Routing Bandwidth Guaranteed Paths with Restoration in Label Switched Networks

Samphel Norden [*]
Dept. of Computer Science
Washington University in St.Louis
samphel@arl.wustl.edu

Milind M. Buddhikot
Lucent Bell Labs
Center for Networking Research
milind@dnrc.bell-labs.com

Marcel Waldvogel
IBM Zurich Research Laboratory
mwl@zurich.ibm.com

Subhash Suri[†]
Dept. of Computer Science
University of California, Santa Barbara
suri@cs.ucsb.edu

Abstract

Label switched networks have become increasingly attractive to both network providers and customers. By creating aggregate, bandwidth-reserved flows, these networks offer routing flexibility, predictable bandwidth usage, and quality-of-service (QoS) provisioning. This flexibility in routing enables fault-persistent QoS reservations, where connectivity and allotted bandwidth remains available, even if some links or network nodes fail. The automatic switch-over from a now-defunct path to a new, working path is known as restoration. Restoring bandwidth-guaranteed paths requires allocation of resources on backup paths that will be used in the event of faults. In this paper, we investigate distributed algorithms for routing with backup restoration. Specifically, we propose a new concept of Backup Load Distribution Matrix that captures partial network state, greatly reducing the amount of routing information maintained and transmitted while achieving efficient bandwidth usage. We present and simulate two new distributed routing algorithms, which provide significant improvements in rejection rates and provide substantial savings in bandwidth used and call setup time compared to existing algorithms.

1 Introduction

The concept of Label Switching has its roots in virtual-circuit networks. Using a simple label to base switching decisions on became widely popular with the advent of Asynchronous Transfer Mode (ATM) technology. It has since been extended and adapted to a wide range of networks, including optical networking technologies such as wavelength switching and new electronic packet switching technologies such as Multi-Protocol Label Switching (MPLS). These networks offer connectivity to their clients in the form of an end-to-end Label Switched Path (LSP) with bandwidth and/or delay guarantees.

When uninterrupted network connectivity is necessary, a client may use LSPs from two (or more) Network Service Providers (NSP) to deal with occasional network failures. However, this requires multiple physical connections (ports) to different NSPs and wastes (at least) half the bandwidth. To avoid this, an NSP may provide an enhanced service with additional guarantees: for every client request, the NSP sets up two LSPs, one primary LSP that is used under normal circumstances, and a backup LSP that is activated in the event of disruption of the primary path due to link or switch failures. It is possible to conceive a service wherein k backup paths are setup for every primary path to protect against potentially $k-1$ simultaneous link failures. As the chances for link or switch outages are already very low, the probability of multiple simultaneous outages is much lower. Therefore, in our work, we focus only on a single backup path for every primary path.

The mechanism used for detection of path disruption and switch-over to backup path has two variants:

(1) *Backup path with protection:* In this case, in the event of a link failure, the endpoints of the path detect path disruption and switch to the secondary path. This however requires hardware support on router/switch ports to detect link failures. (2) *Backup path with restoration:* In this case, secondary path is configured only in the event of disruption of primary path. The failure detection and subsequent signaling to activate the backup path is performed using out-of-band mechanisms implemented in control software.

Note that in both cases, resources are always allocated on primary and backup path. However, in the first case, backup path is always active and always consumes resources. In the

[*] Bell Labs Summer Intern.

[†] Supported in part by NSF grants ANI 9813723 and CCR-9901958.

second case, one can conceivably use resources on backup paths for improving performance for best-effort traffic. In this paper, we focus on the problem of routing bandwidth guaranteed paths with restoration in generic label switching networks.

Clearly, compared to the scenario where the client buys two independent paths, backup restoration provisioned by the service provider improves the overall network utilization.

1.1 Research Contributions

In this paper, we investigate distributed algorithms for routing of bandwidth guaranteed LSPs with backup restoration in the context of static and dynamically reconfigurable label switched IP (MPLS) and optical networks. We propose a new concept of Backup Load Distribution (BLD) matrix that captures partial network state and eliminates problems of bandwidth wastage found in algorithms reported in the literature. We describe two new distributed routing algorithms that utilize the BLD matrix and run in bounded time. The proposed BLD matrix can be exchanged among peer routers or switches using the OSPF extensions for Quality-of-Service (QoS) routing [2] and therefore, our algorithms can be realized in the existing internet architecture. Our simulation results for sample network topologies show 50% reduction in the number of rejected requests and 30-40% savings in total bandwidth used for backup.

1.2 Outline

The outline of the rest of this paper is as follows: Section 2 presents the background material for the remaining discussion in the paper. Section 3 describes the limitations of using partial network state information consisting of only three state variables per link, namely *residual bandwidth*, *bandwidth for primary paths*, and *bandwidth for backup paths*. The concept of BLD matrix that eliminates these limitations is introduced in Section 4. In Section 5, we then describe our two new algorithms that use the BLD matrix. Section 6 describes our simulation experiments with some realistic network topologies. Finally, Section 7 presents our conclusions.

2 Background

In this section, we will present relevant background material on various aspects of the problem of routing bandwidth guaranteed backup paths.

