Throughput Guaranteed Restorable Routing Without Traffic Prediction

M. Kodialam T. V. Lakshman Sudipta Sengupta
Bell Laboratories, Lucent Technologies, NJ, USA

***Abstract*— Two-phase routing, where traffic is first distributed to intermediate nodes before being routed to the final destination, has been recently proposed [10], [11], [18] for handling widely fluctuating traffic without the need to adapt network routing to changing traffic. Pre-configuring the network in a traffic independent manner using two-phase routing simplifies network operation considerably.**

In this paper, we extend this routing scheme by providing resiliency against link failures through two different *fast restoration* mechanisms – local (link/span) based and end-to-end (path) based. We view this as important progress towards adding carrier-class reliability to the robustness of the scheme so as to facilitate its future deployment in Internet Service Provider (ISP) networks. The main contribution of the paper is the development of fast combinatorial algorithms for routing under the scheme with link and path restoration mechanisms so as to minimize the maximum utilization of any link in the network, or equivalently, maximize the throughput. The algorithms developed are Fully Polynomial Time Approximation Schemes (FPTAS) – for any given $\epsilon > 0$, an FPTAS guarantees a solution that is within a $(1+\epsilon)$-factor of the optimum and runs in time polynomial in the input size and $\frac{1}{\epsilon}$. To the best of our knowledge, this is the first work in the literature that considers making the scheme resilient to link failures through pre-provisioned fast restoration mechanisms. We evaluate the performance of link and path restoration (in terms of throughput) and compare it with that of unprotected routing. For our experiments, we use actual ISP network topologies collected for the Rocketfuel project.

I. Introduction

To provide good service when network traffic patterns can change uncontrollably, carriers must either quickly and repeatedly adapt their intra-domain routing to avoid network congestion or must have sufficient capacity set aside a priori to accommodate the different traffic patterns that can occur without resorting to routing changes. Service providers prefer to avoid frequent intra-domain routing changes due to operational complexity and costs, and due to the risk of network instability if link metric changes are not implemented correctly. Moreover, routing changes in one Autonomous System (AS) may cause cascading traffic changes in other ASes affecting the overall stability of many Internet paths. The trade-off in avoiding routing changes is the significant capacity overprovisioning that must be done to accommodate changing traffic patterns while keeping the routing fixed. Ideally, providers would like to use a fixed routing scheme that does not require traffic dependent dynamic adaptation of configuration parameters and which does not use too much extra capacity.

The recently proposed two-phased routing scheme [10], [11], [18] meets the service provider need of reducing operational complexity by allowing the network to handle all traffic patterns possible, within the networks ingress-egress capacity constraints, without dynamic adaptation. Avoiding dynamic adaptation reduces the need for accurate traffic forecasts and constant monitoring of changes to traffic patterns. The two-phase routing scheme works as follows: Incoming traffic is sent during the first phase from the source to a set of intermediate nodes in predetermined proportions that are a function of the intermediate node. Next, in the second phase, the traffic is sent from the intermediate nodes to the final destination.

This routing scheme can be used in a wide variety of networking scenarios as discussed in [11]. Some examples are: (i) provisioning service overlays such as the Internet Indirection Infrastructure (i3) [17], (ii) provisioning Virtual Private Networks, (iii) adding QoS guarantees to services that require routing through a network-based middlebox, and (iv) reducing IP layer transit traffic and handling extreme traffic variability in IP-over-Optical networks without dynamic reconfiguration of the optical layer.

In this paper, we extend the two-phase routing scheme by providing resiliency against link failures through two different *fast restoration* mechanisms – local (link/span) based and end-to-end (path) based. We view this as important progress for two-phase routing towards achieving carrier-class reliability so as to facilitate its future deployment in ISP networks.

The restoration models we consider are introduced in Section V and have been classified in the literature under "fast restoration" because of their (relatively) low restoration latencies. In both restoration models, backup bandwidth is shared across single link failure events so as to reduce restoration capacity overhead. Backup bandwidth can also be allocated in a dedicated manner. The focus on shared allocation in this paper is because of its reduced cost, the rarity of concurrent multiple link failures in networks, and the increased complexity of the optimization problems that arises from sharing backup bandwidth.

The main contribution of the paper is the development of fast combinatorial algorithms for two-phase routing with link and path restoration mechanisms so as to minimize the maximum utilization of any link in the network, or equivalently, maximize the throughput. This is the first work in the literature that considers making two-phase routing resilient to link failures through fast restoration mechanisms.

The combinatorial algorithms developed are Fully Polynomial Time Approximation Schemes (FPTAS). An FPTAS is an algorithm that finds a solution with objective function value within $(1+\epsilon)$-factor of the optimal solution and runs in time that is a polynomial function of the input parameters and $\frac{1}{\epsilon}$. The input parameters in our problem are the number of nodes n and links m in the network, and the size (number of bits) of the input numbers, e.g., link capacities and node ingress-egress capacities. The value of ϵ can be chosen to provide the desired degree of optimality for the solution.

In this paper, we use the inverse of the maximum link utilization as the performance metric. This is because this

1-4244-0593-9/06/$20.00 ©2006 IEEE

metric, which we refer to as throughput in the rest of the paper, is directly related to link congestion. Also, it is a widely used metric in capacity planning. When considering feasibility of a traffic matrix on (various what-if) capacitated network deployment scenarios, throughput is probably the most suitable metric to consider (feasibility is indicated by a throughput greater than or equal to 1).

The paper is structured as follows. In Section II, we discuss some aspects of the inherent difficulty in measuring traffic and reconfiguring the network in response to changes in it and introduce the traffic variation model. Related work is reviewed in Section III. In Section IV, we briefly discuss the two-phase routing scheme so as to provide context for this paper. The restoration models we consider are introduced in Section V. In Section VI, we propose addition of link restoration to the two-phase routing scheme and develop linear programming formulations and fast combinatorial algorithms for maximum throughput routing. In Section VII, we propose addition of path restoration to the two-phase routing scheme and develop linear programming formulations and fast combinatorial algorithms for maximum throughput routing. We evaluate the performance of link and path restoration (in terms of throughput) and compare it with that of unprotected routing in the two-phase scheme in Section VIII. For our experiments, we use actual ISP network topologies collected for the Rocketfuel project [13]. We conclude in Section IX. We briefly describe some notation below before moving on to the next section.

A. Notation

We assume that we are given a network $G = (N, E)$ with node set N and (directed) edge set E where each node in the network can be a source or destination of traffic. Let $|N| = n$ and $|E| = m$. The sets of incoming and outgoing edges at node i are denoted by $E^-(i)$ and $E^+(i)$ respectively. We let (i, j) represent a directed link in the network from node i to node j. To simplify the notation, we will also refer to a link by e instead of (i, j). The capacity of link (i, j) will be denoted by u_{ij}. The *utilization* of a link is defined as the traffic (sum of working traffic and maximum restoration traffic due to any single link failure) on the link divided by its capacity.

II. Traffic Measurement and Variability

In an utopian network deployment scenario where complete traffic information is known and does not change over time, we can optimize the routing for that single traffic matrix – a large volume of research has addressed this problem. The most important innovation of the two-phase routing scheme is the handling of traffic variability in a capacity efficient manner through static pre-configuration of the network and without requiring either (i) measurement of traffic in real-time or (ii) re-configuration of the network in response to changes in it. We address the difficulties associated with (i) and (ii) so as to further bring out the novelty of two-phase routing.

