Efficient and Robust Routing of Highly Variable Traffic

by

Sudipta Sengupta

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

December 2005
[February 2006]

Author .....
Department of Electrical Engineering and Computer Science
December 14, 2005

Certified by .....
James B. Orlin
Edward Pennell Brooks Professor of Operations Research
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Efficient and Robust Routing of Highly Variable Traffic

by

Sudipta Sengupta

Submitted to the Department of Electrical Engineering and Computer Science
on December 14, 2005, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

Many emerging applications for the Internet are characterized by highly variable traffic behavior over time that is difficult to predict. Classical approaches to network design rely on a model in which a single traffic matrix is estimated. When actual traffic does not conform to such assumptions, desired bandwidth guarantees cannot be provided to the carried traffic. Currently, Internet Service Providers (ISPs) use gross capacity over-provisioning and manual routing adaptation to avoid network congestion caused by unpredictable traffic. These lead to increased network equipment and operational costs. Development of routing infrastructures that optimize network resources while accommodating extreme traffic unpredictability in a robust and efficient manner will be one of the defining themes in the next phase of expansion of the Internet.

This thesis proposes *two-phase routing* as a capacity efficient and robust strategy for handling highly variable traffic. The scheme allows preconfiguration of the network such that all traffic patterns permissible within the network's natural ingress-egress capacity constraints can be routed with bandwidth guarantees *without requiring detection of traffic changes in real-time or reconfiguring the network in response to it.* The scheme routes traffic in two phases – traffic entering the network is sent from the source to a set of intermediate nodes in predetermined split ratios that depend on the intermediate nodes, and then from the intermediate nodes to the final destination. The scheme has the desirable properties of supporting static optical layer provisioning in IP-over-Optical networks and indirection in specialized service overlay models unlike previous approaches – like direct source-destination path routing – for handling variable traffic.

This thesis represents the first comprehensive study, problem formulation, and algorithm design for many aspects of two-phase routing. Our contributions can be grouped into three broad parts. First, we consider the problems of minimum cost network design and maximum throughput network routing for the scheme. We give a simple solution for minimum cost network design. For maximum throughput network routing, we design linear programming based and combinatorial algorithms. We

show how the algorithms can handle a total cost constraint for maximum throughput two-phase routing. This can be used to solve the link capacitated version of minimum cost two-phase routing. We establish theoretical bounds on the resource requirements of two-phase routing under throughput and cost models with respect to the optimal scheme that is allowed to make the routing dynamically dependent on the current traffic matrix. We also generalize the traffic split ratios to depend not only on the intermediate nodes but also on source and destination of traffic and solve the corresponding optimization problems.

Second, we consider making two-phase routing resilient to network failures. Two-phase routing in IP-over-Optical networks can be protected against router node failures through redistribution of traffic split ratio for the failed router node to other intermediate nodes. We propose two different schemes for provisioning the optical layer to handle router node failures. We develop linear programming formulations for both schemes and a fast combinatorial algorithm for the second scheme so as to maximize network throughput. Two-phase routing can be made resilient against link failures by protecting the first and second phase paths using pre-provisioned restoration mechanisms. We consider three such restoration mechanisms – local (link/span) restoration, K-route path restoration, and shared backup path restoration. We provide linear programming formulations and combinatorial algorithms for maximum throughput two-phase routing with local restoration and K-route path restoration. We show that the problem of maximum throughput two-phase routing with shared backup path restoration is $\mathcal{NP}$-hard. Assuming an approximation oracle for a certain disjoint paths problem (which we also show to be $\mathcal{NP}$-hard), we design a combinatorial algorithm with provable guarantees.

Third, we consider the application of two-phase routing to multi-hop Wireless Mesh Networks (WMNs). These networks have recently been of much research interest due to their lowered need for wired infrastructure support and due to envisaged new applications like community wireless networks. We extend our optimization framework for maximum throughput two-phase routing in wired networks to handle routing and scheduling constraints that are peculiar to WMNs and arise from the requirement to handle radio transmit/receive diversity and the phenomenon of wireless link interference.

We evaluate various aspects of two-phase routing on actual ISP topologies using the developed algorithms. For the WMN application, we use randomly generated WMN topologies for the evaluations.

Thesis Supervisor: James B. Orlin
Title: Edward Pennell Brooks Professor of Operations Research

Acknowledgments

I am grateful and deeply indebted to my advisor Prof. James B. Orlin for his guidance, support, and encouragement all along the way. He made himself frequently available for discussions and provided many insights on the research. I consider myself fortunate to have had him as my advisor.

I am thankful to Dr. M. Kodialam and Dr. T. V. Lakshman for suggesting the research topic and for their collaboration.

I thank Prof. Hari Balakrishnan and Prof. John N. Tsitsiklis for agreeing to be on my thesis committee and for providing feedback on the thesis and presentation.

I am thankful to my family – my parents, for making many sacrifices so that I could set and and achieve higher goals in academics and for insisting that I complete my Ph.D.; my brother, for his confidence in me and for always looking up to me; my late grandparents, who would have been the happiest persons to see me finish my Ph.D.; my grandmother for her affection and concern; and my wife, for being very supportive and helpful and for patiently bearing with my unavailability during the doctoral program. They laid the foundations on which this thesis rests.

Bibliographic Note

Parts of this thesis have been published as follows. Portions of the content of Chapter 2 appear in the paper [KLS04b] in *ACM Hot Topics in Networks (HotNets) 2004*. Portions of the contents of Chapters 4, 5, and 6 will appear in three papers [KLOS06a, KLS06, KLOS06b] respectively in *IEEE Infocom 2006*.

To my parents

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the rapid rise in new networking applications and the increasing use of the Internet for carrying real-time traffic such as Voice-over-IP (VoIP) and peer-to-peer applications, it has become increasingly important to accommodate widely varying traffic patterns in networks. This requires Internet Service Providers (ISPs) to accurately monitor (or, forecast) traffic and to deploy mechanisms for quickly and repeatedly adapting their (intra-domain) network routing to changing traffic patterns so as to avoid network congestion. Tracking network traffic changes in real-time or making accurate traffic predictions are both difficult problems. Moreover, ISPs prefer to avoid frequent routing changes due to operational complexity and costs, and due to the risk of network instability if such changes are not implemented correctly.

An alternative is to have sufficient capacity set aside a priori to accommodate the different traffic patterns that can occur without resorting to routing changes. This is how traffic variation is mostly handled in current ISP networks and comes at the expense of significant capacity overprovisioning. Ideally, ISPs would like to use a fixed routing scheme that does not require traffic dependent dynamic adaptation and which, at the same time, is *efficient in its use of network capacity.*

This thesis proposes *two-phase routing* as a capacity efficient and robust scheme for handling highly variable traffic. The scheme routes traffic in two phases. In the

first phase, traffic entering the network is sent from the source to a set of intermediate nodes and then, in the second phase, from the intermediate nodes to the final destination. The traffic in the first phase is distributed to the intermediate nodes in predetermined proportions that depend on the intermediate nodes. The scheme provides bandwidth guarantees for traffic that can vary arbitrarily subject to the network's natural ingress-egress capacity constraints. Under the scheme, the network is preconfigured so that neither the paths nor their bandwidth need to be changed in response to changes in the traffic matrix.

The remainder of this introductory chapter is structured as follows. We begin by reviewing some causes for traffic variation in the current Internet in Section 1.1 so as to motivate the need to handle traffic variation in network routing. In order to get some idea of the real-world constraints imposed on the design of routing schemes for handling variable traffic, it is important to understand, from an ISP perspective, the difficulty of deploying and operating a dynamic routing architecture that requires the measurement of possibly changing traffic in real-time as well as reconfiguring the network in response to such changes in order to provide bandwidth guarantees. We address these aspects in Section 1.2.

We introduce basic notation used throughout this thesis in Section 1.3. The traffic variation model is described in Section 1.4. In Section 1.5, we describe some motivating networking applications and identify their requirements. In Section 1.6, we show why an earlier approach for routing variable traffic – direct source-destination routing along fixed paths – does not meet some requirements of the motivating applications. In Section 1.7, we describe the optimization models considered in this thesis and for each model, identify the optimal scheme among the class of all schemes that are allowed to dynamically change the routing in response to changes in network traffic. We will compare the performance of two-phase routing with that of the optimal scheme both in terms of theoretical worst case bounds and a posteriori bounds through computation on actual ISP topologies. The contributions and organization

of the thesis are summarized in Section 1.8.

1.1 Some Causes for Traffic Variation

Traffic variation may be caused by a variety of reasons. We discuss some of these below.

1.1.1 Host Mobility and Dynamic Communication

Mobile hosts can attach to different network nodes over time and use services over an overlay network. Applications like audio/video conferencing require dynamic membership of multicast groups. These cause changes in the point-to-point traffic demands seen by an overlay network providing services like the above.

1.1.2 IP Traffic Variation

Traffic variation can arise from the communication pattern of the application and its changing bandwidth needs over time. With an increasing number of services migrating to or new ones being deployed over IP (e.g., voice-over-IP, storage services, peer-to-peer networks, multi-user games), data traffic may be varying over shorter periods of time, e.g., intra-day instead of weeks or months. The sudden appearance of flash crowds responding to special events on the Internet may cause drastic changes in traffic to and from web sites hosting the events.

1.1.3 BGP Induced Traffic Variation

Internet traffic is routed from a source to a destination across different Autonomous System (AS) domains using the Border Gateway Protocol (BGP) to determine the sequence of AS domains traversed by a packet. Within an ISP domain, the packet is routed from the ingress to the egress using policies that are local to and controlled by

the ISP that owns the domain. Extreme network traffic fluctuations within an AS can happen due to external factors like BGP route changes across AS domains. Consider a large ISP exchanging traffic with several other providers. Typically, the traffic exchange between ISPs is specified by total traffic volumes over long time periods and possibly a peak rate limit (usually just determined by physical link capacities) [N01]. The actual distribution of traffic entering at an ingress to the various network egresses is not known and can change over time. This is because the distribution is determined by many factors such as intrinsic changes of traffic to different destination prefixes and by routing changes either made locally by the carrier or due to changes made in other ASes over which the carrier has no control.

An example of local routing changes that can affect the traffic distribution is IGP (Interior Gateway Protocol) weight changes combined with hot-potato routing that can change the network egress that traffic destined to a set of prefixes would choose. While local routing changes are under an ISP's control and hence change traffic patterns only at planned instants, unpredictable traffic shifts can happen when routing changes in other ASes affect downstream ASes. A recent study of the effects of the prevalent hot-potato routing [TSGR04] shows that IGP weight changes (which can be due to new links being added, maintenance, traffic engineering, etc.) in an AS can cause significant shifts in traffic patterns across ASes – examples are shown where changes in IGP costs can affect the BGP route for 40% of the prefixes, and Netflow measurements are shown to indicate that the affected prefixes can account for up to 35% of the traffic. This indicates that significant shifts in traffic may happen at a carrier due to changes elsewhere in the network.

The causes for traffic variation discussed above involve scenarios that share the following important aspect: *the traffic matrix is unknown and can vary unpredictably over time.* At any given point of time, 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. These are the only constraints imposed on the variability

of the traffic matrix and is captured by the traffic variation model that we consider in this thesis. We introduce the traffic variation model in Section 1.4.

1.2 Traffic Measurement and Dynamic Network Reconfiguration

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 contribution of this thesis is the development of a routing scheme that can handle traffic variability in a capacity efficient manner through static preconfiguration of the network and without requiring either (i) measurement of traffic in real-time or (ii) reconfiguration 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 importance of these imposed constraints on the routing.

1.2.1 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) [CFSD90] provides this 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 [MTSBD02, ZRLD03, ZRDG03]. Moreover, many of these methods are not suitable for measuring traffic in real-time

due to their computation intensive nature.

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% [IBTM03].

1.2.2 Difficulties in Dynamic Network Reconfiguration

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 management and 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 preconfiguration property of a routing scheme in handling traffic variability.