2.1 Characteristics of Routing Algorithms

Our routing schemes are online as they route a new path request based only on the knowledge of the *current* state of the network and do not exploit properties of the *future* requests. Also, our routing schemes need knowledge of only partial or subset of the total network state. One can design schemes with varying degrees of partial state. Lakshman et al. [7] describes one such partial information scenario wherein for every link L with capacity C_L, three state variables are maintained and exchanged among peering routers: (1) $\boldsymbol{F_L}$: Amount of bandwidth used on link L by all primary paths that use link L. (2) $\boldsymbol{G_L}$: Amount of bandwidth used by all backup paths that contain link L. (3) $\boldsymbol{R_L}$: Residual capacity on the link L defined as $C_L - (F_L + G_L)$. Our algorithms use these three per-link state variables and a new form of state called Backup Load Distribution matrix.

2.2 Fault Model

In the context of protected path routing it is important to consider two kinds of failures, namely link failures and router failures. A common fault model for link failures assumed in literature and justified by network measurements [8] is that at any given time only one link in the network fails. In our work, we use this failure model to devise our algorithms. Note that a simple but wasteful way to deal with multiple link failures is to reserve multiple backup connections. For example, if we assume a maximum two simultaneous link failures, reserving two backups for every primary will guarantee uninterrupted connectivity in the worst case when two out of three reserved paths fail. However, if majority of the times, two links that fail simultaneously belong to the same path, the second backup may be a waste.

Unlike telephone switches, modern routers still do not support five or seven nine reliability and therefore, in modern IP networks, unlike phone networks, router failures may be more frequent than link failures. Clearly, if primary and backup paths are routed to account for a single node and/or link failure, they must be link and node disjoint. A easy way to achieve this is to model router failure as a link failure. This technique, often used in distributed systems, represents a router by two nodes connected by a link with infinite capacity. The router failure is then simulated by a failure on this infinite-bandwidth link. In our work, we again assume that at any given time only one link or router fails.

2.3 Backup Path Sharing

The concept of backup path sharing exploits the fault models described above and is central to efficient routing of backup paths. The fault model of single-link failure guarantees that in the event of a link failure, if two primary paths are link disjoint, they won't fail simultaneously. In this case, the backup paths for these primary paths can share all (or fewer) links, since both backup paths will never be active at the same time. In other words, if two LSPs, each with bandwidth requirement of b units, are routed on link-disjoint paths, they can use the same backup path with capacity b. Such bandwidth sharing allows one of the two primary paths to use the backup "for free." The goal of the routing algorithm therefore is to maximize backup path sharing to maximize usable bandwidth.

The amount of sharing that can be achieved by an online algorithm over a series of N requests can depend on the amount of state information at its disposal. Limited amount of state information can lead to pessimistic link selection and increased request rejection.

2.4 Widest Open Shortest Path First

The Widest Shortest Path First (WSPF) algorithm was first proposed by Guerin et al. [3] for primary path routing of bandwidth guaranteed paths. Since our schemes use this algorithm, we will present it briefly.

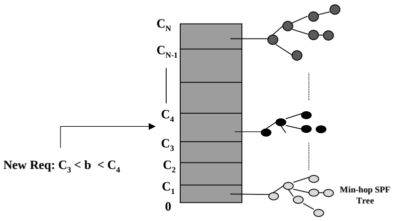


Figure 1. WSPF data structures

The drawback of traditional Shortest Path First (SPF) algorithm is that it yields an optimal solution for a single request, however, over a span of N requests, it can lead to high request rejection and low network utilization [3, 7]. The WSPF algorithm alleviates this problem by selecting a shortest path with maximum (*"widest"*) residual capacity on its component links. To minimize overheads of computing the shortest path and distributing the state information in a distributed implementation, Guerin at al. propose two ideas (Figure 1): (1) ***Quantization:*** Quantize the bandwidth on a link into a fixed set of ranges or bins. When a new path request is received, the request is mapped to a fixed bin and can be satisfied by selecting a path with links that belong to that or a higher bin. (2) ***Pre-computation:*** For each quantization level or bin, store a pre-computed SPF tree from every source edge router to all destination edge routers. Note that every time the residual bandwidth on a link changes a quantization level, the SPF tree for the new level and the old level need to be recomputed. The complexity of the WSPF pre-computation for k bandwidth levels and a network of n nodes, m links in the worst case is $O(kmn \log\ n)$.

A drawback of WSPF is that it does not account for knowledge of nature of traffic between ingress-egress pairs. We choose WSPF as a good tradeoff between simplicity and performance.

3 Limitations of Using Partial Network State with Three Variables per Link

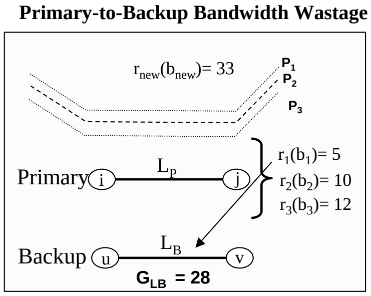


Figure 2. Primary-to-Backup Link Wastage

In the following, we will show that the use of three state variables (R_L, F_L, G_L) per link L leads to what we call *Primary-to-Backup link* bandwidth wastage. Such wastage may cause pessimistic link selection in backup path computation and therefore higher request rejection and reduced bandwidth sharing.