A. Difficulties in Measuring Traffic

Network traffic is not only hard to measure in real-time but even harder to predict based on past measurements. Direct measurement methods do not scale with network size as the number of entries in a traffic matrix is quadratic in the number of nodes. Moreover, such direct real-time monitoring methods lead to unacceptable degradation in router performance. In reality, only aggregate link traffic counts are available for traffic matrix estimation. SNMP (Simple Network Management Protocol) provides these data via incoming and outgoing byte counts computed per link every 5 minutes. To estimate the traffic matrix from such link traffic measurements, the best techniques today give errors of 20% or more [14].

The emergence of new applications on the Internet, like P2P (peer-to-peer), VoIP (voice-over-IP), and video-on-demand has reduced the time-scales at which traffic changes dynamically, making it impossible to extrapolate past traffic patterns to the future. Currently, ISPs handle such unpredictability in network traffic by gross over-provisioning of capacity. This has led to ISP networks being under-utilized to levels below 30% [14].

B. Difficulties in Dynamic Network Re-Configuration

Even if it were possible to track changes in the traffic matrix in real-time, dynamic changes in routing in the network may be difficult or prohibitively expensive from a network operations perspective. In spite of the continuing research on network control plane and IP-Optical integration, network deployments are far away from utilizing the optical control plane to provide bandwidth provisioning in real-time to the IP layer. The unavailability of network control plane mechanisms for reconfiguring the network in response to and at time-scales of changing traffic further amplifies the necessity of the static pre-configuration property of two-phase routing in handling traffic variability.

C. Traffic Variation Model

We consider a traffic variation model where the total amount of traffic that enters (leaves) an ingress (egress) node in the network is bounded by the total capacity of all external ingress links at that node. This is known as the *hose model* and was proposed by Fingerhut et al . [7] and subsequently used by Duffield et al. [6] as a method for specifying the bandwidth requirements of a Virtual Private Network (VPN). Note that the hose model naturally accommodates the network's ingress-egress capacity constraints.

We denote the upper bounds on the total amount of traffic entering and leaving at node i by R_i and C_i respectively. The point-to-point matrix for the traffic in the network is thus constrained by these ingress-egress link capacity bounds. These constraints are the only known aspects of the traffic to be carried by the network, and knowing these is equivalent to knowing the row and column sum bounds on the traffic matrix. That is, any allowable traffic matrix $T = [t_{ij}]$ for the network must obey

$$\sum_{j:j\neq i}^{n} t_{ij} \leq R_i, \quad \sum_{j:j\neq i}^{n} t_{ji} \leq C_i \ \ \forall\ i \in N$$

For given R_i and C_i values, denote the set of all such matrices that are partially specified by their row and column sums by $\mathcal{T}(\mathcal{R}, \mathcal{C})$, that is

$$\mathcal{T}(\mathcal{R}, \mathcal{C}) = \{[t_{ij}] | \sum_{j\neq i} t_{ij} \leq R_i \text{ and } \sum_{j\neq i} t_{ji} \leq C_i \ \ \forall\ i\}$$

We will use $\lambda \cdot \mathcal{T}(\mathcal{R}, \mathcal{C})$ to denote the set of all traffic matrices in $\mathcal{T}(\mathcal{R}, \mathcal{C})$ with their entries multiplied by λ.

Note that the traffic distribution T could be any matrix in $\mathcal{T}(\mathcal{R}, \mathcal{C})$ and could change over time. Two-phase routing provides a routing architecture that does not make any assumptions about T apart from the fact that it is partially specified by row and column sum bounds and can provide QoS guarantees for routing all matrices in $\mathcal{T}(\mathcal{R}, \mathcal{C})$ without requiring any detection of changes in traffic patterns or dynamic network reconfiguration in response to it. For the Rocketfuel topologies, the throughput of two-phase routing is within 6% of the optimal scheme among the class of all schemes that are allowed to reconfigure the network in response to traffic changes [11].

III. Related Work

Direct routing from source to destination (instead of in two phases) along *fixed* paths for the hose traffic model has been considered by Duffield et al. [6] and Kumar et al. [9]. In related work, Applegate et al. [3] consider fixed path routing and provide *relative guarantees* for routing an arbitrary traffic matrix with respect to the best routing for that matrix. However, they do not provide *absolute bandwidth guarantees* for routing variable traffic under the hose model.

Two aspects of direct source-destination path routing, namely, (i) the source needs to *know the destination of a packet* for routing it, and (ii) the bandwidth requirements of the (fixed) paths change with traffic variations, render them unsuitable for some network architectures and applications. Because of (i), these methods cannot be used to provide *indirection in service overlay models* like i3 where the *final destination of a packet is not known at the source*. Because of (ii), the adaptation of these methods for IP-over-Optical networks necessitates detection of changes in traffic patterns and dynamic reconfiguration of the provisioned optical layer circuits in response to it, a functionality that is not present in current IP-over-Optical network deployments.

Applegate et al. [2] extend their work to cope with network failures. However, they do not consider any restoration mechanisms with pre-provisioned backup paths. Instead, they provide linear programming formulations for re-routing affected traffic after failure. In addition to the above points in favor of two-phase routing, providing fast restoration through pre-provisioned shared backup paths is another differentiating aspect of our current work.

Our current work is a sequel to [11]. In [18], the capacity impact of arbitrary IP layer link and node failures is considered for a version of the scheme with *equal traffic split ratios of* $\frac{1}{n}$ and *equal ingress-egress capacities* ($R_i = C_i = c$ for all i). The authors in [18] further assume that the IP layer topology is a full-mesh (fully connected complete graph), so that the Phase 1 and Phase 2 paths are one hop in length. These paths need to be routed (via multi-hop paths) on the physical WDM topology (which is a sparse graph), an important aspect which they do not consider. Also, if the IP topology is not full-mesh, the Phase 1 and Phase 2 paths will be multi-hop at the IP layer itself. Our problem formulation for two-phase routing in [10], [11] (and in this paper) models the multi-hop routing of Phase 1 and Phase 2 paths and can be applied to a general IP layer topology and a physical WDM topology.

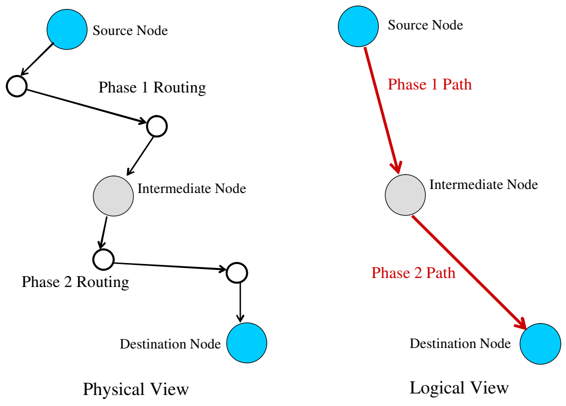


Fig. 1. Two-Phase Routing

IV. Overview of Two-Phase Routing Scheme

In this section, we give an overview of the two-phase routing scheme from [10]. As mentioned earlier, the scheme does not require the network to detect changes in the traffic distribution or re-configure the network in response to it. The only assumption about the traffic is the limits imposed by the ingress-egress constraints at each node, as outlined in Section II-C.

As is indicative from the name, the routing scheme operates in two phases:

- **Phase 1:** A predetermined fraction α_j of the traffic entering the network at any node is distributed to every node j *independent of the final destination of the traffic*.
- **Phase 2:** As a result of the routing in Phase 1, each node receives traffic destined for different destinations that it routes to their respective destinations in this phase.