1.3 Basic Notation

Throughout this thesis, we will work with a directed graph $G = (N, E)$ with node set N and (directed) link (edge) set E that models the network topology. Each node in the network can be a source or destination of traffic. The graph is assumed to be connected in a directed sense, i.e., there is directed path between every pair of nodes. In order to protect against link failures (as in Chapter 7), we require that the network be bi-connected, i.e., the removal of any single link (and its reverse) does not disconnect the graph.

Let $|N| = n$ and $|E| = m$. The nodes in N are labeled $\{1, 2, \ldots, n\}$. The sets of incoming and outgoing links 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} and its cost by c_{ij}. The *utilization* of a link is defined as the maximum traffic usage on the link divided by its capacity. When considering network protection against a link (or, node) failure, the traffic on a link is the sum of working traffic and maximum restoration traffic due to any failure. We will introduce additional notation in several places in the thesis when dealing with the respective topics.

1.4 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 (egress) links at that node. This is known as the *hose model* and was proposed by Fingerhut et al . [FST97] and subsequently used by Duffield et al. [DGGMR99] 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 the network at node i by R_i and C_i respectively. This is illustrated in Figure 1-1. The point-to-point matrix for the traffic in the network could vary over time but is constrained by these aggregate ingress-egress 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

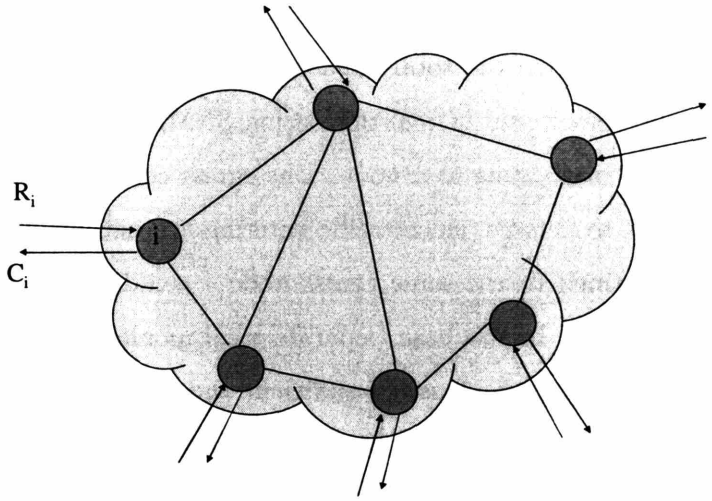


Figure 1-1: Traffic Variation Model.

is, any allowable traffic matrix $T = [t_{ij}]$ for the network must obey

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

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 $\mathcal{T}(\vec{R}, \vec{C})$, that is

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

At any point in time, the point-to-point traffic distribution could be any matrix in $\mathcal{T}(\vec{R}, \vec{C})$.

We will use $\lambda \cdot \mathcal{T}(\vec{R}, \vec{C})$ to denote the set of all traffic matrices in $\mathcal{T}(\vec{R}, \vec{C})$ with their entries multiplied by λ. Note that the traffic distribution T could be any matrix in $\mathcal{T}(\vec{R}, \vec{C})$ and could change over time. We will say that ingress-egress capacities are *symmetric* when $R_i = C_i$ for all node i and that ingress-egress capacities are *balanced* when the total ingress capacity equals the total egress capacity, i.e., $\sum_{i \in N} R_i = \sum_{j \in N} C_j$.

Conformance of network traffic to the ingress constraints is naturally enforced if the capacities R_i correspond to the total capacity of ingress links at each node i. Alternatively, it can be monitored in real-time using SNMP aggregate traffic measurements for the traffic originating at a node. The egress constraints, on the other hand, are more difficult to enforce, since traffic entering the network from different ingress nodes i and destined to the same egress node j could together violate the egress constraints at node j. In this case, separate local monitoring at each ingress node of the traffic to each egress node is not sufficient and may require, for example, cooperation among the ingress nodes in a distributed manner or through a centralized network management system. A routing scheme that provides bandwidth guarantees for the hose traffic model may not be able to avoid congestion within the network if the egress links are over-subscribed. However, this aspect may be moot, since there will be congestion for exiting traffic on the egress links anyways if egress link capacity bounds are violated.

1.5 Motivating Networking Applications and Their Requirements

We discuss some motivating networking architectures and applications that need to handle traffic variation and identify the requirements of a suitable routing scheme for each scenario. In the next section, we argue why such requirements are not met by existing routing methodologies. We will return to these networking scenarios in Chapter 2 when we describe the potential applications of our proposed scheme.

1.5.1 Internet Backbones

Core (long-haul) networks of ISPs form the backbone of the Internet and span vast geographical areas (countries and continents). Each node in such a network, also

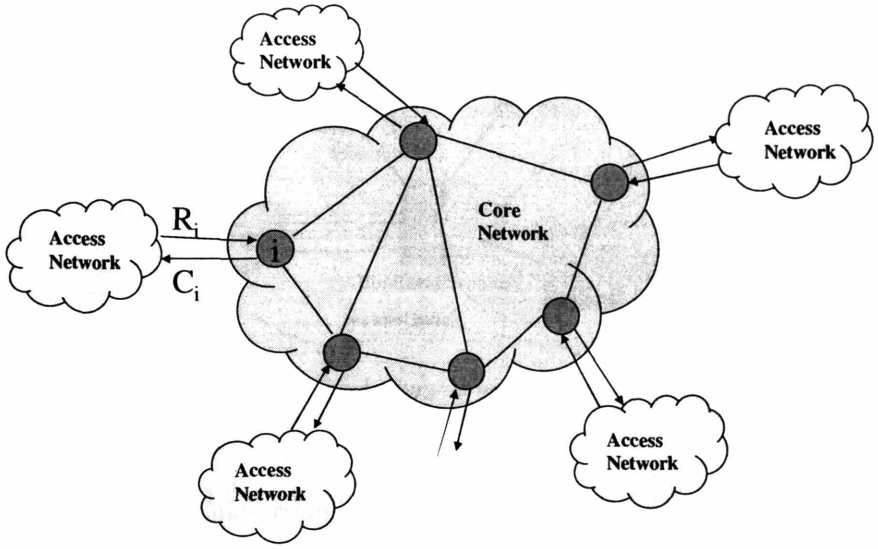


Figure 1-2: Core (long-haul) network interconnecting access networks.

called a Point-of-Presence (PoP), connects an access network (or, regional/metro network) to the core network, as shown in Figure 1-2. Internet backbones are often deployed by interconnecting routers over a switched optical backbone, also called IP-over-OTN (Optical Transport Network). This is illustrated in Figure 1-3.

Because a router line card is typically 3-4 times more expensive than an optical switch card, an IP-over-OTN architecture reduces network cost by keeping traffic mostly in the optical layer [SSK03]. Moreover, the ability of router technology to scale to port counts consistent with multi-terabit capacities without compromising performance, reliability, restoration speed, and software stability is questionable [RS99]. By removing transit traffic from the routers to the optical switches, the requirement to upgrade router PoP configurations with increasing traffic is minimized (since optical switches are more scalable with increasing port count than routers). Also, since optical switches are known to be much more reliable compared to routers [LAJ98], this makes the architecture more robust and reliable.

Routing in IP-over-OTN needs to make a compromise between keeping traffic at

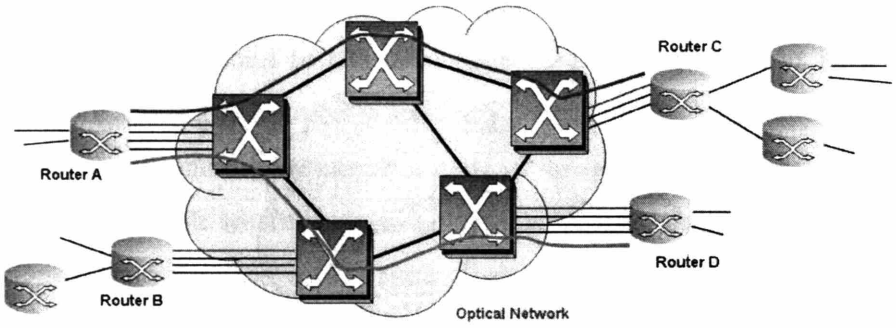


Figure 1-3: Routers interconnected over a switched optical backbone in IP-over-Optical networks.

the optical layer (for the above reasons) and using intermediate routers for packet grooming in order to achieve efficient statistical multiplexing of data traffic. In addition, the routing must be able to handle traffic variability. For reasons explained in Section 1.2, the (current) traffic matrix is not only difficult to estimate but changes in the same may not be detectable in real time. Moreover, 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 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. These translate to the following requirements on the routing methodology:

- IP traffic must be routed "mostly" at the optical layer from source to destination routers. Intermediate IP layer transit may be required for grooming purposes.
- The optical layer (circuits and their bandwidth) must be *statically provisioned a priori* to provide bandwidth guarantees for end-to-end IP traffic. Routing at the IP layer cannot also be adaptive to traffic changes.
- Bandwidth guarantees must be provided for routing all traffic matrices.

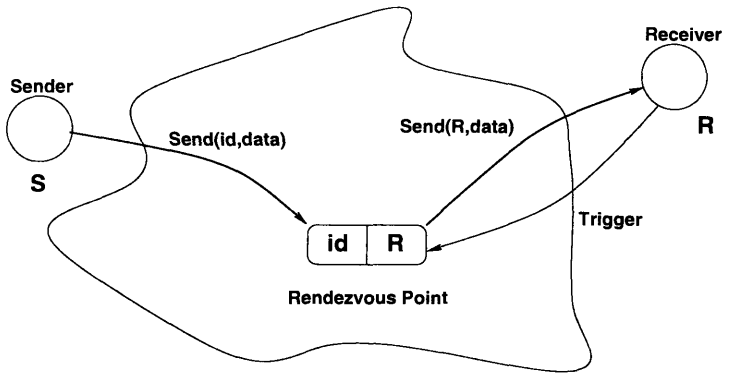


Figure 1-4: Operation of Rendezvous Based Communication (i3).

1.5.2 Specialized Service Overlay Networks

Specialized service overlay architectures like the Internet Indirection Infrastructure (i3) proposed by Stoica et al. [SAZSS02] can be used to ease the deployment of services – like mobility, multicast and anycast – on the Internet. i3 provides a rendezvous-based communication abstraction through indirection – sources send packets to a logical identifier, and receivers express interest in packets sent to a specific identifier. The rendezvous points are provided by i3 servers that forward packets to all receivers that express interest in a particular identifier. This is illustrated in Figure 1-4. The communication between senders and receivers is through these rendezvous points over an overlay network.

As shown in [SAZSS02], it is easy and natural to deploy multicasting, anycasting and mobility services on this rendezvous based infrastructure. Multicasting is illustrated in Figure 1-5 and is achieved by multiple receivers subscribing to the same logical identifier.

The i3 infrastructure does not store packets but only forwards them. It is important to note that i3 provides only a best-effort service like today's Internet – it neither implements reliability nor guarantees ordered delivery on top of IP. It might be desirable to support rendezvous based services *with bandwidth guarantees* in intra-ISP deployments of specialized service overlays like i3. A unique aspect of such service

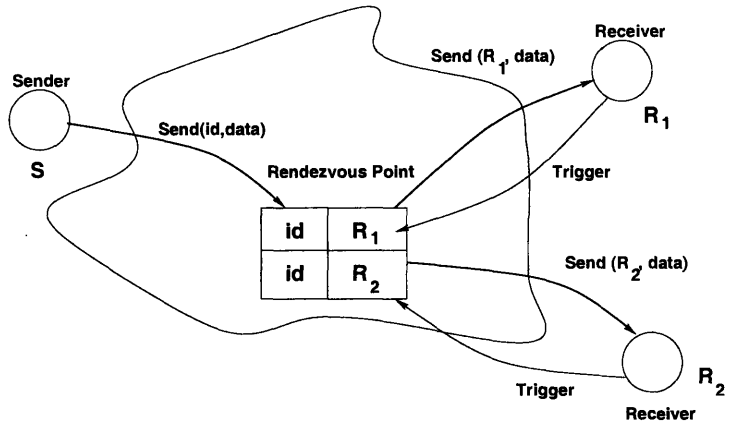


Figure 1-5: Multicasting in Rendezvous Based Communication (i3).

overlays arising from the indirection property is that unlike traditional networks, the final destination of a packet is not known at the network ingress. Hence, methods that set up pre-provisioned paths from a network's ingress to its egress nodes for providing bandwidth guarantees are not usable.