In order to provide bandwidth guarantees, the use of partial network state information leads to the assumption that the *worst case load (maximum load on a link)* over all links in the primary path will be backed up on the backup path. However, in most cases, a link in a backup path may only support a part of the load of the primary path, since the primary path load is potentially distributed over a set of backup paths. But in the partial state information scenario, only the worst case values are known, as there is no information on the distribution of the primary path load on different backup links. Thus, there is a pessimistic assumption as to the amount of free shareable bandwidth on a link belonging to a backup path.

We illustrate this concept with an example shown in Figure 2. Consider link L_p between nodes i, j. Three existing primary paths P_1, P_2, P_3 routed for requests r_1, r_2, r_3 with bandwidth requirements $b_1 = 5, b_2 = 10, b_3 = 12$ use this link which results in a load of $F_{L_p} = 27$ units due to the primary path. Let us assume that the new request to be routed r_{new} requires $b_{new} = 33$ units of bandwidth. The backup path routing is trying to evaluate suitability of link L_B between nodes u, v as a member of the backup path. Let us further assume that only request r_1 is using the link $L_B = (u, v)$ on its backup path. Also, let the current load on L_B induced by backup paths be $G_{L_B} = 28$ units with a residual capacity of $R_{L_B} = 12$.

If we assume the use of *complete network state* information, then, the routing algorithm knows that of the primary path load F_{L_P}, only the primary path for r_1 is backed up on a path that uses link L_B. Therefore, out of $G_{L_B} = 28$, only 5 units is induced by link L_P and an extra 23 units of bandwidth already reserved is available for backing up the new request (free shareable bandwidth). Since, $R_{L_B} = 12 > ((b_{new} = 33) - 23) = 10$, the complete information case will allow the selection of link L_B in the backup path.

On the contrary, in the partial information scenario, only the absolute $F_{L_P}, G_{L_B}, R_{L_B}$ values are known and the algorithm does not know the distribution of F_{L_P} on link L_B. This forces a pessimistic assumption that in the event of failure of link L_P, not $b_1 = 5$ but $b_1 + b_2 + b_3 = 27$ units may need to be backed up on L_B. Clearly, the sum of the free shareable bandwidth and the residual capacity on L_B, $(G_{L_B} - backup + R_{L_B}) = (28 - 27 + 12) = 13$, is less than the new request size 33, and therefore, L_B will not be selected as a potential link in the backup path. In other words, lack of extra information can lead to a disconnected graph on which backup path is routed and cause the request to be rejected. We call this kind of bandwidth wastage and request rejection as *Primary-to-Backup link* wastage.

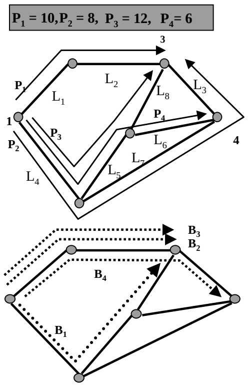
4 Backup Path Routing using Backup Load Distribution Matrix

In this section, we describe a new form of state information called *Backup Load Distribution (BLD)* matrix based on the concept of backup sharing [7] and illustrate how we can employ it to achieve better backup path sharing.

4.1 Concept of Backup Load Distribution (BLD) Matrix

	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8
F = [	10	10	8	26	18	6	8	12]

Backup Links	1	2	3	4	5	6	7	8
1	0		8	26	18	6	8	12
2		0	8	26	18	6	8	12
3			0	6		6		
4	10	10		0				
5	10	10			0			
6						0		
7							0	
8	10	10						0



Capacity C = [50, 50, 150, 150, 50, 50, 50, 150]
Primary Load F = [10, 10, 8, 26, 18, 6, 8, 12], max F = 26
Backup Load G = [26, 26, 6, 10, 10, 0, 0, 10]
Residual Capacity R = [14, 14, 136, 114, 22, 44, 42, 128]

Figure 3. Example of a BLD matrix

Given a network with N links, each network node (router or switch) maintains a $N \times N$ BLD matrix. If the primary load F_j on a link j is B units, entries $BLDM[i,j], 1 \leq i \leq N, j \neq i$, record what fraction of B is backed up on link i. Figure 3 illustrates this concept with an example network with eight links and four primary paths P_1, P_2, P_3, P_4 with bandwidth requirements $10, 8, 12, 6$ units of bandwidth. The corresponding backup paths B_1, B_2, B_3, B_4 are also illustrated. The figure lists four vectors: (1) capacity vector $\boldsymbol{C}$ that records the link capacities, (2) vector $\boldsymbol{F}$ that records the load induced on each link by primary paths, (3) vector $\boldsymbol{G}$ that records the load induced on each link by the backup paths, and (4) residual capacity vector $\boldsymbol{R}$ that records the residual capacity on each link.

