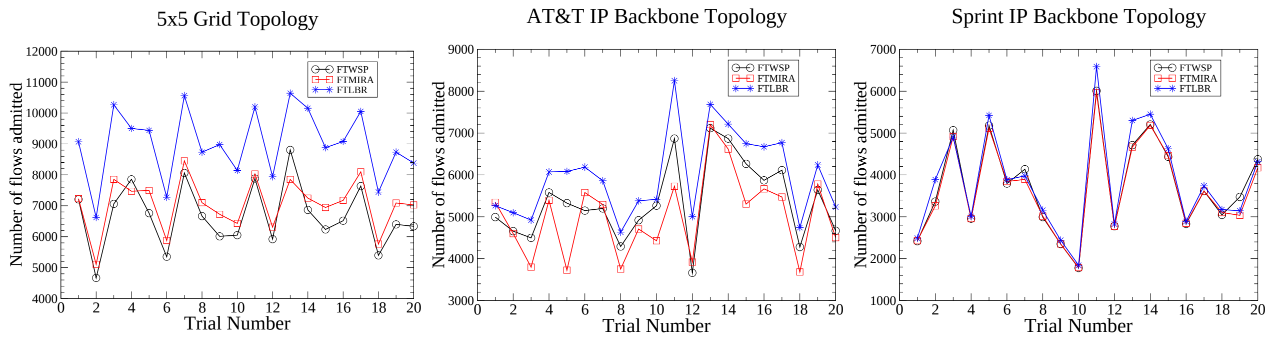

Fig. 3. Number of QVPNs admitted with bandwidth guarantees alone. $\rho_i^{avg} = 100Kbps$, $k = 15$, distance= 5.

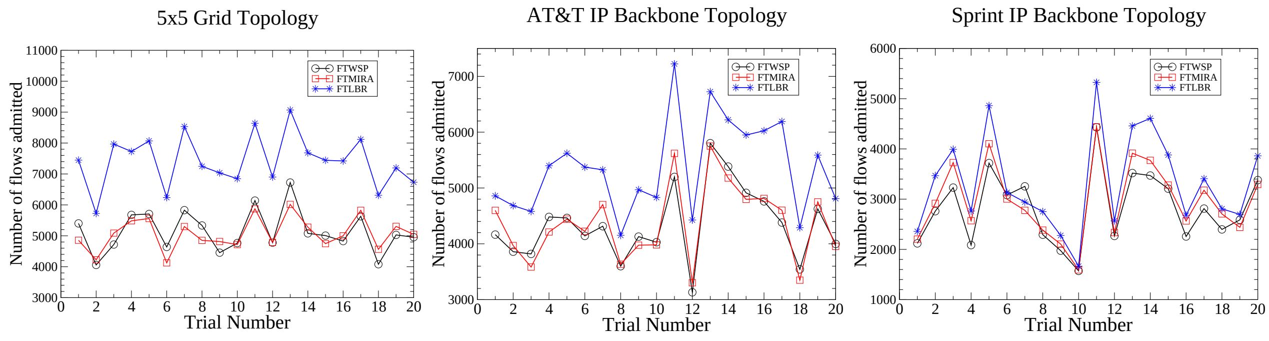

Fig. 4. Number of QVPNs admitted with bandwidth-delay guarantees. $\rho_i^{avg} = 100Kbps$, $D_i = 60ms$, $k = 15$, distance= 5.

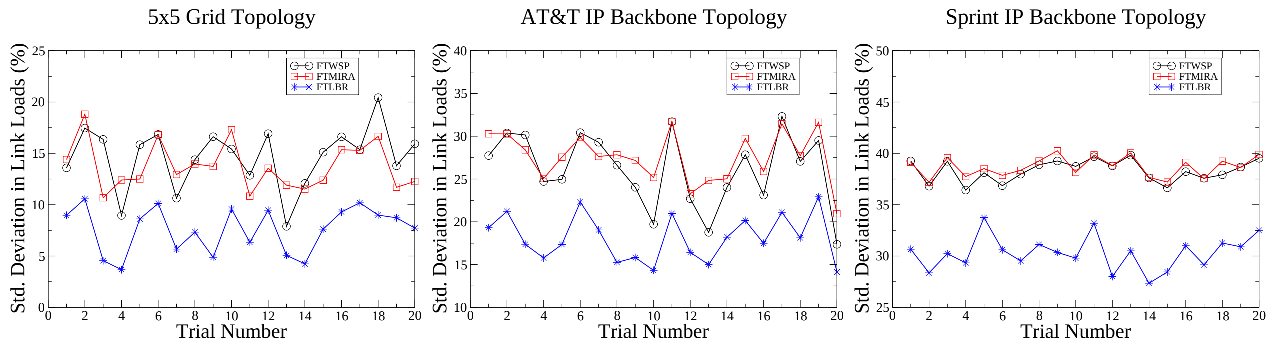

Fig. 5. Comparison of standard deviation in link loads. $\rho_i^{avg} = 100Kbps$, $D_i = 60ms$, $k = 15$, distance= 5.