To support rendezvous based communication for variable traffic with bandwidth guarantees, we need the following:

- The routing from the source node(s) to the rendezvous nodes cannot depend on the final destination(s) of the packet, since this is unknown at the source.
- The traffic from the source node to the rendezvous nodes and from the latter to the destination nodes must be routed along bandwidth-guaranteed paths.
- To avoid reconfiguration, these paths cannot be re-routed in response to changes in traffic patterns, and must have sufficient bandwidth to handle all possible traffic patterns subject to network ingress-egress constraints.

1.5.3 Routing via Middleboxes

Intermediate network elements (so called middleboxes), such as firewalls and transparent caches, are now commonplace. They provide important services like caching,

load-balancing, and content filtering (for network security). To be effective, the services provided by such middleboxes are required to be comprehensive in the sense that every packet routed through the network must pass through at least one middlebox providing the service. In order to support a middlebox routing architecture, the routing scheme needs to not only provide bandwidth guarantees for variable traffic but also handle the additional constraint that all network traffic must pass through at least one intermediate network element node.

1.5.4 Other Applications of Interest

Another example application where the traffic matrix is unknown is the provisioning of network-based VPN services [BK03] to enterprise customers. VPNs typically provide network connectivity among different sites of an enterprise. The traffic distribution between the sites is not known a priori - it may also change depending on time-of-day, day-of-week, special activities, etc. The enterprise customer specifies to the ISP only the total traffic volume and the peak rate out of a given site (e.g., if a site is connected to the ISP through a T1 link, this peak rate is about 1.5 Mbps). It is the ISP's task to transport all of the offered VPN traffic to the network and carry the traffic in accordance with the bandwidth guarantees provided in the Service Level Agreement (SLA). The traffic originating from or destined to a VPN node is limited only by the aggregate bandwidth connection of that node to the VPN.

Networks for grid computing also need to handle highly variable traffic patterns. In grid computing, a complex computational task is partitioned amongst different computing nodes that can be geographically distributed and are connected by a network. The communication patterns amongst grid computing nodes are highly unpredictable and also can require high burst rates. Since the traffic matrix is not known. one option is to dynamically reserve capacity over an underlying network but this approach will be too slow for grid computing applications.

1.6 Direct Source-Destination Routing Along Fixed Paths

We briefly review related work on routing with traffic variability and point out why such existing methods cannot meet the requirements outlined in Section 1.5 for the various application scenarios.

Direct routing from source to destination along *fixed* paths for the hose traffic model has been considered by Duffield et al. [DGGMR99] and Kumar et al. [KRSY01]. In related work, Azar et al. [ACFKR03] consider direct source-destination routing along fixed paths and show how to compute *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.

In all of the above, *direct source-destination paths* are fixed a priori for routing the traffic between each source-destination pair. Thus, the source needs to *know the destination of a packet* for routing it, without which the source cannot determine the path along which the packet should be forwarded. In service overlay models like i3 that support indirection, the *final destination of a packet is not known at the source.* Thus, any of the above approaches cannot be used for routing in service overlay networks.

Another important property of direct source-destination routing renders it unsuitable for use in IP-over-Optical networks. Note that even though the paths are fixed a priori and do not depend on the traffic matrix, their *bandwidth requirements change* with variations in the traffic matrix. Thus, bandwidth needs to be deallocated from some paths and assigned to other paths as the traffic matrix changes. (Alternatively, paths between every source-destination pair can be provisioned a priori to handle the maximum traffic between them, but this leads to gross overprovisioning of capacity, since all source-destination pairs cannot simultaneously reach their peak traffic limit

in the hose traffic model.) Direct source-destination routing, when applied to IP-over-Optical networks, routes packets from source to destination along direct paths in the optical layer. This necessitates *dynamic reconfiguration* of the provisioned optical layer circuits (i.e., change in bandwidth) in response to traffic variations, which as explained earlier, is not possible in current ISP networks.

To illustrate this last point, consider the scenario in Figure 1-3 for direct source-destination routing in IP-over-Optical networks. Here, router A is connected to router C using 3 OC-48 connections and to Router D using 1 OC-12 connection, so as to meet the traffic demand from node A to nodes C and D of 7.5 Gbps and 600 Mbps respectively. Suppose that at a later time, traffic from A to C decreases to 5 Gbps, while traffic from A to D increases to 1200 Mbps. Then, the optical layer must be reconfigured so as to delete one OC-48 connection between A and C and creating a new OC-12 connection between A and D. As such, the *requirement of static provisioning at the optical layer is not met.*

In contrast, the routing scheme we propose in this thesis addresses both of the above issues and has the following properties:

- The source routes packets independent of their intended (or, unknown) destination, and
- Both the paths and their bandwidth are fixed a priori and do not need to be reconfigured as traffic patterns change over time.

1.7 Optimization Models

We introduce the optimization metrics that we consider in this thesis. These include network cost and network throughput. A routing scheme specifies the way each matrix in $\mathcal{T}(\vec{R}, \vec{C})$ is routed. The general problem is as follows: Given a class of routing schemes for routing all matrices in $\mathcal{T}(\vec{R}, \vec{C})$, find a scheme that (i) minimizes network cost, or (ii) maximizes network throughput. For the majority of this, we will

be concerned with the class of schemes that we propose in Chapter 2. In order to evaluate the resource overhead of our proposed scheme, we will compare it with an (the) *optimal scheme* among the class of all schemes that are allowed to even make the routing dynamically dependent on the current traffic matrix. In this section, we define the metrics of network cost and network throughput and the optimal scheme in each case. For any routing scheme $\mathcal{A}$, let $A(e,T)$ be the traffic on link e when matrix T is routed by $\mathcal{A}$.

1.7.1 Minimum Cost Network Design

Minimum cost network design is an optimization model that may be suitable when a network is designed from scratch, also called *greenfield design.* In such scenarios, there is no a priori limit on the capacity that can be installed on a link. The maximum traffic usage on link e is $x_e = \max_{T \in \mathcal{T}(\vec{R},\vec{C})} A(e,T)$, hence this amount of capacity needs to be installed on link e. In other words, a set of link capacity allocations x_e is feasible if for any traffic matrix $T \in \mathcal{T}(\vec{R},\vec{C})$, there exists a multicommodity flow for the demands in T that respects link capacities x_e. Assuming a cost of c_e per unit traffic on every link e, the total network cost C_A of scheme $\mathcal{A}$ is given by

$$C_A = \sum_{e \in E} c_e \max_{T \in \mathcal{T}(\vec{R},\vec{C})} A(e,T)$$

The optimal scheme is the one that achieves minimum cost C_{OPT} among all schemes. This is given by

$$C_{OPT} = \min_{\mathcal{A}} C_A$$

1.7.2 Maximum Throughput Network Routing

Throughput, which is the reciprocal of maximum link utilization, is another important optimization metric for network routing. It is one of the most common metrics used in the literature, it is used in capacity planning decisions by ISPs, it is directly related to

other metrics like link congestion, and is useful for multi-period traffic planning when the traffic patterns scale (roughly) uniformly over time. 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).

In this case, the capacity of each link in the network is given. The maximum traffic usage on link e is $\max_{T \in \mathcal{T}(\vec{R},\vec{C})} A(e,T)$, hence this is the portion of the installed capacity u_e on link e that is used. Then, the throughput λ_A of scheme $\mathcal{A}$ is given by

$$\lambda_A = \min_{e \in E} \frac{u_e}{\max_{T \in \mathcal{T}(\vec{R},\vec{C})} A(e,T)}$$

Thus, all matrices in $\lambda_A \cdot \mathcal{T}(\vec{R},\vec{C})$ can be feasibly routed by scheme $\mathcal{A}$. In other words, for any traffic matrix $T \in \mathcal{T}(\vec{R},\vec{C})$, there exists a multicommodity flow for the demands in λT that respects link capacity constraints u_e.

The optimal scheme is the one that achieves the maximum throughput λ_{OPT} among all schemes. This is given by

$$\lambda_{OPT} = \max_{\mathcal{A}} \lambda_A$$

1.7.3 Hardness Results for Traffic Dependent Routing

The problems of minimum cost network design and maximum throughput network routing are computationally intractable when the routing is allowed to be dependent on the traffic matrix. The problem of computing C_{OPT} for minimum cost network design is known to be $co\mathcal{NP}$-hard - the result is stated without proof in Gupta et al. [GKKRY01]. When the graph is undirected and the ingress-egress capacities are symmetric (and only symmetric matrices are considered), the problem is shown to be $co\mathcal{NP}$-hard in Chekuri et al. [COSS05].

Given the graph G and link capacities u_e and ingress-egress traffic capacities

R_i, C_j, the decision version of the problem of determining the throughput λ_{OPT} of the optimal scheme can be stated as: For a given $\lambda > 0$, determine whether $\lambda_{OPT} \geq \lambda$. This is equivalent to determining whether the link capacities $\frac{u_e}{\lambda}$ are feasible for routing all matrices in $\mathcal{T}(\vec{R}, \vec{C})$. If the answer to this problem is "No", then there exists a matrix $T \in \mathcal{T}(\vec{R}, \vec{C})$ which cannot be feasibly routed under the link capacities $\frac{u_e}{\lambda}$. We will show that the associated feasibility problem stated below is $co\mathcal{NP}$-hard.

> Given the graph G and link capacities x_e and ingress-egress traffic capacities R_i, C_j, determine whether all matrices in $\mathcal{T}(\vec{R}, \vec{C})$ can be feasibly routed and if not, identify a matrix $T \in \mathcal{T}(\vec{R}, \vec{C})$ which cannot be routed.

For a given graph G and ingress-egress capacities R_i, C_j, let X be the set of link capacity vectors $\langle x_e \rangle$ that are feasible for routing all matrices in $\mathcal{T}(\vec{R}, \vec{C})$. We first show that X is a polyhedron.

Lemma 1.7.1 *The set X of link capacity vectors $\langle x_e \rangle$ that are feasible for routing all matrices in $\mathcal{T}(\vec{R}, \vec{C})$ is a polyhedron.*

Proof: For a given matrix T, the set of feasible link capacities $\langle x_e \rangle$ and associated multicommodity flow variables for routing the demands in T can be expressed as a linear program and is, hence, a polyhedron. By projecting this polyhedron onto the dimensions representing the link capacities, we obtain a polyhedron of feasible link capacity vectors $\langle x_e \rangle$ for routing the single matrix T.

Let us consider each traffic matrix as an n^2-dimensional vector formed by its entries. Since the traffic matrices in $\mathcal{T}(\vec{R}, \vec{C})$ are specified by bounds on their row and column sums, the corresponding vectors form a bounded polyhedron. A given link capacity vector is feasible for all matrices in $\mathcal{T}(\vec{R}, \vec{C})$ if and only if it is feasible for each traffic matrix corresponding to the vertices of the polyhedron $\mathcal{T}(\vec{R}, \vec{C})$. Since the number of vertices of $\mathcal{T}(\vec{R}, \vec{C})$ is finite, we obtain the result in the lemma by taking the intersection of the polyhedra of feasible link capacity vectors for each vertex of $\mathcal{T}(\vec{R}, \vec{C})$. ■

To prove the $co\mathcal{NP}$-hardness of the feasibility problem, we use the equivalence of polynomial time optimization and polynomial time separation for a convex polyhedron established by Grötschel, Lovász, and Schrijver [GLS88]. The problem of separation consists of determining whether a given point is inside the polyhedron, and if not, identifying a hyperplane separating the point and the polyhedron. Such a procedure is also called a *separation oracle* for the convex polyhedron.