Consider link L_4. Primary path P_2, P_3, P_4 use this link and therefore, its primary load is $F_{L_4} = F[4] = 8 + 12 + 6 = 26$ units. The corresponding backup paths are $B_2 = (L_1, L_2)$, $B_3 = (L_1, L_2)$, and $B_4 = (L_1, L_2, L_3)$. Therefore, the backup load on these links can be written down as $G_{L_1} = G[1] = 26, G_{L_2} = G[2] = 26, G_{L_3} = G[3] = 6$.

We can now see that out of $F_{L_4} = 26$ units of primary load on L_4, $8 + 12 + 6 = 26$ units are backed up on L_1, L_2, whereas 6 units are backed up on L_3. As per the definition of BLD matrix, this leads to BLDM entries, $BLDM[1,4] = 26, BLDM[2,4] = 26, BLDM[3,4] = 6$.

Also, note that for row 2, $max_{allj} BLDM[2,j] = 26$ represents the maximum backup load on link L_2 induced by any link in the network. In general, for any row i, $max_{allj} BLDM[i,j]$ represents the maximum backup load induced on link i by all other links. Clearly, for any link i, $max_{allj} BLDM[i,j] \leq G_i$. Also, note that if the entries in row i are sorted in decreasing order, we can identify links that induce successively smaller amount of backup load on link i. This knowledge helps in answering questions such as (1) which two links induce the most load on link i or (2) out of N links, which links induce 50% of backup load on i etc.

The Primary-to-Backup link wastage described earlier is avoided by use of BLD matrix. For the example shown in Figure 2, $BLDM[(L_B, L_P)]$ entry would be 5 as only request $r_1 = 5$ that uses L_p is backed up on L_B, and thus avoid the pessimistic assumption that entire primary load on L_p may be backed up on L_B.

Whenever a node routes new primary and backup connections, it recomputes the $BLDM$ entries and distributes them to other network nodes using OSPF. The changes to OSPF and optimizations for minimizing the overheads of this distribution are not described here due to space constraints. Note that for a network of fixed size, the size of the BLD matrix and therefore, the maximum size of state exchange between network nodes is fixed and is independent of the number of paths. In other words, BLD matrix captures only link state induced by paths but no path specific state. Frequent addition or deletion of paths changes the matrix entries and requires state exchange between network nodes. If the state exchange is completely distributed, in the event copies of BLD matrix at different nodes are inconsistent, two or more nodes may end up selecting paths consisting of links that do not have sufficient capacity to accommodate their requests. In this case, the reservation attempt of some of the nodes will fail and their requests will be rejected. The BLD matrix entries will be consistent after subsequent OSPF updates are processed. One way to minimize BLD matrix inconsistencies is to select *repository* nodes that act as repository for BLD matrix state and serve it to other network nodes. In the event of BLD matrix changes, each node registers its changes with the repository nodes and also, receives notification of changes made by others. Details of these schemes are not discussed in this paper.

4.2 Concept of Free Sharable Bandwidth

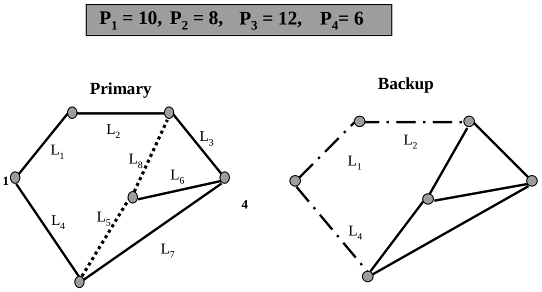


Figure 4. Concept of free bandwidth on a link for backup sharing

Now, we will introduce the concept of *free sharable bandwidth on a link* and show how the use of BLD matrix allows its accurate computation. Consider the example network in Figure 3 with associated BLD matrix and the F, G, R vectors. Figure 4 shows a snapshot of this network where in response to a new LSP request r_{new}, a candidate primary path (L_5, L_8) has been routed but not reserved and (L_4, L_1, L_2) is under consideration as a candidate backup path. We can see from vector G (Figure 3) that the maximum backup load induced on (L_4, L_1, L_2) is $(10, 26, 26)$.

Let us take a closer look at link L_1. From the BLD matrix we know that the backup load induced by links in candidate primary path namely (L_5, L_8) on L_1 is $(BLDM[1, 5]$, $BLDM[1, 8])$ = $(18, 12)$. So a maximum 18 out of 26 units of backup reserved on L_1 will be required for backing up the primary load on (L_5, L_8) even before the new request r_{new} is admitted. In other words, there are extra 8 units of backup bandwidth reserved for backing up some other links. If the new request requires less than 8 units of bandwidth, then no extra bandwidth needs to be reserved on link L_1 in the candidate backup path. We call this 8 units of bandwidth on link L_1 *free* bandwidth.

Formally, given a primary path P, the free bandwidth available on a candidate backup link L is defined as

$$FREE[L] = G[L] - max_{i \in P} BLDM[L, i] \qquad (1)$$

In our example, for backup path (L_4, L_1, L_2), $FREE[L_4] = 10$, $FREE[L_1] = 8$, $FREE[L_2] = 6$, and therefore, if request size b_{new} is 6 units or less, no bandwidth needs to be reserved on the candidate backup path. As shown, the BLD matrix allows more accurate computation of free sharable backup bandwidth on a link.