This is illustrated in Figure 1. Note that the traffic split ratios $\alpha_1, \alpha_2, \ldots, \alpha_n$ in Phase 1 of the scheme are such that $\sum_{i=1}^{n} \alpha_i = 1$. A simple method of implementing this routing scheme in the network is to form *fixed bandwidth paths between the nodes*. In order to differentiate between the paths carrying Phase 1 and Phase 2 traffic, we will refer to them as Phase 1 and Phase 2 paths respectively. The critical reason the two-phase routing strategy works is that the *bandwidth required for these tunnels depends on the ingress-egress capacities R_i, C_i and the traffic split ratios α_j but not on the (unknown) individual entries in the traffic matrix.*

We now derive the bandwidth requirement for the Phase 1 and Phase 2 paths. Consider a node i with maximum incoming traffic R_i. Node i sends $\alpha_j R_i$ amount of this traffic to node j during the first phase for each $j \in N$. Thus, the traffic demand from node i to node j as a result of Phase 1 routing is $\alpha_j R_i$. At the end of Phase 1, node i has received $\alpha_i R_k$ traffic from any other node k. Out of this, the traffic destined for node j is $\alpha_i t_{kj}$ since all traffic is initially split without regard to the final destination. The traffic that needs to be routed from node i to node j during Phase 2 is $\sum_{k \in N} \alpha_i t_{kj} \leq \alpha_i C_j$. Thus, the traffic demand from node i to node j as a result of Phase 2 routing is $\alpha_i C_j$.

Hence, the maximum demand from node i to node j as a result of routing in Phases 1 and 2 is $\alpha_j R_i + \alpha_i C_j$. Note that this does not depend on the matrix $T \in \mathcal{T}(\mathcal{R}, \mathcal{C})$. The scheme handles variability in traffic matrix $T \in \mathcal{T}(\mathcal{R}, \mathcal{C})$ by effectively routing the fixed matrix $D = [d_{ij}] = [\alpha_j R_i + \alpha_i C_j]$

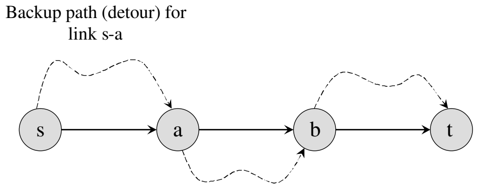


Fig. 2. Link backup detours protecting links on primary path

that depends only on aggregate ingress-egress capacities and the traffic split ratios $\alpha_1, \alpha_2, \ldots, \alpha_n$, and not on the specific matrix $T \in \mathcal{T}(\mathcal{R}, \mathcal{C})$. This is what makes the routing scheme oblivious to changes in the traffic distribution.

An instance of the scheme requires specification of the traffic distribution ratios $\alpha_1, \alpha_2, \ldots, \alpha_n$ and routing of the Phase 1 and Phase 2 paths. In [11], linear programming formulations and fast combinatorial algorithms are developed for computing the above so as to maximize network throughput.

V. Restoration Models

We introduce the two restoration models considered in this paper, namely *link restoration*, also called local (span) restoration, and *path restoration*, also called end-to-end restoration. Both these mechanisms have been classified under *fast restoration* in the literature because of their (relatively) low restoration latency. The explanation for this is provided in the respective description of the mechanisms.

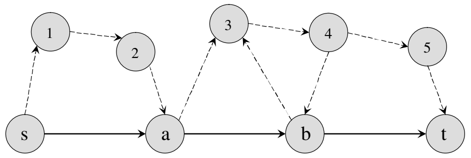
In the two-phase routing scheme described above, each of the Phase 1 and Phase 2 paths can be protected against link failures by link restoration or path restoration. The main contribution of this paper is the development of algorithms for maximum throughput two-phase routing with resiliency against link failures provided by the described restoration mechanisms.

A. Link Restoration

For protecting link failures with link restoration, a path P consists of a primary (working) path, denoted by $W(P)$, and a link backup detour, denoted by $B_e(P)$, for each link e on $W(P)$. This is illustrated in Figure 2. Thus, a primary path with h hops is associated with h link detours for local restoration against link failures. When we refer to a path P in the context of link restoration, it will consist of the primary path and the link backup detours for each link on the primary path.

Under the single link failure model, backup paths for *different* links can share bandwidth both within the same as well as across different connection(s). As illustrated in Figure 3, backup detour a-3-4-b for link a-b and backup detour b-3-4-5-t for link b-t can share bandwidth on their common link 3-4.

The fast nature of link restoration arises from two aspects: (i) fast failure detection by the nodes adjacent to the failed link, and (ii) fast signaling after failure along short link detours paths, in case such signaling is required, as in optical mesh networks in order to setup cross-connects on the link detours [16].



Link	Backup Path (Detour)
s-a	s-1-2-a
a-b	a-3-4-b
b-t	b-3-4-5-t

Fig. 3. Backup bandwidth sharing across link backup detours

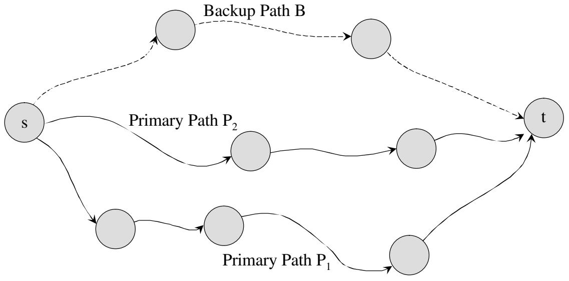


Fig. 4. Diverse primary paths P_1, P_2 and backup path B for path restoration

B. Fast Path Restoration

For path restoration, each connection consists of K (≥ 2) link-disjoint paths from source to destination. For the special case $K = 2$, also called 1:1-protection, a connection P consists of a primary (working) path, and a link-disjoint backup path. Traffic is sent on the primary path during normal (no-failure) conditions and switched to the backup path after any failure that affects the primary path.

The 1:1-protection scheme can be extended to a more general scheme with the objective of reducing the protection capacity overhead of the network. We allow a connection P to consist of $K(\geq 2)$ link-disjoint paths from source to destination. If the working traffic associated with this connection is Δ, then an amount $\frac{\Delta}{K-1}$ of working traffic is sent on each of $K-1$ disjoint paths. The remaining path is designated as the backup path. This is illustrated in Figure 4. Under a single link failure model, only one of the $K-1$ (disjoint) primary paths can fail, in which event the backup path carries $\frac{\Delta}{K-1}$ portion of the working traffic. One can designate any $K-1$ of the paths (usually the $K-1$ shortest ones) as primary and the remaining as backup. Clearly, for $K = 2$, the scheme reduces to 1:1-protection.

The fast nature of the extended scheme arises from the fact that the source needs to just switch traffic to the backup path after one of the primary paths fails and the destination needs to select traffic from the backup path. (The source needs to receive failure notification from the destination or the nodes adjacent to the failed link before switching.) For optical mesh networks, cross-connects are already setup on the backup path during provisioning, hence no signaling on the backup path is required after failure.

VI. Adding Link Restoration to Two-Phase Routing

In order to make two-phase routing resilient to link failures using link restoration, we add link backup detours protecting each link as discussed in Section V-A. Given a network with link capacities and constraints R_i, C_j on the ingress-egress traffic as discussed, we consider the problem of two-phase routing with link restoration so as to minimize the maximum utilization of any link in the network. The problem is equivalent to finding the maximum multiplier λ (throughput) such that all matrices in $\lambda \cdot \mathcal{T}(\mathcal{R},\mathcal{C})$ can be feasibly routed with link restoration.