link l that is larger than its average bandwidth requirement ρ_N. Thus with every provisioned delay-constrained QVPN, there is a tendency towards higher network-wide load imbalance and the benefits from *FTLBR*'s load balancing approach become more evident. Finally, *FTLBR*'s relative performance is much better with AT&T and 5 $\times$ 5 Grid topologies than with Sprint topology, since the former two topologies offer a wider choice of alternative routes between each source and destination.

B. Extent of Load Balance

To show that *FTLBR* indeed maintains better network-wide load balance, Figure 5 plots the standard deviation in final percentage loads among all the links in the network for each of 20 simulation runs. The load deviation is consistently smaller for *FTLBR* than the other algorithms because *FTLBR* explicitly tries to minimize the load differences across all network links.

To illustrate this further, Figure 6 plots the evolution of the network cost metric $B(G)$ in one simulation run with arrival of each successive QVPN request having bandwidth-delay constraints. Whenever a QVPN is admitted, $B(G)$ increases; when a QVPN is rejected, $B(G)$ stays constant. Recall that $B(G)$ represents the vector distance of the current link costs from the idle-state link costs in the network. The figures show that, with successive QVPN requests, *FTLBR* algorithm maintains a lower value of $B(G)$ compared to WSP and MIRA algorithms. While the $B(G)$ values of WSP and MIRA saturate after admitting a smaller number of QVPNs, *FTLBR* continues admitting QVPNs for a longer duration due to better network-wide load balance.

C. Impact of Candidate Set Size

An important factor that determines the performance of different algorithms is the size k of the candidate set of routes pairs that are examined during route selection. Figure 7 shows that with increasing candidate set size k, the number of QVPNs admitted with bandwidth-delay constraints increases initially for all the three algorithms. In general, larger value of k implies more candidate choices for selecting a route leading to more admitted QVPNs. However, the number of admitted routes tends to saturate for large k since the candidate set begins to include longer routes which consume more network resources. Consequently, these longer candidate routes are rarely selected.

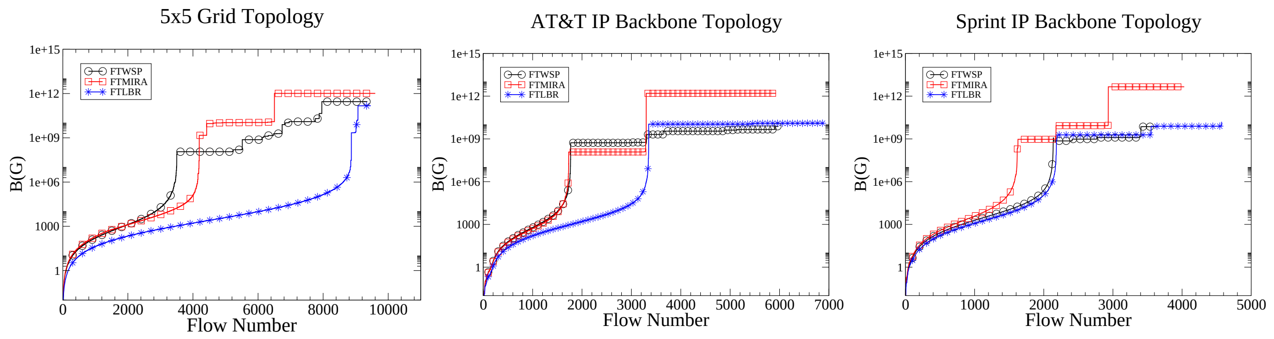

Fig. 6. Evolution of network-wide load balance metric with successive route selections. $\rho_i^{avg} = 100Kbps$, $D_i = 60ms$, $k = 15$, distance= 5.