4.3 Modeling the Link Cost

The backup path computation procedure should favor links that have large *"free backup"* bandwidth. From the perspective of backup routing, every link has two kinds of bandwidth available: (1) ***Free bandwidth (*FR*):*** that is completely sharable and does not require extra resource reservation. (2) ***Residual bandwidth (*R*):*** is the actual capacity left unused on the link. If the LSP request size $b > FR[l]$, then $b - FR[l]$ units of bandwidth must be allocated on the link to account for the worst case backup load on the link. If the residual bandwidth $R[l]$ falls short of $b - FR[l]$ (i.e $b - FR[l] > R[l]$), then the link l cannot be used on the backup path and is called an *"infeasible link"*. Given this, cost of using link l on a backup path consists of two parts: (1) cost of using the free bandwidth on the link and (2) cost of using the residual bandwidth on the link.

$$w[l] \quad = \quad \begin{cases} \infty & \text{if } b - FR[l] > R[l], (l \text{ infeasible}) \\ FR[l] * C_F & \text{if } b \leq FR[l], \\ FR[l] * C_F + \\ (b - FR[l]) * C_R, & \text{if } b > FR[l], \&(b - FR[l] < R[l]) \end{cases} \qquad (2)$$

The cost metrics C_F (C_R) should be selected in such a way that selecting a link with high residual capacity $R[l]$, results in smaller cost. In other words, if $R[l_1] < R[l_2]$, then $C_R[l_1] > C_R[l_2]$. One such function is $C_R[l] = a(1 - \frac{R[l]}{R_{max}})^p$, where $R_{max} = max_l R[l]$. Similarly, if $F_{max} = max_l F$, then $C_F[l] = c(1 - \frac{F[l]}{F_{max}})^q$, satisfies the constraint that if $FR[l_1] < FR[l_2]$, then $C_F[l_1] > C_F[l_2]$.

For primary path routing, the "free bandwidth" does not play a role as the bandwidth has to be always reserved and no sharing is possible. The cost in this case therefore is only the cost incurred in using the residual bandwidth.

Given this cost function for a link, our routing algorithms attempt to find backup paths with minimum cost, where cost of the path is the sum of the cost of component links.

5 Routing Algorithms

In this section, we will describe two types of algorithms: (1) **Two step algorithm:** This algorithm first computes a primary path using one of the many available algorithms such as Minimum Interference Routing (MIRA) [6], or Widest Open Shortest Path First (WSPF) [3]. For this candidate primary path, the algorithm then computes a least cost backup path. (2) **Iterative or Enumeration based algorithm:** This algorithm enumerates pairs of candidate primary and backup paths and picks the pair with smallest joint cost. It uses the WSPF heuristic and associated data structures and and is therefore less generic. Both these algorithms use $\boldsymbol{F}, \boldsymbol{G}, \boldsymbol{R}$ variables per link and the BLD matrix, and run in bounded amount of time. Note that both our algorithms can be deployed along with traditional OSPF and WSPF routing algorithms.

5.1 Generic Two Step Algorithm

The basic pseudo-code for this algorithm is as shown in the Algorithm in Table 1:

Table 1. Generic two-step algorithm

```
 1:   Tree T;
 2:   NetwGraph G,G';
 3:   Path bkup, prm;
 4:  req = $(s,\ d,\ b)$ ;
 5:  prm = GetPrimaryPath (G,req);
 6:  if  (!prm) return (null);
     //  Do Backup path computation
 7:  G'=RemoveLinks(G,prm);
 8:  G'=RemoveInfeasibleLinks(G',BLD,prm);
 9:  AssignCostW(G',BLD,prm);
10: bkp = SPFBackUpPath(G');
11: if  (!bkp) return(null);
12: UpdateNetworkState(G,prm,bkp);
13: return(prm,bkp,cost);
```

The first step in this algorithm (line 5) computes the primary path P using an algorithm such as MIRA, PBR, WSPF. If this step fails, the request is rejected (line 6). Since backup and primary paths must be link disjoint, all links in P are removed from the graph on which backup path is routed (line 7). For the candidate primary path, using the BLD matrix, and Equation 1, the algorithm then computes the $FREE[L]$ on each link in the graph. Then the algorithm removes all infeasible links from the graph and computes new graph G' (line 8). Using the cost metric defined in Equation 2, it then assigns cost $w[l]$ to each link l and computes the backup path using the Shortest Path (SP) algorithm on graph G' (line 10). If no path is found, the path request is rejected. Otherwise, an attempt is made to reserve the resources for primary and backup paths using protocols such as RSVP or LDP. If reservation succeeds, the algorithm updates the path related link state variables and corresponding BLDM matrix entries. It then sends state change packets to the appropriate neighbors (line 12). If the reservation fails, the request is rejected.

We evaluated a specific instantiation of this generic algorithm using the WSPF algorithm for primary path computation. We call this algorithm the Enhanced Widest Shortest Path First (EWSPF). The complexity of this algorithm is $O(m \log n)$ where n is the number of nodes and m is the number of links or edges in the network graph.