Theorem 1.7.2 *Given a link capacity vector $\langle u_e \rangle$, the problem of determining whether all matrices in $\mathcal{T}(\vec{R}, \vec{C})$ can be feasibly routed, and if not, identifying a matrix $T \in \mathcal{T}(\vec{R}, \vec{C})$ which cannot be routed, is $co\mathcal{NP}$-hard.*

Proof: The problem of finding a minimum cost link capacity vector in X is the minimum cost network design problem and is, hence, $co\mathcal{NP}$-hard. Since X is a convex polyhedron (Lemma 1.7.1), we use the equivalence of polynomial time optimization and polynomial time separation for X to obtain that the separation problem for X is $co\mathcal{NP}$-hard. To prove the theorem, it suffices to show that we can obtain a separating hyperplane for $\langle u_e \rangle$ in polynomial time if we are given a matrix $T \in \mathcal{T}(\vec{R}, \vec{C})$ that cannot be feasibly routed under link capacities u_e.

Assume that we are given a matrix $T = [t_{ij}] \in \mathcal{T}(\vec{R}, \vec{C})$ that cannot be feasibly routed under link capacities u_e. The routing problem for this matrix can be expressed as the following linear program where the variables y_e^{ij} represent the flow on link e for routing demand of value t_{ij} from node i to node j:

---

maximize 0

subject to

$$\sum_{e \in E^+(k)} y_e^{ij} - \sum_{e \in E^-(k)} y_e^{ij} = \begin{cases} t_{ij} & \text{if } k = i \\ -t_{ij} & \text{if } k = j \\ 0 & \text{otherwise} \end{cases} \qquad \forall\, i, j, k \in N \qquad (1.1)$$

$$\sum_{i,j \in N} y_e^{ij} \leq u_e \quad \forall\, e \in E \tag{1.2}$$

$$y_e^{ij} \geq 0 \quad \forall\, e \in E, \quad \forall\, i,j \in N \tag{1.3}$$

The dual of this linear program assigns variables $\pi(k,i,j)$ with each constraint in (1.1) and non-negative variables $w(e)$ with each constraint in (1.2). The dual program can be written as :

$$\text{minimize} \quad \textstyle\sum_{i,j \in N} t_{ij}(\pi(i,i,j) - \pi(j,i,j)) + \sum_{e \in E} u_e w(e)$$

subject to

$$\pi(a,i,j) - \pi(b,i,j) + w(e) \geq 0 \quad \forall\, e = (a,b) \in E, \quad \forall\, i,j \in N \tag{1.4}$$

$$w(e) \geq 0 \quad \forall\, e \in E \tag{1.5}$$

Since the primal linear program is infeasible, the dual linear program has a feasible solution with objective function value less than zero and this solution is computable in polynomial time. Let the values of the dual variables in this solution be $\pi(k,i,j) = \pi_1(k,i,j)$ for all $k,i,j \in N$, and $w(e) = w_1(e) > 0$ for all $e \in E$. Then, we have

$$\sum_{i,j \in N} t_{ij}(\pi_1(i,i,j) - \pi_1(j,i,j)) + \sum_{e \in E} u_e w_1(e) < 0$$

If the link capacities u_e were feasible for routing the matrix T, then the dual objective function value would have been at least zero for any feasible solution. Thus, the following constraint in the variables x_e is violated by the given link capacities $x_e = u_e$:

$$\sum_{e \in E} x_e w_1(e) \geq \sum_{i,j \in N} t_{ij}(\pi_1(j,i,j) - \pi_1(i,i,j))$$

This is the hyperplane separating the link capacity vector $\langle u_e \rangle$ and the polyhedron X. ■

In Section 4.5, we give a characterization of (an) the optimal scheme for maximum throughput network routing that allows us to simulate the optimal scheme in polynomial time, i.e., the routing for any given matrix $T \in \mathcal{T}(\vec{R}, \vec{C})$ under the optimal scheme can be computed in polynomial time. The optimal scheme for minimum cost network design does not appear to have a similar characterization for polynomial time simulation.

1.8 Contributions and Organization of Thesis

This thesis proposes *two-phase routing* as a capacity efficient and robust strategy for handling highly variable traffic. The scheme allows preconfiguration of the network such that all traffic patterns permissible within the network's natural ingress-egress capacity constraints can be routed with bandwidth guarantees *without requiring detection of traffic changes in real-time or reconfiguring the network in response to it.* The scheme routes traffic in two phases. In the first phase, traffic entering the network is sent from the source to a set of intermediate nodes and then, in the second phase, from the intermediate nodes to the final destination. The traffic in the first phase is distributed to the intermediate nodes in predetermined proportions that depend on the intermediate nodes. The traffic split ratios can be generalized to depend on source and destination of traffic also. We now outline the three main parts of the thesis.

Part I

In the first part of the thesis, we develop the two-phase routing scheme and consider the problems of minimum cost network design and maximum throughput network routing. We establish theoretical bounds on the resource requirements of the scheme

under throughput and cost models with respect to the optimal scheme. We evaluate various aspects of two-phase routing on actual ISP topologies. We also generalize the traffic split ratios to depend not only on the intermediate nodes but also on source and destination of traffic.

In **Chapter 2**, we develop the two-phase routing scheme and show how it can handle traffic variability without requiring any detection of traffic changes or reconfiguration of the routing in response to it. We show how the routing scheme meets the additional requirements of the networking applications described in Section 1.5. In particular, the scheme supports static optical layer provisioning in IP-over-Optical networks and indirection in specialized service overlay models unlike previous approaches – like direct source-destination path routing – for handling variable traffic. We address some aspects of two-phase routing related to packet reordering and end-to-end delay. We also discuss how the scheme can be extended to handle network failures. These extensions are considered in detail in the second part of the thesis.

In **Chapter 3**, we consider the problem of minimum cost network design for two-phase routing. We use a link cost model where the cost associated with usage of a link is the capacity allocated on the link multiplied by a cost (per unit capacity) for the link. The total network cost is the sum of cost of usage of all links in the network. We prove that the cost of two-phase routing is at most twice that of the optimal routing scheme when ingress-egress capacities are symmetric, i.e., $R_i = C_i$ for all nodes i. The symmetry of the ingress-egress capacities is not a restrictive assumption in practice because network routers and switches have bidirectional ports (line cards), hence the ingress and egress capacities are equal. We also point out connections of the optimal solution for two-phase routing with that of tree solutions for direct source-destination path routing. This leads to a general bound for arbitrary ingress-egress capacities when the underlying graph is undirected.

In **Chapter 4**, we consider the problem of maximum throughput two-phase routing. We provide link flow based and path flow based linear programming formulations

for determining the intermediate node traffic split ratios and routing of Phase 1 and Phase 2 paths so as to maximize throughput. We show how additional capacity constraints for router-to-OXC links can be incorporated for an IP-over-Optical network architecture. We develop three fast combinatorial algorithms with performance guarantees for the problem using a primal-dual approach on the path indexed linear program – the second and third algorithms improve the running time of the first through two different modifications. We show how the combinatorial algorithms can handle a total cost constraint for maximum throughput two-phase routing. This can be used to solve the capacitated version of minimum cost two-phase routing. The combinatorial algorithms developed are Fully Polynomial Time Approximation Schemes (FPTAS) and find a solution with objective function value within $(1+\epsilon)$-factor of the optimal solution for any given $\epsilon > 0$. The running time is a polynomial function of the input parameters and $\frac{1}{\epsilon}$.

We show that when ingress-egress capacities are symmetric, the throughput of the optimal scheme is at most twice that of two-phase routing. Since computing the throughput of the optimal scheme is a hard optimization problem, we develop heuristics for upper bounding the throughput of the optimal scheme. This will be useful in obtaining a posteriori bounds on the throughput of two-phase routing relative to that of the optimal scheme on actual problem instances. We evaluate various aspects of two-phase routing on actual ISP network topologies collected for the Rocketfuel project [SMWH]. For the evaluated topologies, the throughput of two-phase routing is within 6% of that of the optimal scheme. This is significantly better than our theoretical result which says that two-phase routing is 2-optimal (hence within 50% of optimal scheme).

In **Chapter 5**, we generalize the traffic split ratios to depend on source and destination of traffic. This is conceivably the most general form of two-phase routing. It is motivated by the fact that source nodes should not be required to split traffic through intermediate nodes that are distant from them or the destination nodes of

traffic. This generalization has the potential of reducing network cost or increasing network throughput.

For each of the optimization models of minimum cost network design and maximum throughput network routing, we first provide a linear programming formulation with an infinite number of constraints and a polynomial time separation oracle (another linear program) that is suitable for solution using the ellipsoid method for linear programming. By taking the dual of the separation oracle and combining it with the main linear program, we reduce the number of constraints to polynomial size, thus significantly reducing the running time. For the minimum cost network design problem, we use an upper bound on the demand values to obtain a simplified linear programming formulation that can be interpreted as a fraction Steiner forest problem (this problem admits combinatorial algorithms). Using a technique similar to that used for two-phase routing, we give a polynomial size linear programming formulation for maximum throughput direct source-destination routing of variable traffic – this serves to compare the throughput requirement of two-phase routing with that of direct source-destination path routing on the Rocketfuel ISP topologies.

Part II

In the second part of the thesis, we consider making two-phase routing resilient to network failures.

In **Chapter 6**, we consider how two-phase routing in IP-over-Optical networks can be made resilient against router node failures. In such networks, the first and second phase paths are realized at the optical layer with the routers at intermediate nodes being responsible for (de)multiplexing traffic to its final destination. Routers are known to be much more unreliable than circuit switching based optical switches [LAJ98]. If the router at a node goes down, the node ceases to perform its intermediate node functionality. Thus, the traffic split ratio corresponding to this node has to be redistributed to other intermediate nodes.

We propose two different schemes for provisioning the optical layer to handle router node failures – one that is failure node independent and static (called *failure independent provisioning*), and the other that is failure node dependent and dynamic (called *failure dependent provisioning*). For both mechanisms, we consider sharing bandwidth across single router failure scenarios. For failure independent provisioning, the optical layer is statically provisioned a priori so as to handle any single router failure scenario. For failure independent provisioning, the paths in the optical layer along which traffic is redistributed to other intermediate nodes are provisioned after failure -- this allows sharing of optical layer bandwidth at the link level and may lead to lower restoration capacity overhead compared to the first scheme.

We explain why a simple choice of redistribution ratios proportional to the original traffic split ratios does not lead to optimal throughput. Hence, our problem formulations must accommodate arbitrary redistribution of traffic split ratios after failure. We develop linear programming formulations for both schemes and a fast combinatorial algorithm for the second scheme so as to maximize network throughput. In each case, we determine (i) the optimal distribution of traffic to various intermediate routers for both normal (no-failure) and failure conditions, and (ii) provisioning of optical layer circuits to provide the needed inter-router links.

For failure independent provisioning, we prove that the throughput is at most $\frac{n-1}{n}$ times that for the unprotected case, where n is the number of nodes. For the Rocketfuel topologies, the achieved throughput is within 2% of this theoretical upper bound. Also, the throughput for failure dependent provisioning is less than 1% more than that for failure independent provisioning on the evaluated topologies. Hence, given the static optical layer provisioning property of failure independent provisioning, it might be the preferred scheme for protecting against router node failures in IP-over-Optical networks.

In **Chapter 7**, we consider making two-phase routing resilient to link failures through three different restoration mechanisms -- (i) local (link/span) restoration

(LR), (ii) K-route path restoration (KPR), and (iii) shared backup path restoration (SBPR). In two-phase routing, the first and second phase paths can be protected using any of the above three restoration mechanisms so as to provide resiliency against link failures. The first mechanism reroutes traffic locally around a failure and continues to use the portion of the primary path unaffected by failure. The last two mechanisms are end-to-end (path) based and switch traffic to a diverse backup path after a failure on the primary path. In all of the three restoration models that we consider, backup bandwidth is shared across single link failure events so as to reduce restoration capacity overhead.