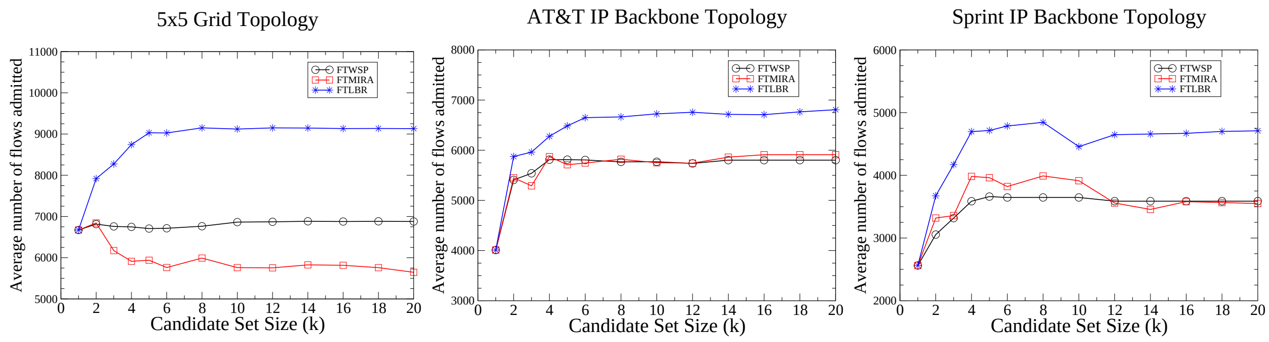

Fig. 7. Number of QVPNs admitted vs. candidate set size k. $\rho_i^{avg} = 100Kbps$, $D_i = 60ms$, distance= 5.

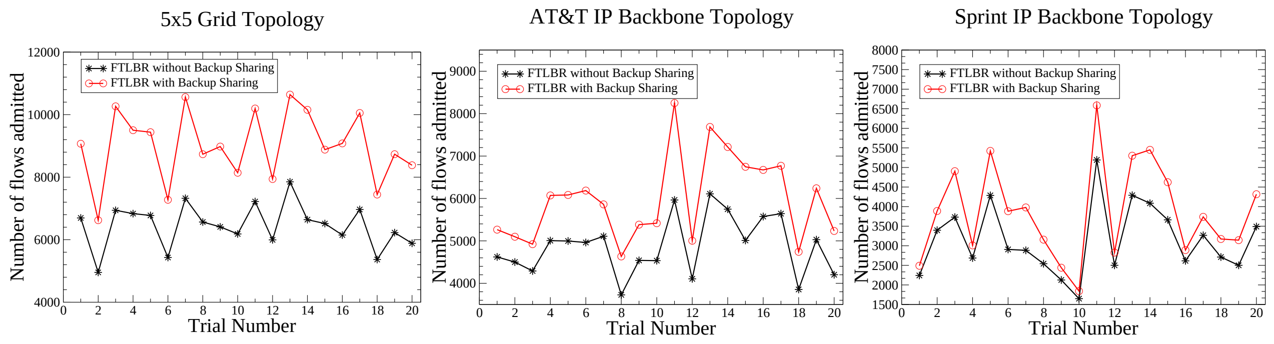

Fig. 8. Improvement in *FTLBR* due to backup resource sharing. $\rho_i^{avg} = 100Kbps$, $k = 15$, distance= 5.

With increasing k, *FTLBR* algorithm admits significantly higher number of QVPNs than the other two algorithms, since it tries to select the candidate route which best maintains network-wide load balance. *FTLBR* also produces the maximum improvement in number of admitted QVPNs with initial increase in k since it has a greater choice of candidates whose $B(G)$ values can be compared for network-wide load balance. WSP is least affected by increase in k since it always selects the shortest feasible route with widest bottleneck capacity. WSP does not benefit from a larger candidate set, which helps only if it includes more viable route pairs with shortest path length. MIRA benefits from a larger k only if it produces more candidate route pairs that contain fewer links that affect the maxflows values.

D. Effectiveness of Backup Resource Sharing

In this section, we evaluate the extent of gain obtained from backup resource sharing. Figure 8 shows the extent of gain obtained by *FTLBR* with the use of backup resource sharing. In all cases, *FTLBR* with resource sharing admits more QVPNs than the one without backup resource sharing. The extent of performance gain, however, varies across different topologies, with the Grid topology showing maximum improvement, followed by AT&T and Sprint. The reason for higher level of improvement with Grid and AT&T topologies is the correspondingly higher level of connectivity among their nodes. Consequently smaller proportion of primary routes intersect with each other and, in the process, permit greater proportion of backup reservations to be shared. Sprint topology has sparser connectivity and thus greater intersection among primary routes, and enables less sharing among backup paths. Grid topology shows most improvements due to its rich connectivity, providing a large number of non-intersecting alternatives for each primary route.