5.2 Enumeration Based Algorithm (ENUM-WSPF)

This algorithm enumerates candidate pairs of primary and backup paths using pre-computed data structures in the WSPF implementation and therefore is called ENUM-WSPF. The basic idea in this algorithm is as follows: Given a path request for (s, d, b) units, find the bandwidth bin the request is quantized to (line 5) (See figure 1). Using the SPF trees stored in the bin, find the shortest path from s to d (line 9,10) . Treat this path as a *hypothetical backup path* and find a primary path that induces least cost $w[l]$ on this path by searching SPF trees in all other bins. The search is accomplished by the *for* loop (lines 12-25). When searching for the primary path, it is likely that after links for the backup path are removed, the tree at a given bin may be disconnected for the required s and d pair (line 16). In this case, a more expensive shortest path computation is done on the original graph (lines 17, 18). Using the BLDM matrix, Equations 1 and 2, and cost of primary path, the joint cost of the (bkp, prm) pair is computed (line 20) and compared to the current best pair (line 22-25). At the end of the inner *for* loop (line 26), a best primary path for the backup path from bin is selected. The process is then repeated for every higher bin ($bin \leq lvl \leq k$) (for loop: line 8-27). Clearly, this approach enumerates pairs of primary and backup paths and selects the one with least joint cost. The complexity of this algorithm is $(kmn \log n)$ for pre-computation and $O(k^2)$ for the cost comparison.

6 Simulation Results

In this section, we describe simulations that characterize the benefits of our proposed schemes. We conducted two sets of experiments: (1) **Experiment Set I (EXPTSET-I)** compares three different schemes: EWSPF, ENUM-WSPF, and simple Shortest Path First (SPF). We simulated two different SPF schemes: (1) *SPF-HOP*: uses min-hop-count as path metric and (2) *SPF-RES*: uses link costs based on the residual capacity and computes lowest cost path. Both SPF schemes compute two independent paths: one used as primary and other as backup and do not attempt to share backup paths. (2) **Experiment Set II (EXPTSET-II)** compares our EWSPF scheme with Kodialam's et al scheme's using data sets in their [7] paper.

Table 2. ENUM-WSPF

```
Tree T;
NetwGraph G,G';
Path bkup, prm;
req = $(s, d, b)$ ;
bin = Quantize(b);
if ($bin > k$ )
    bin= k;
for ($lvl = bin; lv++; lvl \leq k$)
  T = GetSPFTree (lvl,s);
    bkp = GetPath(T,d);
    if (!bkp) continue;
    for ($j = 1; j++; j \neq lvl \& j \leq MaxBin$)
        T = GetSPFTree (j,s);
        T'= RemoveLinks(T,bkp);

        prm = GetPath(T',d);
        if (!prm)
            G'= RemoveLinks(G,bkp);
            prm = SPF(G');
        endif
        cost= AssignCostW(bkp,BLD,prm);
        cost  = JointCost(prm,bkp);
        if (mincost > cost)
            best_prm = prm;
            best_bkp = bkp;
        endif
    endfor
endfor
```

6.1 Simulator Details

In the following, we describe the network topologies, traffic parameters and performance metrics we used.

LSP Request Load Table 3 shows the parameters used to run the experiments in EXPTSET-I. Note that in reality, request load at various nodes may not be random and certain node pairs may see dis-proportionate amount of requests. However, no real life call traffic datasets are currently available in public domain.

Table 3. Simulation parameters for EXPTSET-I

Property	Values
Request (REQ) arrival	Poisson at every router
Call holding time (MHT)	100 time units, exponentially distributed
REQ Volume (RV)	50,000 to 300,000
Simulation time (STT)	Fixed 50,000 units
Max LSP REQ SZ (LF)	2.5, 10% of the link capacity
Mean REQ inter-arrival time	Computed using RV and STT
Destination node selection	Randomly distributed

For experiments in EXPTSET-II, we obtained from authors Kodialam et al. a modified version of datasets they used in their paper [7]. Their dataset contains 5 runs each with 100 demands. All demands have infinite call duration – once they are admitted, they do not terminate. The clear drawbacks of this dataset are: (1) The number of demands in the dataset is too small and will not capture the statistical range required to achieve better averaging of performance metrics. (2) Also, unlike the dataset in EXPTSET-I, the infinite connection holding time used in this dataset does not resemble real network conditions, where connections are set up and torn down.

Network Topologies Figure 5, 6 illustrate the homogeneous, heterogeneous network topologies we used in our first set of experiment (EXPTSET-I). These topologies represent the Delaunay triangulation for the 20 largest metros in continental U.S.

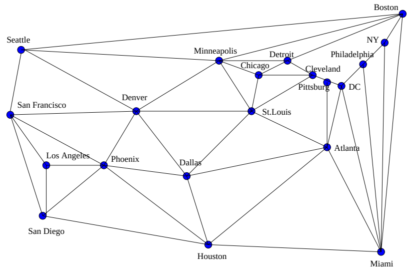


Figure 5. Homogeneous network topology

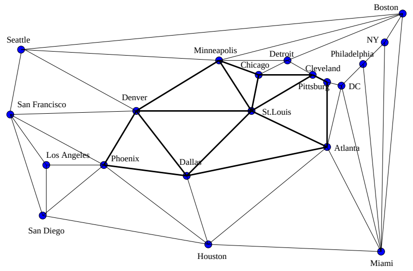


Figure 6. Heterogeneous network topology

For the experiment set II (EXPTSET-II), to compare our EWSP scheme with Kodialam et al.'s scheme [7] we obtained the network topology they used in their paper.