We provide linear programming formulations and fast combinatorial algorithms with performance guarantees for maximum throughput two-phase routing with local restoration and K-route path restoration against link failures. We show that the optimization problem for maximum throughput two-phase routing with shared backup path restoration is $\mathcal{NP}$-hard. Assuming an approximation oracle for a certain disjoint paths problem (called SBPR-DISJOINT-PATHS, which is also $\mathcal{NP}$-hard) involving the dual variables of a path indexed linear programming formulation for the problem, we design a combinatorial algorithm with provable guarantees. We also provide heuristics for finding approximating solutions to the SBPR-DISJOINT-PATHS problem. We evaluate some aspects of two-phase routing with the above three restoration mechanisms on the Rocketfuel ISP topologies.

Part III

In the third part of the thesis, we consider the application of two-phase routing to Wireless Mesh Networks (WMNs). This application imposes significant additional constraints on the optimization model so as to merit separate investigation.

Multi-hop wireless mesh networks have recently been of much research interest due to their lowered need for wired infrastructure support and due to envisaged new applications like community wireless networks. In such networks, most of the nodes

are either stationary or minimally mobile and do not rely on batteries. It is difficult to provide (theoretical) bandwidth guarantees for variable traffic in WMNs. The main reason for this is that the dynamic MAC and routing protocols in such networks are limited by local knowledge to do link transmission scheduling and packet forwarding – this leads to inefficient use of physical layer resources. We propose application of the two-phase routing scheme to WMNs so as to avoid this difficulty of distributed routing and scheduling while providing throughput guarantees for variable ingress-egress traffic at each WMN node.

In **Chapter 8**, we extend our optimization framework for maximum throughput two-phase routing in wired networks to handle routing and scheduling constraints that are peculiar to WMNs and arise from the requirement to handle (i) radio transmit/receive diversity, and (ii) the phenomenon of wireless link interference. Link transmission scheduling is equivalent to a graph edge coloring problem which is $\mathcal{NP}$-hard. We use the linear relaxation of the scheduling constraints associated with (i) and (ii) above and incorporate them as link utilization constraints into our earlier linear programming formulations for two-phase routing in wired networks. Our overall approach is to first solve the routing problem after incorporating scheduling constraints into it and then schedule the link transmissions for the obtained link data rates.

For the case of narrow beam forming (directional) antennae in which link interference can be ignored, we design a combinatorial algorithm with performance guarantees. For the case of omnidirectional antennae, we model link interference using link utilization constraints corresponding to cliques and independent sets in the conflict graph. Clique constraints provide an upper bound on the maximum throughput (which may not always be achievable), while independent set constraints provide an achievable lower bound. Our combinatorial algorithm for the narrow beam antenna case can accommodate clique constraints and hence be used to provide upper bounds on throughput for the omnidirectional antenna case. We show how our optimiza-

tion framework can be generalized to handle multiple wireless channels and multiple radios.

We investigate the performance of two-phase routing in WMNs under variability of many of the parameters involved, e.g., transmission power, introduction of relay nodes, and link interference. We also compare the throughput performance of two-phase routing in WMNS with that of the optimal scheme that can change the routing with changes in the traffic.

Summary

In summary, this thesis represents the first comprehensive study, problem formulation, and algorithm design for many aspects of two-phase routing, including:

- Minimum cost network design,
- Maximum throughput network routing,
- Generalization of traffic split ratios to depend on intermediate node and source/destination of traffic,
- Comparison of the resource requirements of two-phase routing with that of (i) the optimal scheme, and (ii) direct source-destination routing along fixed paths,
- Resiliency against router node failures in IP-over-Optical networks through traffic redistribution to other intermediate nodes,
- Protection against link failures through three different restoration mechanisms – local restoration, K-route path restoration, and shared backup path restoration, and
- Application of two-phase routing in wireless mesh networks for handling traffic variation without dynamic routing and scheduling. Additional aspects like link

interference and transmit/receive diversity of communication need to be handled in this case.

Chapter 2

Two-Phase Routing

In this chapter, we develop the routing scheme that is investigated in the thesis and develop some of its properties. We call the scheme *two-phase routing* due to its nature of sending traffic from the source to destination via some intermediate node. We show how the routing scheme meets the requirements of the networking applications described in Section 1.5. We address some aspects of two-phase routing related to packet reordering and end-to-end delay. We discuss how the scheme can be extended to handle network failures. These extensions are considered in detail in Chapters 6 and 7. We also review related work.

2.1 Two-Phase Routing Scheme

The routing scheme that we develop in this section allows the network to accommodate arbitrary (and possibly rapidly changing) traffic demands without sophisticated traffic engineering mechanisms or additional network signaling. In fact, the scheme does not even require the network to detect changes in the traffic distribution. The only assumption about the traffic is the limits on the aggregate traffic originating or terminating at a node, in accordance with the traffic variation model discussed in Section 1.4.

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 2-1. 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.* Depending on the underlying routing architecture, the Phase 1 and Phase 2 paths can be implemented as IP tunnels, optical layer circuits, or Label Switched Paths in Multi-Protocol Label Switching (MPLS) [RVC01].

It is important to observe a subtle aspect of the scheme that may not be apparent from its above description. Notwithstanding the two-phase nature of the scheme, some fraction of the traffic is actually routed to its destination in one phase, i.e., directly from source to destination (this may not be necessarily on a single-hop path). To see this, consider the traffic originating from node i and destined to node j. The fraction α_i of this traffic that should go to (intermediate) node i in Phase 1 does not appear on the network because it originates at node i. Hence, this traffic is routed directly to its destination in Phase 2. Similarly, a fraction α_j of the traffic that goes to (intermediate) node j in Phase 1 actually reaches its final destination after Phase

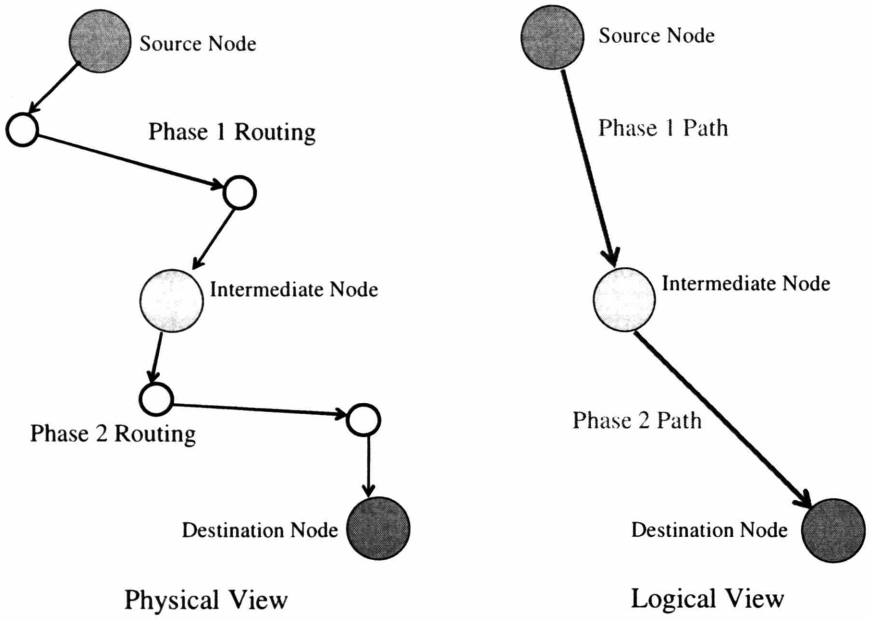


Figure 2-1: Two-Phase Routing Scheme.

1. Hence, this traffic does not appear on the network in Phase 2. Thus, for each source-destination pair (i, j), a fraction $\alpha_i + \alpha_j$ of the traffic between them is directly routed to its destination using only one of the phases (either Phase 1 or Phase 2).

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 $\alpha_i C_j$. The Phase 1 and Phase 2 demands from node i to node j are illustrated in Figure 2-2.

Hence, the maximum demand from node i to node j as a result of routing in Phases

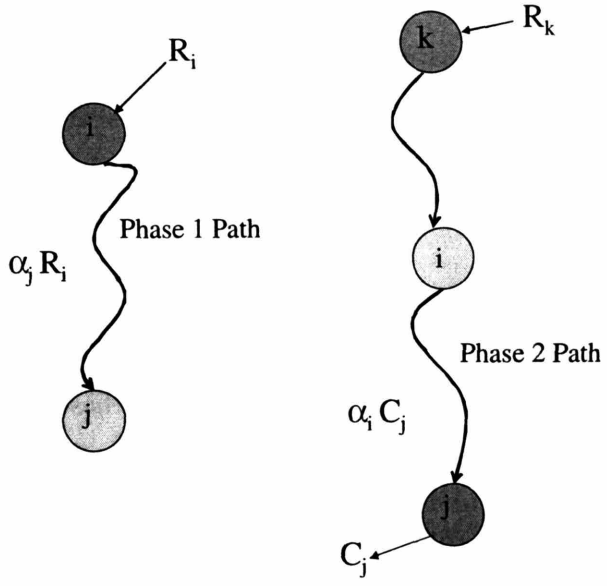


Figure 2-2: Routed demands on Phase 1 and Phase 2 paths.

1 and 2 is $\alpha_j R_i + \alpha_i C_j$. Note that this does not depend on the matrix $T \in \mathcal{T}(\vec{R}, \vec{C})$. Two important properties of the scheme become clear from the above discussion. These are as follows:

Property 1 (Routing Oblivious to Traffic Variations): The routing of source-destination traffic is along fixed paths with predetermined traffic split ratios and *does not depend* on the current traffic matrix $T \in \mathcal{T}(\vec{R}, \vec{C})$.

Property 2 (Provisioned Capacity is Traffic Matrix Independent): The total demand from node i to node j as a result of routing in Phases 1 and 2 is $\alpha_j R_i + \alpha_i C_j$ and does not depend on the traffic matrix $T \in \mathcal{T}(\vec{R}, \vec{C})$ but only on the aggregate ingress-egress capacities. The *bandwidth of the Phase 1 and Phase 2 paths* are *fixed.*

Property 2 implies that the scheme handles variability in traffic matrix $T \in$

$\mathcal{T}(\vec{R}, \vec{C})$ by effectively routing the fixed matrix $D = [d_{ij}] = [\alpha_j R_i + \alpha_i C_j]$ 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 $T \in \mathcal{T}(\vec{R}, \vec{C})$. This is what makes the routing scheme oblivious to changes in the traffic distribution.

We highlight below the main aspects of the novelty of the proposed scheme:

1. Routing decisions at each source node during Phase 1 are local and do not require any network-wide state information (e.g., how the traffic at other ingress-egress points is varying). Routing decisions during Phase 2 are based on the packet destination only as with current IP network routing.

2. The network can meet any traffic distribution as long as the ingress-egress points are not over-subscribed.

3. The routing scheme is oblivious of and robust to any changes in the traffic distribution. Providing end-to-end bandwidth guarantees does not require any reconfiguration of the network in real-time.

From a network operations perspective, we outline below the stages involved in the implementation of two-phase routing:

A. Determine ingress-egress capacities R_i, C_i at each node (using, for example, inter-AS peering agreements, or aggregate rates of network ingress-egress line cards at each node).

B. Determine traffic split ratios $\alpha_1, \alpha_2, \ldots, \alpha_n$ and routing of Phase 1 and Phase 2 paths. (Computing these so as to optimize metrics like network cost and network throughput as well as to cope with network failures is considered in subsequent chapters in the thesis.)

C. For each node pair i, j, provision the fixed bandwidth connections from node i to node j for routing Phase 1 and Phase 2 traffic.