Before proceeding, we give an alternative (but equivalent) definition of throughput. Suppose we relax the requirement that the traffic split ratios sum to 1 in a feasible solution of the problem. Recall that the demand from i to j is $\alpha_j R_i + \alpha_i C_j$. Consider the sum

$$\lambda = \sum_{i \in N} \alpha_i$$

The traffic split ratios can be normalized (divided) by λ so that they sum to 1, in which case all matrices in $\lambda \cdot \mathcal{T}(\mathcal{R},\mathcal{C})$ can be feasibly routed. Thus, the appropriate measure of throughput is the quantity λ as defined above *when the traffic split ratios are not constrained to sum to* 1.

We first present a path indexed linear programming formulation for this problem. This will be subsequently used to develop the fast combinatorial algorithm in Section VI-B.

A. Path Indexed Linear Programming Formulation

Let $\mathcal{P}_{ij}$ denote the set of all paths (with link detours) from node i to node j. Let $x(P)$ denote the working traffic associated with path P. Then, the problem of two-phase routing with link restoration so as to maximize the network throughput can be formulated as the following path-indexed linear program:

$$\text{maximize} \quad \textstyle\sum_{i \in N} \alpha_i$$

subject to

$$\begin{aligned}
\sum_{P \in \mathcal{P}_{ij}} x(P) \quad &= \quad \alpha_j R_i + \alpha_i C_j \\
& \qquad \forall\ i,j \in N, i \neq j \qquad (1) \\
\sum_{i,j} \sum_{P \in \mathcal{P}_{ij}, e \in W(P)} x(P) \quad &+ \quad \sum_{i,j} \sum_{P \in \mathcal{P}_{ij}, e \in B_f(P)} x(P) \leq u_e \\
& \qquad \forall\ e,f \in E, e \neq f \qquad (2) \\
x(P), \alpha_i \quad &\geq \quad 0 \ \ \forall\ P \in \mathcal{P}_{ij}, \ \ \forall\ i,j \in N \ (3)
\end{aligned}$$

Constraints (1) correspond to the routing of $\alpha_j R_i + \alpha_i C_j$ amount of flow from node i to node j along link-restored paths. Constraints (2) state that that the sum of working traffic on a link and the restoration traffic that appears on that link after failure of any other link is at most the capacity of the link.

Let α_i^* be the α_i values in an optimal solution of the above linear program. Then, the maximum achievable throughput is given by $\lambda^* = \sum_i \alpha_i^*$. The α_i^* values can be reduced by a factor of λ^* to get the actual split ratios that sum to 1.

In general, a network can have an exponential number of paths (in the size of the network). Hence, this linear program can have possibly exponential number of variables and is not suitable for running on medium to large sized networks. The path-indexed formulation can be converted to a polynomial size link-indexed program, thus allowing it to be solved in polynomial time using a general linear programming algorithm. We omit this for lack of space. In Section VI-B, we consider the dual of the above linear program. The usefulness of the primal and dual formulation is in designing a fast (polynomial time) combinatorial algorithm for each problem.

B. Combinatorial Algorithm

In this section, we develop a fast combinatorial algorithm (FPTAS) for two-phase routing with link restoration. We begin with the dual formulation of the linear program discussed above. The primal-dual approach we develop is adapted from the technique in Garg and Könemann [8] for solving the maximum multicommodity flow problem, where flows are augmented in the primal solution and dual variables are updated in an iterative manner.

The dual formulation of the linear program outlined in Section VI-A associates a variable $w(e,f)$ with each link capacity constraint in (2) and a variable π_{ij} with each demand constraint in (1).

For each link $e = (i,j) \in E$, denote by $g(e)$ the cost of the shortest detour from node i to node j under link costs $c(e') = w(e', e) \ \ \forall\ e' \in E, e' \neq e$ and $c(e) = \infty$. Also, let $SP(i,j)$ denote the cost of the shortest path from node i to node j under links costs

$$c(e) = g(e) + \sum_{f \in E, f \neq e} w(e,f) \ \ \forall\ e \in E$$

Essentially the definition of $SP(i,j)$ corresponds to a minimum cost path $P \in \mathcal{P}_{ij}$ whose links e on working path $W(P)$ have cost $\sum_{f \neq e} w(e,f)$ and backup detours $B_e(P)$ protecting each primary link e have cost $g(e)$.

With the definition of the quantity $SP(i,j)$ as above and simplification and removal of the dual variables π_{ij}, the dual linear program can be written as:

$$\text{minimize} \quad \textstyle\sum_{e \in E} \sum_{f \in E, f \neq e} u_e w(e,f)$$

subject to

$$\begin{aligned}
\sum_{i: i \neq k} R_i SP(i,k) \quad &+ \quad \sum_{j: j \neq k} C_j SP(k,j) \geq 1 \ \ \forall\ k \in N \qquad (4) \\
w(e,f) \quad &\geq \quad 0 \ \ \forall\ e,f \in E, e \neq f \qquad (5)
\end{aligned}$$

For a given node $k \in N$, let $V(k)$ denote the left-hand-side (LHS) of constraint (4). Given the weights $w(e,f)$, note that $V(k)$ can be computed in polynomial time by simple shortest path computations.

Given a set of weights $w(e,f)$, it is a feasible solution for the dual program if and only if

$$\min_{k \in N} V(k) \geq 1$$

The algorithm works as follows. Start with equal initial weights $w(e,f) = \delta$ (the quantity δ depends on ϵ and is derived later). Repeat the following until the dual feasibility constraints are satisfied:

1) Compute the node $k = \bar{k}$ for which $V(k)$ is minimum. This identifies a node $\bar{k}$ as well as paths (with link

detours) P_i from node i to node $\bar{k}$ for all i and paths (with link detours) Q_j from node $\bar{k}$ to node j for all j.

2) For each $e, f \in E, e \neq f$, let $N_P(e)$ be the set of nodes i for which $W(P_i)$ contains link e and $N'_P(e, f)$ be the set of nodes i for which $B_f(P_i)$ contains link e. Similarly, let $N_Q(e)$ be the set of nodes j for which $W(Q_j)$ contains link e and $N'_Q(e, f)$ be the set of nodes j for which $B_f(Q_j)$ contains link e. Compute the quantity α as follows:
$$S(e) = \sum_{i \in N_P(e)} R_i + \sum_{j \in N_Q(e)} C_j \quad \forall\, e \in E$$
$$S'(e, f) = \sum_{i \in N'_P(e,f)} R_i + \sum_{j \in N'_Q(e,f)} C_j \quad \forall\, e \neq f$$
$$\alpha = \min_{e,f \in E} \frac{u_e}{S(e) + S'(e, f)}$$
3) Send αR_i amount of flow on path P_i for all i and αC_j amount of flow on path Q_j for all j. For each link e, compute the total working flow $\Delta(e)$ and the flow $\Delta'(e, f)$ that appears on link e after failure of any other link $f \neq e$.
4) For each $e, f \in E$, $e \neq f$, update weights $w(e, f)$ as
$$w(e, f) \leftarrow w(e, f) \left(1 + \frac{\epsilon[\Delta(e) + \Delta'(e, f)]}{u_e}\right)$$
5) Increment the split ratio $\alpha_{\bar{k}}$ associated with node $\bar{k}$ by α.

When the above procedure terminates, dual feasibility constraints will be satisfied. However, primal capacity constraints on each link will be violated, since we were working with the original (and not residual) link capacities at each stage. To remedy this, we scale down the split ratios α_i uniformly so that capacity constraints are obeyed.

Note that since the algorithm maintains primal and dual solutions at each step, the optimality gap can be estimated by computing the ratio of the primal and dual objective function values. The computation can be terminated immediately after the desired closeness to optimality is achieved.