Quantization of Link Bandwidth for WSPF There are two quantization schemes for the bandwidth that we use in the EWSPF and ENUM-WSPF schemes. The exponential quantization (EXP) uses three bandwidth levels of 0.01, 0.1 and 1.0 times the maximum requested bandwidth. The uniform quantization (UNIFORM) uses a more linear set of 6 levels which varies from 0.05,0.1,0.3,0.5,0.7 and 1.0 times the maximum requested bandwidth.

Performance Metrics We used following performance metrics in our experiments: (1) **Fraction Rejected (FR):** is the fraction of requests that were dropped. (2) **Total Bandwidth Saved Fraction (TBSF):** is the fraction of total bandwidth saved when compared to SPF-RES. It is defined as $TBSF = \frac{TotalBW_{new} - TotalBW_{SPF}}{TotalBW_{SPF}}$ In EXPTSET-I, we measured both metrics, whereas in EXPTSET-II we measured the Fraction Rejected (FR) metric as we do not have values for other metrics available from Kodialam et al [7].

6.2 Experiment Set I (EXPTSET-I)

Figure 7(a,b)[1] illustrates the FR and TBSF performance metrics described above measured for the four routing schemes, namely EWSPF, ENUM-WSPF, SPF-HOP, SPF-RES.

Homogeneous Case (1) *Fraction Rejected:* As expected, the FR increases as the load or RV increases. EWSPF and ENUM-WSPF are significantly better than SPF-HOP and SPF-RES with up to 66% gains for 150,000 requests. As the load (volume) increases, EWSPF performs better than ENUM-WSPF. At 300,000 requests EWSPF provides 20% improvement over ENUM-WSPF and 50% gains on SPF. ENUM-WSPF performs slightly worse than EWSPF due to using the pre-computed trees for both primary and backup paths, whereas EWSPF uses the pre-computed trees only for the primary path and recalculates the link weights for the backup path. ENUM-WSPF trades off additional SPF computation and attempts to use the existing trees as much as possible. The main problem with using existing pre-computed information is that the same tree may appear in multiple bandwidth levels nullifying the enumeration process and forcing ENUM-WSPF to resort to the shortest path using residual capacity as the last resort. Hence the performance of ENUM-WSPF which is still significantly better than SPF-RES will tend to SPF-RES especially at higher loads. For the rest of the discussion, we will compare both EWSPF and ENUM-WSPF to SPF-RES which performs slightly better than SPF-HOP. (2) *TBSF vs Request volume:* In terms of the overall bandwidth saved when compared to SPF-RES, we see that EWSPF saves 33% and ENUM-WSPF saves around 18% over SPF. The gains decrease with increase in load since links are saturated and finding free shareable bandwidth becomes increasingly difficult with an increase in the number of requests.

Heterogeneous Case

(1) *Fraction Rejected vs Volume:* Figure 7(c) shows FR vs the volume for an LF value of 10% of the OC-48 links. Since we have the access links at OC-12, an LF of 10% of OC-48 will result in some requests that are nearly 50% of the access link. This causes the access links to get saturated very quickly and leads to higher rejection probabilities for all the schemes. The gains of EWSPF/ENUM-WSPF are also less over SPF-RES at the higher LF due to the early saturation leading to dropping requests. (2) *TBSF vs Volume:* We found that gains of EWSPF/ENUM-WSPF are sensitive to LF; they are less at an LF of 10% as compared to an LF of 2.5%. From Figure 8, we see that EWSPF provides around 28% gains and ENUM-WSPF provides 13% gains for a volume of 150,000 requests. However, once the LF is reduced to 2.5% [2] the gains for EWSPF and ENUM-WSPF improve to 35% and 20% respectively for the same request volume.

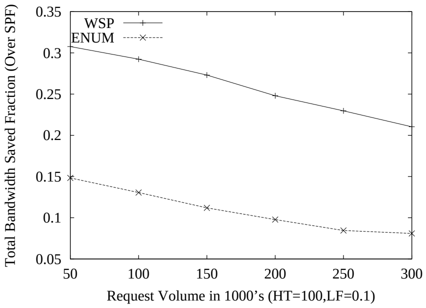


Figure 8. TBSF (heterogeneous (LF=0.10)