An instance of the scheme requires specification of the traffic split ratios $\alpha_1, \alpha_2, \ldots, \alpha_n$ and routing of the Phase 1 and Phase 2 paths. In Chapter 3, we consider computing these so as to minimize network cost. In Chapter 4, we consider maximum throughput network routing. (The Phase 1 and Phase 2 paths may need to be implemented through multi-path routing in order to maximize throughput.) In Chapter 5, we generalize the traffic split ratios to depend on source and/or destination nodes of the traffic and consider the problems of minimum cost network design and maximum throughput network routing for this general case.

2.2 Potential Applications of Two-Phase Routing

We now return to the networking architectures and applications described in Section 1.5 and discuss how two-phase routing can meet the requirements we identified for them.

2.2.1 IP-over-Optical Networks

Two-phase routing, as envisaged for IP-over-Optical networks, establishes the fixed bandwidth Phase 1 and Phase 2 paths at the optical layer. Thus, the *optical layer is statically provisioned* and does not need to be reconfigured in response to traffic changes. IP packets are routed end-to-end with *IP layer processing at a single intermediate node only.* While in transit at the optical layer inside either Phase 1 or Phase 2 paths, packets do enter the router but appear as transit traffic at the Optical Cross-Connect (OXC) only. The IP layer packet processing at an intermediate node works as follows. The optical layer circuit is dropped at the IP router at the node (through OXC-to-router links), wherein the packets are multiplexed back to the OXC (through router-to-OXC links) to be routed through direct optical layer circuits to their final destinations. Figure 2-3 illustrates optical layer transit traffic and intermediate node packet processing functionality at a node for two-phase routing.

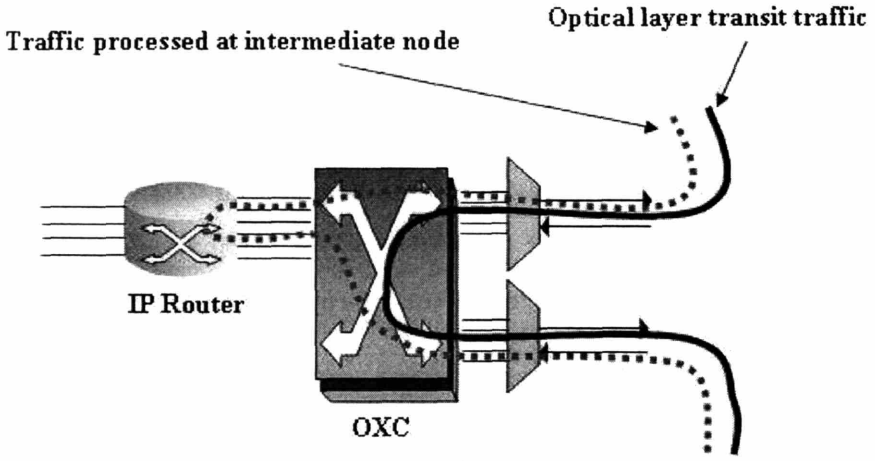


Figure 2-3: Intermediate node packet processing for Two-Phase Routing in IP-over-Optical networks.

This architecture provides the desirable statistical multiplexing properties of packet switching for handling highly variable traffic *without significantly increasing IP layer transit traffic*. Compare this with the high levels of IP layer transit traffic in IP-over-WDM architecture where routers are directly connected to WDM systems and need to process packets at each hop.

In summary, two-phase routing when applied to IP-over-Optical networks leads to an architecture with the following salient features:

- IP traffic is routed "mostly" at the optical layer from source to destination routers with *packet grooming at one intermediate router* only.
- The optical layer (circuits and their bandwidth) are statically provisioned a priori to provide bandwidth guarantees for end-to-end IP traffic. Routing at the IP layer is static – there is no need to detect changes in traffic or reconfigure the routing in response to it.
- Bandwidth guarantees are provided for routing all traffic matrices within the network's natural ingress-egress capacity constraints.

An IP layer (logical) link between node i and node j is realized by a circuit in the optical layer. If the IP layer logical topology is a full mesh complete graph, then the optical layer needs to be provisioned for $\Theta(n^2)$ circuits. Such an architecture does not scale well with increasing network size from an ISP deployment and network operations and management perspective. Hence, it is generally desirable that the IP layer topology have sparse connectivity, so that the number of circuits in the optical layer scales linearly (or, at most sub-quadratically) with network size. In two-phase routing, if the number of intermediate nodes i with traffic split ratios $\alpha_i > 0$ is k, then the number of IP layer links is

$$k(n-k) + \frac{k(k-1)}{2} = kn - \frac{1}{2}k^2 - \frac{1}{2}k$$

The first term $k(n-k)$ corresponds to the number of IP links for splitting traffic originating from the $n-k$ non-intermediate nodes to the k intermediate nodes. The second term $\frac{k(k-1)}{2}$ corresponds to the number of IP links for splitting traffic originating from each intermediate node to the other $k-1$ intermediate nodes. Thus, if the number of intermediate nodes k is small compared to the number of network nodes n, then the number of IP layer links scales linearly with n. Experiments on actual ISP topologies collected for the Rocketfuel project [SMWH] for maximum throughput two-phase routing in Chapter 4 indicate that the number of intermediate nodes in two-phase routing is indeed small compared to the total number of nodes in the network.

2.2.2 Specialized Service Overlay Networks

Two-phase routing can be used to provide Quality-of-Service (QoS) guarantees for variable traffic and support indirection in intra-ISP deployments of specialized service overlays like i3. (Note that we are not considering Internet-wide deployment here.) The intermediate nodes in the two-phase routing scheme are ideal candidates for lo-

cating i3 servers. Because we are considering a network whose topology is known, the two-phase routing scheme can be used to not only pick the i3 server locations (intermediate nodes) but also traffic engineer paths for routing with bandwidth guarantees between sender and receiver through i3 server nodes. Because the two-phase routing scheme can route the Phase 1 and Phase 2 paths with protection (as discussed in Chapters 6 and 7), it can also provide network level reliability of the services provided.

In service overlay models like i3, the *final destination of a packet is not known at the source* but only at the i3 infrastructure nodes. Because the final destination of a packet needs to be known only at the intermediate nodes in two-phase routing, it is well-suited for providing indirection in service overlay models. In contrast, for direct source-destination path routing, the source needs to *know the destination of a packet* for routing it, thus rendering it unsuitable for such service overlay networks.

The ingress-egress traffic constraints R_i, C_j in the two-phase routing scheme now apply to network nodes to which hosts attach for using the services provided. For example, the host could be a laptop and a node could be an enterprise site or an ISP PoP. Mobility of the hosts manifests itself as changes in traffic originating from or destined to the network points of attachment (nodes), since mobile hosts will attach themselves to different nodes over time. The Phase 1 and Phase 2 paths of the specified bandwidth will provide bandwidth guarantees across all i3 applications described in [SAZSS02], including mobility, multicast, and anycast. This is because the traffic arising from such applications must obey, by definition, the aggregate ingress-egress constraints at each node.

In adapting the two-phase routing scheme for providing i3-like functionality, there remains the issue of routing a packet within the i3 infrastructure that we address next. Recall that this is handled in the original i3 scheme through a peer-to-peer routing protocol like Chord [SAZSS02]. We obviate the need for routing within the i3 infrastructure (in our case, the intermediate nodes) in an intra-ISP deployment in either of two ways.

In the first method, a trigger is replicated at all intermediate nodes. This is easy to implement since the network node to which the receiver host attaches participates in the two-phase routing scheme and is, hence, aware of all intermediate nodes. Trigger replication at intermediate nodes also provides the added functionality of robustness against i3 server failures at intermediate nodes, and, in this case, guarantees fast recovery of sender-receiver connectivity through traffic redistribution to other i3 server (intermediate) nodes (this mechanism is considered in Chapter 6). This aspect is handled in the original scheme through either (i) receiver-initiated soft-state refresh (reinsertion) of triggers, (ii) backup triggers, or (iii) trigger replication across i3 servers. Because the failure of intermediate nodes is handled by two-phase routing (and, hence at the network layer), the restoration latency is expected to be smaller. Also, to the extent that the number of intermediate nodes is configurable in two-phase routing, trigger replication across (a relatively small number of) intermediate nodes may not pose a scalability problem.

In the second method, there is no trigger replication – triggers with the same id are assigned to a single intermediate node. In order to preserve the bandwidth guarantees of the two-phase routing scheme, it must be ensured that assignment of triggers to intermediate nodes splits the resulting traffic for each sender to the intermediate nodes in (approximately) the predetermined traffic split ratios.

2.2.3 Routing via Middleboxes

Two-phase routing can naturally accommodate a middlebox routing architecture in ISP networks and also provide QoS guarantees for variable traffic. The intermediate nodes in two-phase routing are ideal locations for deploying middleboxes that provide functionalities like caching and content filtering. Because all traffic passes through one of the intermediate nodes in the scheme, the requirement of the middleware service to be comprehensive (in the sense that every packet routed through the network must be examined at least once) is also met. The routing can now provide end-to-

end bandwidth guarantees for variable traffic patterns. Experiments on actual ISP topologies collected for the Rocketfuel project [SMWH] for maximum throughput two-phase routing in Chapter 4 indicate that the number of intermediate nodes in two-phase routing is small compared to the total number of nodes in the network. Given that the deployment of services like content filtering are expensive (from a hardware perspective), a smaller number of intermediate nodes can lead to cost-effective deployment of such services.

2.3 Addressing Some Aspects of Two-Phase Routing

We address some practical aspects of two-phase routing related to packet reordering and end-to-end delay and explain why they should not pose any hurdles in the deployment of the routing scheme in ISP networks.

2.3.1 Packet Reordering

In two-phase routing, as described in Section 2.1, the source node splits traffic to different intermediate nodes regardless of the final destination. Thus, packets belonging to the same end-to-end connection can arrive out of order at the destination node if the traffic is split within the same connection. The question arises whether this packet reordering is an issue that needs to be addressed.

The Internet standard for IP router requirements, RFC 1812, does not prohibit packet reordering in routers [B95]. In fact, parallelism in router/OXC components and links causes packet reordering under normal operation and has been observed in the Internet [BPS99, JIDKT03]. Packet reordering can affect the performance of TCP (Transmission Control Protocol) [D00] and other traffic that relies on packet ordering. In its current version, TCP, which is used to carry most of the traffic in

today's Internet, interprets packet reordering as a loss indicator. This triggers unnecessary retransmissions and TCP timeouts that cause a decrease in TCP throughput and increase in packet delay. Proposals have been made to make TCP more robust to packet reordering [BA02]. However, any new TCP feature requires protocol standardization and modification/upgrade of TCP software implementations running on hundreds of millions of client devices. Hence, it might be desirable to avoid packet reordering.

Packet reordering can be avoided in two-phase routing by splitting traffic at the application flow level at the source node (rather than at the packet level). An application level flow corresponds to a single end-to-end session of communication between two users on different machines and is commonly identified by a 5-tuple consisting of source IP address, destination IP address, source port number, destination port number, and protocol id [D00]. In order to prevent congestion along the Phase 2 paths in two-phase routing, it is necessary to split the set of flows corresponding to each destination node among intermediate nodes in accordance with the traffic split ratios $\alpha_1, \alpha_2, \ldots, \alpha_n$.

The question then is whether two-phase routing with per-flow splitting can provide bandwidth guarantees to all traffic matrices within the network's natural ingress-egress capacity constraints. The answer, as we explain below, is in the affirmative. We are advocating this routing scheme for core networks where recent advances in DWDM (Dense Wavelength Division Multiplexing) transmission technologies [RS02] have resulted in link bandwidths of 10 Gbps and heading higher. Individual flows at best are in the Mbps range and hence small compared to link rates -- tens of thousands of flows share a single 10 Gbps link. Moreover, TCP is not really good for gigabit flows -- the throughput goes down as $1/(RTT * \sqrt{\text{random loss probability}})$ [D00] and so very low random loss (due to bursty cross traffic) is needed to get gigabit throughputs for any reasonable RTT (round-trip time).