VI. Conclusions

Customers increasingly expect their network carriers to support fault-tolerant VPNs with stronger guarantees on bandwidth and delay bounds. Network carriers thus face the twin challenges of satisfying the QoS requirements of each VPN (or QVPN), and maximizing the number of QVPNs they can support simultaneously. In this paper, we have argued that maintaining network-wide load balance in the route selection process is the key to maximizing network resource usage efficiency under reliability and performance constraints. Maintaining network-wide load balance makes it less likely that some critical links may exhaust their capacity long before others, rendering network resources fragmented and unusable. Specifically, we have proposed the *Fault Tolerant Load Balancing Routing* (*FTLBR*) algorithm, which selects primary and backup routes for wide-area QVPNs with fault-tolerance, bandwidth, and end-to-end delay guarantees. *FTLBR* is the first algorithm to employ an explicit quantitative metric for network-wide load balance to select fault-tolerant routes for QVPNs. *FTLBR*'s metric promotes the selection of route pairs that include less critical links from the standpoint of long-term load balancing. Our simulation results show that *FTLBR* can support significantly higher number of QVPNs under the same input workloads, when compared against fault-tolerant variants of the existing traffic engineering routing algorithms.

Acknowledgment: This work is supported in part by the following NFS awards: ACI-0234281, CNS-0435373, and CCF-0541096.

REFERENCES

[1] C. Hedrick, "Routing information protocol," RFC 1058, June 1988.

[2] J. Moy, "OSPF Version 2," RFC 2328, April 1998.

[3] D.O. Awduche, L. Berger, D. Gan, T. Li, G. Swallow, and V. Srinivasan, *Extensions to RSVP for LSP Tunnels*, Internet Draft, Sept. 1999.

[4] R. Guerin, A. Orda, and D. Williams, "QoS routing mechanisms and OSPF extensions," in *Proc. of IEEE GLOBECOM'97, Phoenix, AZ*, Nov. 1997, vol. 3, pp. 1903–1908.

[5] K. G. Shin and C.-C. Chou, "A distributed route selection scheme for selecting real-time channel," in *Proc. of Sixth Intl. Conf. on High Performance Networking*, Sept. 1995, pp. 319–329.

[6] S. Plotkin, "Competitive routing of virtual circuits in ATM networks," *IEEE J. Selected Areas in Comm.*, vol. 13(6), pp. 1128–1136, 1995.

[7] R. Hassin, "Approximation schemes for restricted shortest path problem," *Mathematics of Operations Research*, vol. 17(1), pp. 36–42, February 1992.

[8] S. Chen and K. Nahrstedt, "On finding multi-constrained paths," in *Proc. IEEE ICC'98*, June 1998.

[9] Z. Wang and J. Crowcroft, "Bandwidth-delay based routing algorithms," in *Proc. of IEEE GLOBECOM'95, Singapore*, Nov. 1995, pp. 2129–2133.

[10] D. Raz and Y. Shavitt, "Optimal partition of QoS requirements with discrete cost functions," in *Proc. of INFOCOM 2000, Tel Aviv, Israel*, March 2000.

[11] D. Eppstein, "Finding the *k* shortest paths," in *Proc. of the 35th Annual Symposium on Foundations of Computer Science*, November 1994, pp. 154–155.

[12] F. Ergun, R. Sinha, and L. Zhang, "QoS routing with performance dependent costs," in *Proc. of INFOCOM 2000, Tel Aviv, Israel*, March 2000.

[13] D.H. Lorenz, A. Orda, D. Raz, and Y. Shavitt, "Efficient QoS partition and routing of unicast and multicast," in *Proc. of IWQoS 2000, Pittsburgh, PA*, June 2000, pp. 75–83.

[14] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. of IEEE Infocom 2000, Tel Aviv, Israel*, March 2000, pp. 519–528.

[15] M.S. Kodialam and T.V. Lakshman, "Minimum interference routing with applications to MPLS traffic engineering," in *Proc. of INFOCOM'2000, Tel Aviv, Israel*, March 2000, pp. 884–893.

[16] N. Shen and H. Smit, "Calculating IGP routes over traffic engineering LSPs," Internet Draft, June 1999.

[17] H. Smit and T. Li, "IS-IS extensions for traffic engineering," Internet Draft, August 2003.

[18] A. Sridharan, R. Guerin, and C. Diot, "Achieving near optimal traffic engineering solutions in current OSPF/ISIS networks," in *Proc. of IEEE Infocom 2003, San Fransisco, CA*, March 2003.

[19] S. Suri, M. Waldvogel, D. Bauer, and P.R. Warkhede, "Profile-based routing and traffic engineering," *Computer Communications*, vol. 26(4), pp. 351–365, 2003.