6.3 EXPTSET-II: Comparison to Kodialam et al.'s scheme

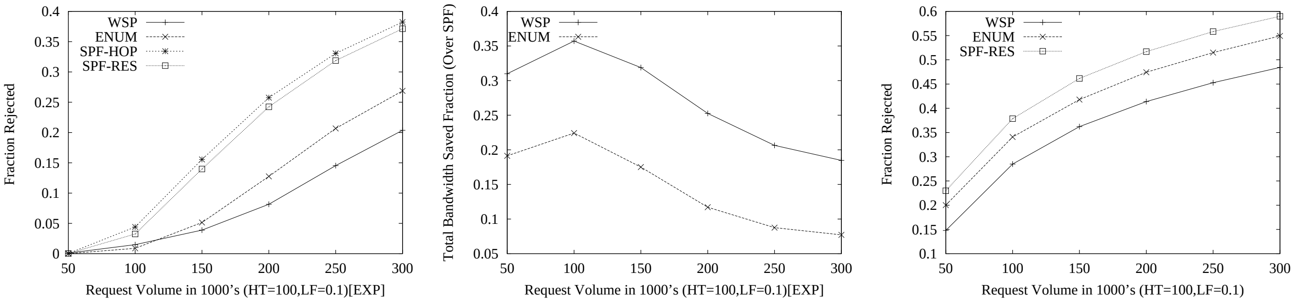
From the results of EXPTSET-I, we see that EWSP performs well in all the cases we considered and is very simple to implement. Therefore, we selected EWSPF as a candidate algorithm to compare with Kodialam et al. algorithm. We modified our simulator to handle the dataset described in 6.1. Kodialam et al.'s scheme models the backup path routing as a linear programming problem that uses only three variables F, G, R. It develops a dual-based algorithm that solves the primal linear program to obtain an upper bound UB and it's dual problem to obtain a lower bound LB. Iteratively running the algorithm reduces (UB-LB) difference and brings the solutions closer to the optimal. Each iteration involves solving multiple shortest path problems and a large number of iterations ranging from 100-500 may be required to get a satisfactory convergence. We call this scheme as *Linear Programming Approach* (LPA) for the rest of this discussion. We performed two kinds of experiments: (a) EXPT A: when links have infinite capacity and no request is rejected; (b) when links have finite capacity and requests are dropped. For the first set of experiments,

Table 4. Comparison of EWSP with LPA

Scheme	Total BW (EXPT A)	Request Rejection Fraction (EXPT B)
EWSP	2722	0.062
LPA	2736	0.064

[1] In all graphs, legend WSP stands for EWSPF and ENUM stands for ENUM-WSPF.

[2] Graph for LF=2.5% not shown



(a) Homogeneous - Fraction rejected

(b) Homogeneous- TBSF

(c) Heterogeneous - Fraction rejected

Figure 7. Performance results for various topologies

we measured the total bandwidth that is reserved by the schemes for all requests. For the second EXPT B, The goal of the second set of experiments is to measure the rejection fraction. Our results are summarized in Table 4.

We can see that our scheme shows improvement in the rejection fraction over LPA. The savings accrue from the use of BLDM to reduce Primary-to-Backup link wastage that was described in 3. However, we also noticed significant standard deviation among 5 runs (each with 100 demands). We believe that the limited size of the datasets is the cause of such large deviations, and also the relatively small performance gains. Our scheme is very simple as it involves only two shortest-path computations, unlike Kodialam et al's scheme which requires 10-100s of SPFs. It is also easy to deploy, since it is directly based on link state protocols. In our on going work, we plan to obtain source code of Kodialam et al's algorithm to test it on larger datasets.

7 Conclusions

In this paper, we addressed the problem of distributed routing of bandwidth guaranteed paths with restoration in generic label switched networks . We proposed a new form of constant size state information called Backup Load Distribution Matrix (BLDM) that captures for each link, the distribution of primary load backed up on other links in the network. This matrix eliminates the bandwidth wastage common in approaches that use only three variables per link l, namely: $\boldsymbol{F}$, load induced by the primary paths, $\boldsymbol{G}$, load induced by the backup paths, and $\boldsymbol{R}$, residual bandwidth. We proposed two new algorithms: (1) Enhanced Widest Shortest Path First (EWSPF) and (2) Enumeration Widest Shortest Path First (ENUM-WSPF) that use the BLD matrix and run in bounded amount of time. Our simulation results for sample topologies show 30-50% reduction in number of rejected requests and 30-40% savings in total bandwidth used for backup connections.

References

[1] Andersson, L., et al., "LDP Specification," *IETF RFC 3036*, January, 2001.

[2] Apostolopoulos, G., et al.,"Quality of Service Routing: A Performance Perspective," *ACM SIGCOMM98*, September, 1998.

[3] Apostolopoulos, G., et al., "QoS routing mechanisms and OSPF extensions", IETF draft, December 1998.

[4] Der-Hwa Gan, Li, et al., "RSVP-TE: Extensions to RSVP for LSP Tunnels", IETF Internet Draft.

[5] Ma, H., et al.,"Constraint Based Design of ATM Networks, an Experimental Study," *Technical Report WUCS-97-15,, Washington University in St. Louis*, 1997.

[6] Kodialam, M., et al., "Minimum Interference Routing with Applications to MPLS Traffic Engineering," *INFOCOM2000*, March 2000.

[7] Kodialam, M., et al., "Dynamic Routing of Bandwidth Guaranteed Paths with Restoration," *IEEE INFOCOM2000*, March 2000.

[8] Zhou, D., and Subramaniam, S., "Survivability in Optical Networks," *IEEE Network*, pp 16-23, Vol 14, No. 6, Dec 2000.