We briefly outline an efficient method for computing the value of the LHS of dual constraint (4) for a given $k \in N$ at each iteration of the algorithm. The quantity $SP(i, j)$, for each $i, j \in N$, is computed as follows:

A. For each link $e = (i, j) \in E$, compute the cost $g(e)$ of the shortest link detour from node i to node j under link costs $c(e') = w(e', e) \ \ \forall\, e' \in E, e' \neq e$ and $c(e) = \infty$, using Dijkstra's algorithm [1].

B. Using an all-pairs shortest paths computation, compute the cost $SP(i, j)$ of the shortest path from i to j under link costs $c(e) = g(e) + \sum_{f \in E, f \neq e} w(e, f) \ \ \forall\, e \in E$.

The complete path $P \in \mathcal{P}_{ij}$ (primary path with link detours) with cost $SP(i, j)$ is identified as follows. The primary path $W(P)$ is given by the path computed in Step B. The link backup detours for each $f \in W(P)$ are obtained from Step A.

Step A involves m single shortest path computations. Step B involves n single shortest path computations. Hence, the above procedure involves m+n Dijkstra shortest path computations plus $O(n + m)$ time. Dijkstra's shortest path algorithm can be implemented in $O(m + n \log n)$ time using Fibonacci heaps [1]. Hence, each iteration of the procedure can be implemented in $O(m^2 + nm \log n)$ time.

In the context of optical mesh networks [16], cross-connects need to be setup on the link backup detour(s) after failure for restoration. This involves end-to-end signaling on the link detour. Hence, in order to bound restoration latency in optical networks, it may be necessary to impose a hop constraint (say, at most h hops) on each link detour. This is easily incorporated into our algorithm by restricting link backup detours to have at most h hops and using the Bellman-Ford algorithm [1] in Step 1 above to compute shortest cost paths bounded by a hop count of h.

```
Algorithm LINK_RESTORATION:

$\alpha_k \leftarrow 0 \quad \forall\, k \in N$ ;
$w(e, f) \leftarrow \delta \quad \forall\, e, f \in E, e \neq f$ ;
$work(e) \leftarrow 0 \quad \forall\, e \in E$ ;
$bkp(e, f) \leftarrow 0 \quad \forall\, e, f \in E, e \neq f$ ;
$G \leftarrow 0$ ;

while $G < 1$ do
    For each $e = (i, j) \in E$, compute shortest path
    from $i$ to $j$ that excludes link $e$ under link costs
    $c(e') = w(e', e)$ and denote its cost by $g(e)$ ;
    For each $i, j \in N$, compute shortest path from
    $i$ to $j$ under link costs $c(e) = [\sum_{f:f \neq e} w(e, f)+$
    $g(e)]$ and denote its cost by $SP(i, j)$ ;
    $V(k) \leftarrow \sum_{i \neq k} R_i SP(i, k) + \sum_{j \neq k} C_j SP(k, j)$ ;
    $G \leftarrow \min_{k \in N} V(k)$ ;
    if $G \geq 1$ break ;
    Let $\bar{k}$ be the node for which $V(k)$ is minimum ;
    Let $P_i$ be shortest path from $i$ to $\bar{k}$ for all $i$ ;
    Let $Q_j$ be shortest path from $\bar{k}$ to $j$ for all $j$ ;
    $N_P(e) \leftarrow \{i : P_i$ contains $e\}$ for all $e$;
    $N'_P(e, f) \leftarrow \{i : B_f(P_i)$ contains $e\}$ for all $e \neq f$;
    $N_Q(e) \leftarrow \{j : Q_j$ contains $e\}$ for all $e$;
    $N'_Q(e, f) \leftarrow \{i : B_f(Q_j)$ contains $e\}$ for all $e \neq f$;
    $\alpha \leftarrow \min_{e \in E} \frac{u_e}{\sum_{i \in N_P(e)} R_i + \sum_{j \in N_Q(e)} C_j}$ ;
    Send $\alpha R_i$ flow on path $P_i$ for all $i$ and
    $\alpha C_j$ flow on path $Q_j$ for all $j$ and compute
    resulting working flow $\Delta(e)$ on link $e$ for all $e$
    and restoration flow $\Delta'(e, f)$ on link $e$ due to
    failure of link $f$ for all $e \neq f$ ;
    $work(e) \leftarrow work(e) + \Delta(e) \ \ \forall\, e$ ;
    $bkp(e, f) \leftarrow bkp(e, f) + \Delta'(e, f) \ \ \forall\, e \neq f$ ;
    $w(e, f) \leftarrow w(e, f)(1 + \epsilon[\Delta(e) + \Delta'(e, f)]/u_e)$
                                                    $\forall\, e \neq f$ ;
    $\alpha_{\bar{k}} \leftarrow \alpha_{\bar{k}} + \alpha$ ;
end while

$bkp\_max(e) \leftarrow \max_{f \neq e} bkp(e, f) \ \ \forall\, e \in E$ ;
$scale(e) \leftarrow \frac{work(e) + bkp\_max(e)}{u_e} \ \ \forall\, e \in E$ ;
$scale\_max \leftarrow \max_{e \in E} scale(e)$ ;
$\alpha_k \leftarrow \alpha_k / scale\_max$ for all $k \in N$ ;
Output the traffic split ratios $\alpha_k$ ;
```

The pseudo-code for the above procedure, called Algorithm LINK_RESTORATION, is provided in the box above. Arrays $work(e)$ and $bkp(e, f)$ keep track respectively of the working traffic on link e and the restoration traffic that appears on link e due to failure of link f. The variable G is initialized to 0 and remains < 1 as long as the dual constraints remain unsatisfied. After the **while** loop terminates, the factor by which the capacity constraint on each link e gets violated is computed into array $scale(e)$. Finally, the α_i values are divided by the maximum capacity violation factor and the resulting values output as the optimum.

Let $L = (n-1)(n+m-2)(\sum_{i \in N} R_i + \sum_{j \in N} C_j)$ and L' denote the minimum non-zero value of the R_i's and C_j's. The values of ϵ and δ are related, in the following theorem, to the approximation factor guarantee of Algorithm LINK_RESTORATION.

Theorem 1: For any given $\epsilon' > 0$, Algorithm LINK_RESTORATION computes a solution with objective function value within $(1+\epsilon')$-factor of the optimum for

$$\delta = \frac{1+\epsilon}{L'[(1+\epsilon)\frac{L}{L'}]^{1/\epsilon}} \quad \text{and} \quad \epsilon = 1 - \frac{1}{\sqrt{1+\epsilon'}}$$

We end this section with a bound on the running time of Algorithm LINK_RESTORATION.

Theorem 2: For any given $\epsilon > 0$ chosen to provide the desired approximation factor guarantee in accordance with Theorem 1, Algorithm LINK_RESTORATION runs in time

$$O\left(\frac{m^3}{\epsilon}(m + n \log n) \log_{1+\epsilon} \frac{L}{L'}\right)$$

which is polynomial in the network size, the number of bits used to represent the R_i, C_j values, and $\frac{1}{\epsilon}$.

Proofs of the above theorems are omitted for lack of space.

VII. Adding Path Restoration to Two-Phase Routing

In order to make two-phase routing resilient to link failures using path restoration, Phase 1 and Phase 2 traffic is routed along link-disjoint path sets as discussed in Section V-B. Given a network with link capacities and constraints R_i, C_j on the ingress-egress traffic as discussed, we consider the problem of two-phase routing with path restoration so as to minimize the maximum utilization of any link in the network. The problem is equivalent to finding the maximum multiplier λ (throughput) such that all matrices in $\lambda \cdot \mathcal{T}(\mathcal{R}, \mathcal{C})$ can be feasibly routed with path restoration.

We first present a path indexed linear programming formulation for this problem. This will be subsequently used to develop the fast combinatorial algorithm in Section VII-B.