[20] K. Gopalan, T. Chiueh, and Y.J. Lin, "Network-wide load balancing routing with performance guarantees," in *Proc. of ICC 2006, Istanbul, Turkey*, June 2006.

[21] S. Blake, D. Black, D. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," RFC 2475, Dec. 1998.

[22] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS weights in a changing world," *IEEE Journal on Selected Areas in COmmunications*, vol. 20, pp. 756–767, May 2002.

[23] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "Overview and principles of internet traffic engineering," RFC 3272, May 2002.

[24] Srikanth Kandula, Dina Katabi, Bruce Davie, and Anna Charny, "Walking the Tightrope: Responsive Yet Stable Traffic Engineering," in *ACM SIGCOMM*, Philadelphia, PA, August 2005.

[25] D. Xu, M. Chiang, and J. Rexford, "DEFT: distributed exponentially-weighted flow splitting," in *Proc of IEEE INFOCOM'07*, May 2007.

[26] A. Banerjea, C. Parris, and D. Ferrari, "Recovering guaranteed performance service connections from single and multiple faults," in *Proc. of IEEE GLOBECOM'94, San Francisco, CA*, Nov. 1994, pp. 162–168.

[27] A. Banerjea, "Simulation study of the capacity effects of dispersity routing for fault-tolerant real-time channels," in *Proc. of ACM SIGCOMM'96*, Oct. 1995, vol. 26(4), pp. 194–205.

[28] K. Dovrolis and P. Ramanathan, "Resource aggregation for fault tolerance in integrated services packet networks," *ACM Computer Communications Review*, vol. 28(2), pp. 39–53, April 1998.

[29] N. Taft-Plotkin, R. Ogier, and B. Bellur, "Quality-of-service routing using maximally disjoint paths," in *Proc. of IWQoS, London, UK*, June 1999.

[30] Q. Zheng and K.G. Shin, "Fault-tolerant real-time communication in distributed computing systems," in *Proc. of 22nd Intl. Symposium on Fault-Tolerant Computing*, July 1992, pp. 86–93.

[31] K. Kar, M. Kodialam, and T.V. Lakshman, "Routing restorable bandwidth guaranteed connections using maximum 2-route flows," in *Proc of IEEE INFOCOM'02*, June 2002.

[32] M. Kodialam and T.V. Lakshman, "Dynamic routing of bandwidth guaranteed multicasts with failure backup," in *Proc of IEEE ICNP'02*, Nov 2002.

[33] S. Han and K.G. Shin, "Fast restoration of real-time communication service from component failures in multihop networks," in *Proc. of ACM SIGCOMM'97*, 1997, pp. 77–88.

[34] S. Kim, D. Qiao, S. Kodase, and K.G. Shin, "Design and evaluation of routing schemes for dependable real-time connections," in *Proc. of Intl. Conference on Dependable Systems and Networks (DSN'01), Goteborg, Sweden*, July 2001.

[35] S. Ramamurthy and B. Mukherjee, "Survivable WDM mesh networks: Part II – Restoration," in *Proc. of IEEE International Conference on Communications (ICC '99), Vancouver, Canada*, June 1999.

[36] A. Fumagalli, I. Cerutti, M. Tacca, F. Masetti, R. Jagannathan, and S. Alagar, "Survivable networks based on optimal routing and WDM self-healing rings," in *Proc. of INFOCOM'99, New York, NY*, March 1999, pp. 726–733.

[37] E. Modiano and A. Narula-Tam, "Survivable routing of logical topologies in WDM networks, anchorage, alaska," in *Proc. of IEEE INFOCOM 2001*, April 2001, pp. 348–357.

[38] O. Crochat and J.Y.L. Boudec, "Design protection for WDM optical networks," *IEEE Journal on Selected areas in Communication*, vol. 16(7), pp. 1158–1166, Sept. 1998.

[39] J.W. Suurballe and R.E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, pp. 325–336, 1984.

[40] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proc. of IEEE*, vol. 83(10), pp. 1374–1396, October 1995.

[41] K. Gopalan and T. Chiueh, "Delay budget partitioning to maximize network resource usage efficiency," in *Proc. IEEE INFOCOM'04, Hong Kong, China*, March 2004.

[42] R. Nagarajan, J. Kurose, and D. Towsley, "Local allocation of end-to-end quality-of-service in high-speed networks," in *Proc. of 1993 IFIP Workshop on Perf. analysis of ATM Systems, North Holland*, Jan. 1993, pp. 99–118.