2.3.2 Increase in End-to-End Delay

Because of the two-phase nature of the routing scheme, packets might incur about twice the delay in end-to-end routing compared to direct source-destination path routing along shortest paths. However, for the portion of traffic that is routed directly to its destination (as explained in Section 2.1, or when the intermediate node already lies on the shortest source-destination path, no additional delay is encountered, thus suggesting that the average delay may be less than a factor of two compared to that in direct source-destination path routing. In fact, experiments on actual ISP topologies collected for the Rocketfuel project [SMWH] for maximum throughput two-phase routing indicate that the end-to-end hop count as a result of two-phase routing is about 1.4-1.6 times that of shortest path routing.

More importantly, this increase in delay may be tolerable for most applications. Consider an Internet backbone spanning the transcontinental US. A `ping` from MIT (US east coast) to UC Berkeley (US west coast) gives a round-trip time of about 90 msec. This round trip time includes two traversals of the long-haul network *and* two traversals each of Boston and Oakland metro access networks. Thus, the one-way end-to-end traversal time with two-phase routing deployed in the long-haul network can be expected to be much less than 90 msec. An end-to-end delay of up to 100 msec is acceptable for most applications.

Moreover, the bandwidth guarantees provided by two-phase routing under highly variable traffic reduce the delay variance (jitter). This reduction in jitter and the guarantee of predictable performance to unpredictable traffic provides a reasonable trade-off for the fixed increase in propagation delay. The effect of bursty traffic on jitter is mitigated by the source splitting of traffic to multiple intermediate nodes in two-phase routing.

Delay-sensitive traffic that cannot tolerate traversal of the core backbone twice can be routed along shortest paths in a hybrid architecture that accommodates both two-phase routing and direct source-destination path routing.

2.4 Extensions for Handling Network Failures

We provide an overview of extensions to two-phase routing that can make the scheme resilient to network failures. Detailed descriptions and the associated optimization problems are considered in Chapters 6 and 7.

2.4.1 Router Node Failures in IP-over-Optical Networks

Studies like [LAJ98] indicate that IP routers are 200 times more unreliable than traditional carrier-grade switches and average 1219 minutes of down time per year. Given this unreliability of routers, it is worthwhile to consider how two-phase routing in IP-over-OTN can be made resilient against router node failures. In the term "router node failure", node refers to a PoP, hence it includes the failure of all routers in a PoP. When a router at a node f fails, any other node i cannot split any portion of its originating traffic to intermediate node f. Hence, it must redistribute the traffic split ratio α_f among other nodes $j \neq f$. In Chapter 6, we propose two different schemes for provisioning the optical layer to handle this redistribution of traffic – one that is failure node independent and static, and the other that is failure node dependent and dynamic.

Note that since only the router (and not the OXC) at node f fails, this node can continue to be on the Phase 1 and Phase 2 paths for optical layer switching. However, the total traffic that was supposed to originate at that node, i.e., R_f, no longer enters the network.

2.4.2 Link Failures

Link failures can be caused by events like fiber cuts and malfunctioning of router/switch line cards. In Chapter 7, we extend two-phase routing by providing resiliency against link failures through three different *pre-provisioned* restoration mechanisms. By pre-provisioned, we mean that the backup paths are computed and "soft-reserved" a priori

– the main action after failure is the rerouting of affected traffic on backup paths. The pre-provisioned nature of these mechanisms increases the reliability of the network by guaranteeing availability of backup resources after failure. It also allows fast restoration of traffic in an attempt to provide failure transparency to upper network layers. In two-phase routing, each of the Phase 1 and Phase 2 paths can be protected against link failures by any of the three restoration mechanisms.

The first restoration mechanism we consider is local (or, link based) and consists of rerouting traffic around the failed link through pre-provisioned backup paths (link detours). The routing of traffic on portions of the primary path unaffected by the failure remains unchanged. Backup paths protecting different links can share bandwidth on their links so as to guarantee complete recovery against any single link failure.

The other two mechanisms are end-to-end (or, path based) and different in the way backup bandwidth is shared across different link failure scenarios. In *K-route path restoration*, a connection consists of $K \geq 2$ link disjoint paths from source to destination. One of these paths is designated as the backup path and the others as primary paths. The backup path carries traffic when any one of the primary paths fail due to a link failure.

In *shared backup path restoration*, a primary path is protected by a link disjoint backup path. Different 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 link failures. Shared backup path restoration has been shown to have lower restoration capacity overhead compared to the other two mechanisms described above [G03].

2.4.3 Complete Node Failures

We discussed how to make two-phase routing resilient to router node failures in IP-over-Optical networks. Failure of OXC switches at a node in an IP-over-Optical network or of routers at a node in a pure IP router architecture (IP routers directly

connected to WDM systems) leads to failure of the node for routing Phase 1 and Phase 2 paths also (in addition to loss of intermediate node functionality). The handling of such complete 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 extending the mechanisms for protecting against link failures – we use detours around nodes in local restoration or node-disjoint paths in path restoration. The failure of intermediate nodes can be handled through redistribution of traffic to other intermediate nodes. Because a complete node failure can lead to both of the above scenarios, a combination of the corresponding mechanisms can be used to protect against such failures.

2.5 Related Work

In Section 1.6, we reviewed related work [DGGMR99, KRSY01, ACFKR03] for routing variable traffic that uses direct paths from source to destination. We pointed out two aspects of these approaches that do not meet the requirements of application scenarios discussed in Section 1.5, namely (i) the source needs to *know the final destination of a packet* for routing it, and (ii) the bandwidth requirements of the (fixed) paths change with traffic variations.

Because of (i), these methods cannot be used to provide indirection in specialized 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.

Valiant [V82] introduced a routing scheme in the context of communication between parallel processors interconnected in an $N = 2^n$ node hypercube topology. The nodes of this hypercube can be represented as n-dimensional binary vectors. A

packet can move in the k-th dimension from node i to node j if vectors i and j differ in dimension k only. Valiant's scheme operates in two steps. In a first step, a node forwards its packet to a random intermediate node irrespective of the packet's final destination. This is done by considering the n dimensions in sequential order and deciding at random whether to move in that dimension or not. In a second step, each node determines the final destination nodes of the packets it has received in the first step, and routes it directly to the destination nodes. This is done by considering the n dimensions in sequential order and moving in a dimension if it is closer to the destination. Valiant showed that when each node initially contains one packet addressed to distinct nodes of the cube (permutation demand matrix), then under the restriction that no two packets can pass down the same wire at any one time, this strategy can route every packet to its destination and finishes in $O(\log N)$ time with overwhelming probability.

Chang et al. [CLJ02] proposed a Valiant-type approach as an effective scheme for avoiding scheduling bottlenecks in high-speed input-buffered network switches. In this work, packets are initially routed to a randomly chosen intermediate port in a switch and then subsequently to the true output port.

The origins of the routing scheme proposed in this thesis can be traced back to the two-step nature of Valiant's scheme, where routing is through a *randomly and uniformly chosen intermediate node.* We propose a deterministic scheme with possibly *unequal traffic split ratios*, show how it can accommodate all traffic matrices within the network's natural ingress-egress capacity constraints, and consider many new aspects arising from its potential application to routing Internet traffic in ISP backbone networks and wireless mesh networks.

We describe two-phase routing, as presented in this chapter, in Kodialam, Lakshman, and Sengupta [KLS04a, KLS04b]. Zhang and McKeown [ZM04] consider a restricted 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). They 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 [KLS04a, KLS04b] (and in this thesis) 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.

Chapter 3

Minimum Cost Network Design

In this chapter, we consider the problem of minimum cost network design for two-phase routing. We use a link cost model where the cost associated with usage of a link is the capacity allocated on the link multiplied by a unit capacity cost for the link. The total network cost is the sum of cost of usage of all links in the network. We show that the optimal solution sends all traffic through one intermediate node along shortest cost paths.

We prove that the cost of two-phase routing is at most twice that of the optimal routing scheme *when ingress-egress capacities are symmetric*, i.e., $R_i = C_i$ for all nodes i. The proof actually establishes the bound for an instance of two-phase routing where the traffic split ratios to each intermediate node is proportional to the ingress (egress) capacity of that node. We also point out connections of the optimal solution for two-phase routing with that of tree solutions for direct source-destination path routing. This leads to a general bound for arbitrary ingress-egress capacities *when the underlying graph is undirected.*

3.1 Optimal Solution for Minimum Cost Two-Phase Routing

In this section, we consider the problem of minimum cost network design for two-phase routing. We need to choose the traffic split ratios α_i and route $\alpha_j R_i + \alpha_i C_j$ amount of demand from node i to node j, for all $i, j \in N$, such that the resulting capacity allocation x_e on each link e minimizes the total network cost $\sum_{e \in E} c_e x_e$. We begin by a simple characterization the structure of any feasible solution for the problem.

Lemma 3.1.1 *Any feasible solution for two-phase routing can be expressed as a convex combination of solutions that route through a single intermediate node. The weights in the convex combination are the traffic split ratios α_i for associated intermediate node i.*

Proof: Consider an intermediate node k with traffic split ratio α_k in a feasible solution of two-phase routing. Since the demand routed from any node $i \neq k$ to node k in the solution is $\alpha_k R_i + \alpha_i C_k$, it can be decomposed into two sets of flows from node i to node k of values $\alpha_k R_i$ and $\alpha_i C_k$ respectively. From the first flow, we obtain the routing of Phase 1 paths from node i to intermediate node k. (The second flow gives us the routing of Phase 2 paths from intermediate node i to node k.) Similarly, by considering the demand routed from intermediate node k to any node $j \neq k$, we obtain the routing of the Phase 2 paths from intermediate node k to node j. If we divide the demands along the obtained Phase 1 and Phase 2 paths associated with intermediate node k by α_k, we get a solution for two-phase routing in which node k is the only intermediate node.

The given solution for two-phase routing can thus be decomposed into demands along Phase 1 and Phase 2 paths associated with each intermediate node k. Since the traffic split ratios α_k sum to 1, this gives us a convex combination of solutions that route through a single intermediate node. ■

Because we are considering the network design problem where no a priori link capacities are given, the Phase 1 and Phase 2 paths will be routed along shortest cost paths in an optimal solution. Let d_{ij} denote the cost of the shortest path from node i to node j under link costs c_e. Then, the minimum cost $C(k)$ of a solution that sends all traffic through a single intermediate node k is given by

$$C(k) = \sum_{i \in N, i \neq k} d_{ik} R_i + \sum_{j \in N, j \neq k} d_{kj} C_j$$

Consider an optimal solution for two-phase routing in which the traffic split ratios are $\alpha_1, \alpha_2, \ldots, \alpha_n$. Using the above decomposition lemma, the cost of this optimal solution can be written as

$$\sum_{k \in N} \alpha_k C(k)$$

Let $k = \bar{k}$ be the node for which $C(k)$ is minimum. Since the traffic split ratios α_k sum to 1, the above cost is a minimum when $\alpha_{\bar{k}} = 1$ and $\alpha_k = 0$ for all $k \neq \bar{k}$. Thus, all traffic is sent through a single intermediate node $\bar{k}$. The structure of this solution is illustrated in Figure 3-1. The result is summarized in the following theorem.

Theorem 3.1.2 *Let $k = \bar{k}$ be the node for which*

$$C(k) = \sum_{i \in N, i \neq k} d_{ik} R_i + \sum_{j \in N, j \neq k} d_{kj} C_j$$

is minimum. Then, the minimum cost solution for two-phase routing sends all traffic through the single intermediate node $\bar{k}$ along Phase 1 and Phase 2 paths that are shortest paths with respect to the given link costs c_e.