A. Path Indexed Linear Programming Formulation

Let $\mathcal{P}_{ij}^K$ denote the set of all K link-disjoint path sets from node i to node j. Let $x(P)$ denote the traffic associated with *each* (link-disjoint) path in the set P. Let $\chi(P)$ denote the number of link-disjoint paths in P. Then, the working traffic that is carried on P is $(\chi(P)-1)x(P)$. The problem of routing with path restoration under the scheme so as to maximize the network throughput can be formulated as the following path-indexed linear program:

$$\text{maximize} \quad \textstyle\sum_{i \in N} \alpha_i$$

subject to

$$\sum_K \sum_{P \in \mathcal{P}_{ij}^K} (K-1)x(P) = \alpha_j R_i + \alpha_i C_j \quad \forall\, i,j \in N \qquad (6)$$

$$\sum_{i,j \in N} \sum_K \sum_{P \in \mathcal{P}_{ij}^K, e \in P} x(P) \le u_e \quad \forall\, e \in E \qquad (7)$$

$$x(P), \alpha_i \ge 0 \quad \forall\, P \in P_{ij}^K, \quad \forall\, K, \quad \forall\, i,j \in N \qquad (8)$$

Constraints (6) correspond to the routing of $\alpha_j R_i + \alpha_i C_j$ amount of demand from node i to node j along link-disjoint path sets. Constraints (7) are the link capacity constraints. Similarly to the formulation for the link restoration case, the throughput is the sum of the traffic split ratios α_i *when these ratios are not constrained to sum to 1.*

In Section VII-B, we state the dual of the linear program. The usefulness of the primal and dual formulation is in designing a fast (polynomial time) combinatorial algorithm for the problem.

B. Combinatorial Algorithm

In this section, we develop a fast combinatorial algorithm (FPTAS) for two-phase routing with link restoration that can compute the split ratios and routed paths up to $(1+\epsilon)$-factor of the optimal objective function value for any $\epsilon > 0$. The value of ϵ can be chosen to provide the desired degree of optimality for the solution.

We begin with the dual formulation of the linear program outlined in Section VII-A. The dual program associates a variable $w(e)$ with each link capacity constraint in (7) and a variable π_{ij} with each demand constraint in (6). For any $i, j \in N$, define $DP(i,j)$ as

$$DP(i,j) = \min_K \min_{P \in \mathcal{P}_{ij}^K} \frac{\sum_{e \in P} w(e)}{K-1}$$

After simplification and removal of the dual variables π_{ij}, the dual linear program can be written as:

$$\text{minimize} \quad \textstyle\sum_{e \in E} u_e w(e)$$

subject to

$$\sum_{i: i \neq k} R_i DP(i,k) \quad + \quad \sum_{j: j \neq k} C_j DP(k,j) \ge 1 \quad \forall\, k \in N \qquad (9)$$

$$w(e) \quad \ge \quad 0 \quad \forall\, e \in E \qquad (10)$$

For any node $k \in N$, let $U(k)$ denote the left-hand-side (LHS) of constraint (9). Given the weights $w(e)$, note that $DP(i,j)$ can be computed in polynomial time using, for example, a successive shortest paths based K-disjoint paths routing algorithm [4]. Thus, $U(k)$ can be computed in polynomial time for all $k \in N$.

Given a set of weights $w(e)$, it is a feasible solution for the dual program if and only if

$$\min_{k \in N} U(k) \ge 1$$

The algorithm works as follows. Start with equal initial weights $w(e) = \delta$ (the quantity δ depends on ϵ and is derived later). Repeat the following until the dual feasibility constraints are satisfied:

1) Compute the node $k = \bar{k}$ for which $U(k)$ is minimum. This identifies a node $\bar{k}$ as well as link-disjoint path set P_i from node i to node $\bar{k}$ for all i and link-disjoint path set Q_j from node $\bar{k}$ to node j for all j.
2) For each $e \in E$, let $Y_P(e)$ be the set of nodes i for which P_i contains link e and $Y_Q(e)$ be the set of nodes

j for which Q_j contains link e. Compute the quantity α as follows:

$$S(e) = \sum_{i \in Y_P(e)} \frac{R_i}{\chi(P_i) - 1} + \sum_{j \in Y_Q(e)} \frac{C_j}{\chi(Q_j) - 1} \quad \forall\, e \in E$$

$$\alpha = \min_{e \in E} \frac{u_e}{S(e)}$$

3) Send αR_i amount of working flow from node i to node $\bar{k}$ along path set P_i for all i (this sends a flow of value $\alpha R_i/(\chi(P_i)-1)$ on each path in P_i). Send αC_j amount of working flow from node $\bar{k}$ to node j along path set Q_j for all j (this sends a flow of value $\alpha C_j/(\chi(Q_j)-1)$ on each path in Q_j). Compute the total flow $\Delta(e)$ that is sent on link e for all $e \in E$. Increment the flow on link e by $\Delta(e)$.
4) Update the weights $w(e)$ for all $e \in E$ as

$$w(e) \leftarrow w(e)\left(1 + \frac{\epsilon \Delta(e)}{u_e}\right)$$

5) Increment the split ratio $\alpha_{\bar{k}}$ associated with node $\bar{k}$ by α .

When the above procedure terminates, dual feasibility constraints will be satisfied. However, primal capacity constraints on each link will be violated, since we were working with the original (and not residual) link capacities at each stage. To remedy this, we scale down the split ratios α_i uniformly so that capacity constraints are obeyed.

```
Algorithm PATH_RESTORATION:

$\alpha_k \leftarrow 0 \quad \forall\, k \in N$ ;
$w(e) \leftarrow \delta \quad \forall\, e \in E$ ;
$flow(e) \leftarrow 0 \quad \forall\, e \in E$ ;
$G \leftarrow 0$ ;

while $G < 1$ do
    For all $i, j \in N$, compute link-disjoint path set $P$
    from $i$ to $j$ such that $\sum_{e \in P} \frac{w(e)}{\chi(P)-1}$ is minimized
    and denote this value by $DP(i,j)$ ;
    $U(k) \leftarrow \sum_{i \neq k} R_i DP(i,k) + \sum_{j \neq k} C_j DP(k,j)$ ;
    $G \leftarrow \min_{k \in N} U(k)$ ;
    if $G \geq 1$ break ;
    Let $\bar{k}$ be the node for which $U(k)$ is minimum ;
    Let $P_i$ be associated path set from $i$ to $\bar{k} \quad \forall\, i$ ;
    Let $Q_j$ be associated path set from $\bar{k}$ to $j \quad \forall\, j$ ;
    $Y_P(e) \leftarrow \{i : P_i$ contains $e\}$ for all $e$;
    $Y_Q(e) \leftarrow \{j : Q_j$ contains $e\}$ for all $e$;
    $S(e) \leftarrow \sum_{i \in Y_P(e)} \frac{R_i}{\chi(P_i)-1} + \sum_{j \in Y_Q(e)} \frac{C_j}{\chi(Q_j)-1} \quad \forall\, e$ ;
    $\alpha \leftarrow \min_{e \in E} \frac{u_e}{S(e)}$ ;
    Send $\alpha R_i$ working flow on $P_i$ for all $i$ and
    $\alpha C_j$ working flow on $Q_j$ for all $j$ and compute
    resulting capacity usage $\Delta(e)$ on link $e$ for all $e$ ;
    $flow(e) \leftarrow flow(e) + \Delta(e)$ for all $e$ ;
    $w(e) \leftarrow w(e)(1 + \epsilon\Delta(e)/u_e)$ for all $e$ ;
    $\alpha_{\bar{k}} \leftarrow \alpha_{\bar{k}} + \alpha$ ;
end while

$scale(e) \leftarrow flow(e)/u_e$ for all $e \in E$ ;
$scale\_max \leftarrow \max_{e \in E} scale(e)$ ;
$\alpha_k \leftarrow \alpha_k / scale\_max$ for all $k \in N$ ;
Output the traffic split ratios $\alpha_k$ ;
```

The pseudo-code for the above procedure, called Algorithm PATH_RESTORATION, is provided in the box below. Array $flow(e)$ keeps track of the traffic on link e. The variable G is initialized to 0 and remains < 1 as long as the dual constraints remain unsatisfied. After the **while** loop terminates, the factor by which the capacity constraint on each link e gets violated is computed into array $scale(e)$. Finally, the α_i values are divided by the maximum capacity violation factor and the resulting values output as the optimum.

Let $L = m(\sum_{i \in N} R_i + \sum_{j \in N} C_j)$ and L' denote the minimum non-zero value of the R_i's and C_j's. The values of ϵ and δ are related, in the following theorem, to the approximation factor guarantee of Algorithm PATH_RESTORATION.

Theorem 3: For any given $\epsilon' > 0$, Algorithm PATH_RESTORATION computes a solution with objective function value within $(1+\epsilon')$-factor of the optimum for

$$\delta = \frac{1+\epsilon}{L'[(1+\epsilon)\frac{L}{L'}]^{1/\epsilon}} \quad \text{and} \quad \epsilon = 1 - \frac{1}{\sqrt{1+\epsilon'}}$$

We end this section with a bound on the running time of Algorithm PATH_RESTORATION. Let K_{max} be the maximum number of link-disjoint paths between any pair of nodes in G.

Theorem 4: For any given $\epsilon > 0$ chosen to provide the desired approximation factor guarantee in accordance with Theorem 1, Algorithm PATH_RESTORATION runs in time

$$O\left(\frac{n^2 m}{\epsilon} K_{max}(m + n \log n) \log_{1+\epsilon} \frac{L}{L'}\right)$$

which is polynomial in the network size, the number of bits used to represent the R_i, C_j values, and $\frac{1}{\epsilon}$.

VIII. Evaluation on ISP Topologies

In this section, we evaluate the throughput performance of the unprotected, link restoration protected, and path restoration protected versions of two-phase routing. To compute the throughput for two-phase routing with link restoration and path restoration, we use the fast combinatorial algorithms from Sections VI-B and VII-B respectively. For the throughput of the unprotected scheme, we use the fast combinatorial algorithm from [11]. We run all algorithms so as to provide solutions up to 5% of optimality. The running times range from tens of seconds to few minutes on a Pentium III 1GHz 256MB machine.

A. Topologies and Link/Ingress-Egress Capacities

For our experiments, we use six ISP topologies collected by Rocketfuel, an ISP topology mapping engine [13]. These topologies list multiple intra-PoP (Point of Presence) routers and/or multiple intra-city PoPs as individual nodes. We coalesced PoPs into nodes corresponding to cities so that the topologies represent geographical PoP-to-PoP ISP topologies. Some data about the original Rocketfuel topologies and their coalesced versions is provided in Table I.

Link capacities, which are required to compute the maximum throughput, are not available for these topologies. Rocketfuel computed OSPF/IS-IS link weights for the topologies so that shortest cost paths match observed routes. In order to deduce the link capacities from the weights, we assumed that the given link weights are the default setting for OSPF weights in Cisco routers, i.e., inversely proportional to the link capacities [5]. The link capacities obtained in this manner turned out to be symmetric, i.e., $u_{ij} = u_{ji}$ for all $(i,j) \in E$.

Topology	Routers	Links	PoPs	Links
	(original)	(inter-router)	(coalesced)	(inter-PoP)
Telstra (Australia) 1221	108	306	57	59
Sprintlink (US) 1239	315	1944	44	83
Ebone (Europe) 1755	87	322	23	38
Tiscali (Europe) 3257	161	656	50	88
Exodus (Europe) 3967	79	294	22	37
Abovenet (US) 6461	141	748	22	42

TABLE I

Rocketfuel topologies: Original number of routers and inter-router links, and number of coalesced PoPs and inter-PoP links.

There is also no available information on the ingress-egress traffic capacities at each node. Because ISPs commonly engineer their PoPs to keep the ratio of add/drop and transit traffic approximately fixed, we assumed that the ingress-egress capacity at a node is proportional to the total capacity of network links incident at that node. We also assume that $R_i = C_i$ for all nodes i since network routers and switches have bidirectional ports (line cards) – hence the ingress and egress capacities are equal. Thus, we have $R_i (= C_i) \propto \sum_{e \in E^+(i)} u_e$.

The coalesced Rocketfuel topologies are not bi-connected and hence do not allow diverse link detours for some links and link-disjoint paths between some source-destination pairs. Any graph that is not bi-connected has one or more *bridge* links whose removal disconnects the graph into two connected components. We overcome this limited connectivity of the topologies by splitting bridge links into two diverse links, each of half the capacity as the original link. Because the original Rocketfuel topologies contained many parallel links that were coalesced, this bridge splitting transformation preserves the essential ISP-like topological properties of the networks. Also, the throughput of unprotected routing remains unchanged as a result of this transformation.

B. Experiments and Results

We denote the throughput values for the different cases as follows: (i) λ_{unp} for unprotected, (ii) λ_{lr} for link restoration, and (iii) λ_{pr} for path restoration. Clearly, $\lambda_{unp} > \lambda_{lr}$ and $\lambda_{unp} > \lambda_{pr}$. We are also interested in the number of intermediate nodes i with $\alpha_i > 0$, which we denote for the three cases by N_{unp}, N_{lr}, and N_{pr}.

1) Throughput: In Table II, we list the lambda values for the three cases for the six Rocketfuel topologies. When either the link capacities or ingress-egress capacities are scaled by a constant, the throughput values are scaled by the same constant. Hence, for comparison purposes, we have normalized the values so that the throughput for the unprotected case is $\lambda_{unp} = 3.0$.

Topology	λ_{unp}	λ_{lr}	λ_{pr}
Telstra (Australia) 1221	3.0	1.3265	1.3245
Sprintlink (US) 1239	3.0	1.8642	1.6800
Ebone (Europe) 1755	3.0	1.1565	1.1289
Tiscali (Europe) 3257	3.0	1.8882	1.5984
Exodus (Europe) 3967	3.0	1.6383	1.6170
Abovenet (US) 6461	3.0	1.2957	1.2552

Table II. Throughput of two-phase routing for unprotected (λ_{unp}), and link restoration (λ_{lr}), path restoration (λ_{pr}) extensions.

The overhead of protecting against link failures can be measured by the percentage decrease in network throughput over that for the unprotected case. For link restoration, this is $O_{lr} = (\lambda_{unp} - \lambda_{lr})/\lambda_{unp}$. For path restoration, this is $O_{pr} = (\lambda_{unp} - \lambda_{pr})/\lambda_{unp}$. These values are listed in Table III.

For link restoration, the overhead ranges from 35-60% for the six topologies. For path restoration, the overhead ranges from 45-60% for the six topologies. Both overheads are relatively high because of the limited diversity available in these six topologies. Also, in all cases, we have $O_{pr} > O_{lr}$. Because path restoration shares its backup capacity among link-disjoint paths, it is more constrained by the physical diversity of the network than link restoration. Hence, the increased overhead of path restoration.

Topology	O_{lr}	O_{pr}
Telstra (Australia) 1221	55.78%	55.85%
Sprintlink (US) 1239	37.86%	44.00%
Ebone (Europe) 1755	61.45%	62.37%
Tiscali (Europe) 3257	37.06%	46.72%
Exodus (Europe) 3967	45.39%	46.10%
Abovenet (US) 6461	56.81%	58.16%

Table III. Overhead of link restoration (O_{lr}) and path restoration (O_{pr}) compared to unprotected case for two-phase routing.

2) Number of Intermediate Nodes: In Table IV, we list the number of intermediate nodes i with $\alpha_i > 0$ for the three cases for the six Rocketfuel topologies. Interestingly, the number of intermediate nodes in each case is a small fraction of the total number of nodes. This may have favorable implications in the adaptation of the scheme to providing i3-like functionality and middlebox routing (e.g., content filtering) in networks. In these two application scenarios, the intermediate nodes are sites for locating i3 servers and middleboxes respectively.

Topology	N_{unp}	N_{lr}	N_{pr}
Telstra (Australia) 1221	1	1	1
Sprintlink (US) 1239	5	6	4
Ebone (Europe) 1755	4	5	3
Tiscali (Europe) 3257	7	6	2
Exodus (Europe) 3967	3	7	3
Abovenet (US) 6461	7	6	1

Table IV. Number of intermediate nodes in two-phase routing for unprotected (N_{unp}), and link restoration (N_{lr}), path restoration (N_{pr}) extensions.

The number of intermediate nodes for link restoration is the same as that for the unprotected case for almost all the topologies. However, we observe a marked decrease in the number of intermediate node for the path protection case.

This is again because of the limited diversity available in the six topologies. For path restoration, the algorithm intelligently selects only those intermediate nodes that have sufficient path diversity to many other nodes in the network. In contrast, link restoration involves the restoration of working traffic on a link in a *local manner* – it is much less constrained by global diversity in the selection of intermediate nodes.

IX. Conclusion and Future Work

The two-phase routing scheme was recently proposed for routing highly dynamic and changing traffic patterns with bandwidth guarantees and in a traffic-oblivious manner. It allows preconfiguration of the network such that all traffic patterns, permissible within the network's natural ingress-egress capacity constraints, can be handled in a capacity efficient manner *without the necessity to detect any traffic changes in real-time*. For the Rocketfuel topologies, the throughput of two-phase routing is within 6% of the optimal scheme among the class of all schemes that are allowed to reconfigure the network in response to traffic changes [11].

In this paper, we extended the two-phase routing scheme by providing resiliency against link failures through two different *fast restoration* mechanisms – local (link/span) based and end-to-end (path) based. We developed combinatorial algorithms for determining optimal traffic split ratios and routing along primary and backup paths for the two restoration mechanisms. We view this as important progress for two-phase routing towards achieving carrier-class reliability so as to facilitate its future deployment in ISP networks.

One would ideally like the network to be quasi-static in its configuration and not require frequent adaptation to network events. ISPs typically use a combination of overprovisioning and dynamic network adaptation to avoid network congestion caused by unpredicted events. However, both overprovisioning and frequent adaptation lead to increased costs. In particular, frequent adaptation incurs high operational costs and risks further instability elsewhere in the network. Two-phase routing, with the extensions for restoration proposed in this paper, can handle extreme traffic variability and link failures in a network with a static network configuration and without requiring high capacity overprovisioning. The ability to handle traffic variation in a failure resilient manner without any routing adaptation will lead to more stable and robust Internet behavior.

We evaluated the throughput performance of two-phase routing with link restoration and path restoration on actual ISP topologies taken from the RocketFuel project and compared it with the unprotected case. We also looked at the number of intermediate nodes for all cases.

We have also considered another restoration mechanism, shared backup path restoration, for protecting against link failures in two-phase routing [15]. Under this, backup paths can share bandwidth on common links so long as their primary paths are link disjoint. Thus, backup bandwidth is shared to provide completely recovery against single failures. Shared backup path restoration has been shown to have lower restoration capacity overhead compared to the two mechanisms considered in this paper.

The handling of node failures in two-phase routing poses additional challenges. Failure of non-intermediate nodes lying on Phase 1 or Phase 2 paths can be restored by using node detours in link restoration or node-disjoint paths in path restoration. The failure of intermediate nodes is naturally accommodated in two-phase routing by redistributing traffic split ratios to other intermediate nodes, as proposed in [12]. Because a single node failure can lead to both of the above scenarios, the corresponding mechanisms can be integrated.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, February 1993.

[2] D. Applegate, L. Breslau, and E. Cohen, "Coping with Network Failures: Routing Strategies for Optimal Demand Oblivious Restoration", *ACM SIGMETRICS/Performance 2004*, June 2004.

[3] D. Applegate and E. Cohen, "Making Intra-Domain Routing Robust to Changing and Uncertain Traffic Demands: Understanding Fundamental Tradeoffs", *ACM SIGCOMM 2003*, August 2003.

[4] R. Bhandari, *Survivable Networks: Algorithms for Diverse Routing*, Kluwer International, January 1999.

[5] "Configuring OSPF", Cisco Systems Product Documentation, http://www.cisco.com/univercd/home/home.htm.

[6] N. G. Duffield, P. Goyal, A. G. Greenberg, P. P. Mishra, K. K. Ramakrishnan, J. E. van der Merwe, "A flexible model for resource management in virtual private network", *ACM SIGCOMM 1999*, August 1999.

[7] J. A. Fingerhut, S. Suri, and J. S. Turner, "Designing Least-Cost Nonblocking Broadband Networks", *Journal of Algorithms*, 24(2), pp. 287-309, 1997.

[8] N. Garg and J. Könemann, "Faster and Simpler Algorithms for Multicommodity Flow and other Fractional Packing Problems", *39th Annual Symposium on Foundations of Computer Science (FOCS)*, 1998.

[9] A. Kumar, R. Rastogi, A. Silberschatz , B. Yener, "Algorithms for provisioning VPNs in the hose model", *ACM SIGCOMM 2001*, August 2001.

[10] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Efficient and Robust Routing of Highly Variable Traffic", *Third Workshop on Hot Topics in Networks (HotNets-III)*, November 2004.

[11] M. Kodialam, T. V. Lakshman, and S. Sengupta, "A Versatile Scheme for Routing Highly Variable Traffic in Service Overlays and IP Backbones", *IEEE Infocom 2006*, April 2006.

[12] M. Kodialam, T. V. Lakshman, J. B. Orlin, and S. Sengupta, "Preconfiguring IP-over-Optical Networks to Handle Router Failures and Unpredictable Traffic", *IEEE Infocom 2006*, April 2006.

[13] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP Topologies with Rocketfuel", *IEEE/ACM Transactions on Networking*, vol. 12, no. 1, pp. 2-16, February 2004.

[14] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, C. Diot, "Traffic Matrix Estimation: Existing Techniques and New Directions", *ACM SIGCOMM 2002*, August 2002.

[15] Sudipta Sengupta, *Efficient and Robust Routing of Highly Variable Traffic*, Ph.D. Thesis, Massachusetts Institute of Technology (MIT), December 2005.

[16] S. Sengupta and R. Ramamurthy, "From Network Design to Dynamic Provisioning and Restoration in Optical Cross-Connect Mesh Networks: An Architectural and Algorithmic Overview", *IEEE Network Magazine*, vol. 15, No. 4, July/August 2001.

[17] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, S. Surana, "Internet Indirection Infrastructure", *ACM SIGCOMM 2002*, August 2002.

[18] R. Zhang-Shen and N. McKeown "Designing a Predictable Internet Backbone Network", *Third Workshop on Hot Topics in Networks (HotNets-III)*, November 2004.