3.2 How Optimal is Two-Phase Routing?

In Section 1.7.1, we defined the optimal scheme for minimum cost network design from a general class of schemes that can *make the routing dependent on the (current)*

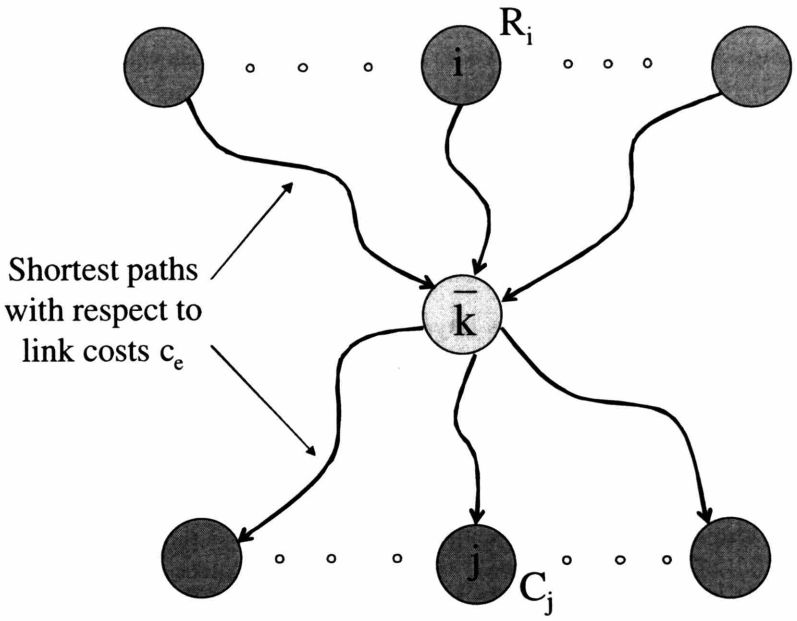


Figure 3-1: Structure of a Minimum Cost Solution for Two-Phase Routing.

traffic matrix. The optimal scheme minimizes $\sum_{e \in E} c_e x_e$ where x_e is a set of valid link capacities for routing all matrices in $\mathcal{T}(\vec{R}, \vec{C})$. A set of link capacities x_e is valid if for every traffic matrix $T \in \mathcal{T}(\vec{R}, \vec{C})$, there exists a multicommodity flow for the demands in T that respects link capacities x_e. In this section, we show that the cost of two-phase routing is at most twice that of the optimal scheme.

We assume that $R_i = C_i$ for all nodes i. Note that this is not a restrictive assumption in practice because network routers and switches have bidirectional ports (line cards), hence the ingress and egress capacities are equal.

Theorem 3.2.1 *Let $R_i = C_i$ for all nodes i, and $R = \sum_{i \in N} R_i$. For minimum cost network design, the cost of two-phase routing is at most*

$$2\left(1 - \frac{1}{R} \min_{i \in N} R_i\right)$$

times that of the optimal scheme.

Proof: Let α_i be the traffic split ratios associated with each intermediate node i in two-phase routing. Set $\alpha_i = \frac{R_i}{R}$ for all $i \in N$. Clearly, the cost of this solution is at least that of the minimum cost solution. The demand matrix $D = [d_{ij}]$ as a result of two-phase routing for the above traffic split ratios is given by

$$\begin{aligned} d_{ij} &= \alpha_j R_i + \alpha_i C_j \\ &= \alpha_j R_i + \alpha_i R_j \\ &= 2\frac{R_i R_j}{R} \end{aligned}$$

for $i \neq j$ and $d_{ii} = 0$ for all i.

Now consider the traffic matrix $T = [t_{ij}]$ where

$$t_{ij} = \frac{R_i R_j}{R}$$

for all $i \neq j$ and $t_{ii} = 0$ for all i. Let β be the maximum multiplier such that $\beta T \in \mathcal{T}(\vec{R}, \vec{C})$. Then, we must have

$$\begin{aligned} \beta \sum_{j \in N, j \neq i} t_{ij} &\leq R_i \ \ \forall\, i \in N \\ \beta \sum_{j \in N, j \neq i} \frac{R_i R_j}{R} &\leq R_i \ \ \forall\, i \in N \\ \beta \frac{R_i(R - R_i)}{R} &\leq R_i \ \ \forall\, i \in N \\ \beta &\leq \frac{R}{R - R_i} \ \ \forall\, i \in N \end{aligned}$$

whence,

$$\beta = \frac{R}{R - \min_{i \in N} R_i}$$

Since $D = 2T$, hence $\beta T = \frac{\beta}{2} D$. Since the optimal scheme must route the matrix $\beta T \in \mathcal{T}(\vec{R}, \vec{C})$, its cost is at least the minimum cost for routing matrix $\beta T = \frac{\beta}{2} D$. Since D is the demand matrix for two-phase routing, we conclude that the cost of

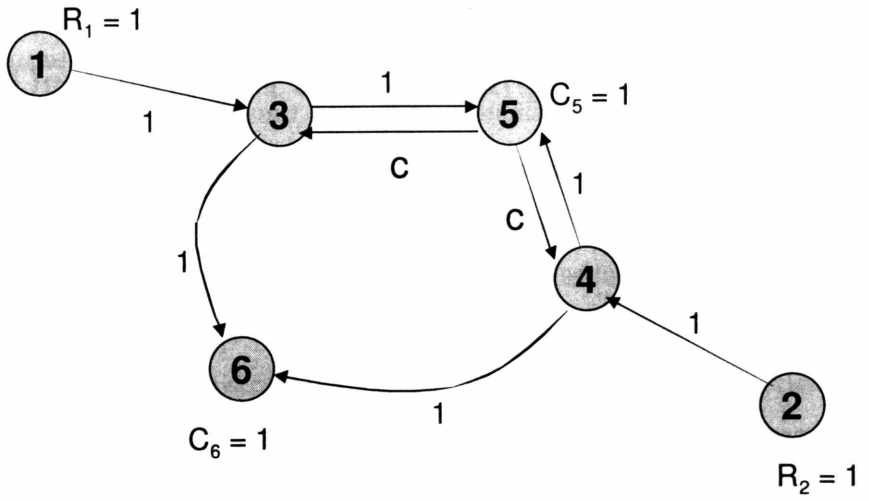


Figure 3-2: 6-node network to illustrate gap between cost of Two-Phase Routing with α_i split ratios and that of optimal scheme when ingress-egress capacities are not symmetric.

two-phase routing is at most

$$\frac{2}{\beta} = 2\left(1 - \frac{1}{R}\min_{i \in N} R_i\right)$$

times that of the optimal scheme. ■

If ingress-egress capacities are not symmetric, then the gap between cost of two-phase routing with intermediate node dependent split ratios α_i and that of optimal scheme can be made arbitrarily large, as shown by the following example. Consider the 6-node network shown in Figure 3-2. Here $R_1 = R_2 = 1$ and $C_5 = C_6 = 1$. All other R_i, C_j values are zero. The costs of links $(5,3)$ and $(5,4)$ are each equal to some large quantity c. All other links shown have unit cost.

Observe that node 1 has a unit cost path to node 3 but the cost of the path to node 4 is large $(= c+2)$. Similarly, node 2 has a unit cost path to node 4 but the cost of the path to node 3 is large $(= c+2)$. Thus, when minimizing cost, node 4 is not a good choice for serving as intermediate node for the traffic originating at node 1.

Similarly, node 3 is not a good choice for serving as intermediate node for the traffic originating at node 2. Since our split ratios can be made dependent on intermediate nodes *only* (and not on source or destination of traffic), the cost of two-phase routing will be large. The optimal scheme, on the other hand, can completely avoid routing along the links with large costs. In fact, the gap between the cost of two-phase routing and that of optimal scheme can be made arbitrarily large by making the value of L arbitrarily large.

The arbitrarily large gap in the above example between the cost of two-phase routing with intermediate node dependent traffic split ratios and the cost of optimal scheme can be mitigated by generalizing the split ratios to depend on source and/or destination of traffic. We consider this in Chapter 5. The rationale behind this generalization is that it might be expensive to route traffic through a given intermediate node for certain source-destination pairs while relatively cheap for others.

The optimal solution for minimum cost two-phase routing in Section 3.1 sends all traffic through a single intermediate node. This is not desirable in practice because of at least two reasons. First, the single intermediate node in an optimal solution becomes a single point of failure in the network. Secondly, it requires huge routing capacity to handle all of the network traffic – it may not be feasible in practice to scale router capacity at a single node to required levels. The instance of two-phase routing in the proof of Theorem 3.2.1 splits traffic to intermediate nodes in ratios proportional to their respective ingress (egress) capacities – this solution does not suffer from the above drawback of the optimal solution for two-phase routing and its cost is still within a factor of two of that of the optimal scheme.

3.3 Connections with Tree Solutions for Direct Source-Destination Path Routing

We briefly discussed direct source-destination path routing for the hose traffic model in Section 1.6. Tree solutions for the minimum cost network design version of the problem on an *undirected graph* were first considered in [KRSY01]. By a tree solution, we mean that the links on which bandwidth is reserved, i.e., all links e for which $x_e > 0$, form a tree. The traffic from node i to node j is routed along the unique path from i to j in the tree. The link capacities x_e must be consistent for the routing of all matrices $T \in \mathcal{T}(\vec{R}, \vec{C})$.

Consider the following simple algorithm for a tree solution for direct source-destination path routing in an undirected graph. Fix some node $k \in N$ and form the shortest path tree rooted at node k and reaching all other nodes. For each node $i \neq k$, reserve $R_i + C_i$ units of capacity on the shortest path between node i and node k. The effect of reserving capacities on links for different shortest paths is cumulative. The cost of the solution is $\sum_{e \in E} c_e x_e$, where x_e is the total capacity allocation on link e as a result of the above procedure. Repeat this procedure for each initial fixed node $k \in N$ and choose the solution that has the lowest cost.

It is shown in [GKKRY01] that the above procedure produces an optimum tree solution in undirected graphs when the ingress-egress capacities are symmetric, i.e., $R_i = C_i$ for all i. The solution is an optimum tree solution even when the ingress-egress capacities are balanced, i.e., $\sum_{i \in N} R_i = \sum_{j \in N} C_j$, as shown in [ILO02].

Fractional solutions for direct source-destination routing along fixed paths are allowed to split the traffic between a source-destination pair into multiple paths. The paths and the ratios in which traffic is split among them must be fixed a priori and cannot depend on the traffic matrix to be routed. Let $R = \sum_{i \in N} R_i$ and $C = \sum_{j \in N} C_j$. It is shown in [EGOS05] that for undirected graphs and general ingress-egress capacities (not necessarily symmetric or balanced), the cost of the tree solution obtained by

the above procedure is within a factor of $(1 + \max(\frac{R}{C}, \frac{C}{R}))$ of the optimum fractional solution for direct source-destination routing along fixed paths. The arguments in [EGOS05] can be used to establish the same bound for the above tree solution *with respect to the optimal scheme that is allowed to change the routing with changes in the traffic matrix.*

In our problem formulation for minimum cost two-phase routing, we assumed a directed graph. However, the arguments for optimality of the two-phase routing solution obtained in Section 3.1 continue to hold when the underlying graph is undirected (since the allocated capacity on an undirected link can be used to send traffic in either direction). When the graph is undirected, it should be easy to see that the minimum cost solution for two-phase routing in Theorem 3.1.2 is the shortest path tree rooted at node $\bar{k}$ and it is identical in structure, link capacity allocation, and total cost to that obtained by the above procedure for the optimum tree solution for direct source-destination path routing. (Note that even though the solutions are identical, the path along which traffic is routed for a given source-destination pair could be different for two-phase routing and direct source-destination path routing.)

From the above, we conclude that for undirected graphs, the cost of two-phase routing is within a factor of $(1 + \max(\frac{R}{C}, \frac{C}{R}))$ of that of the optimal scheme. As a corollary, we obtain a 2-optimality bound for two-phase routing on undirected graphs when ingress-egress capacities are balanced. Moreover, the optimality bound degrades linearly with the gap between the total ingress and egress capacities.

Chapter 4

Maximum Throughput Network Routing

In this chapter, we consider the problem of maximum throughput two-phase routing. The throughput is the maximum multiplier λ such that all matrices in $\lambda \cdot \mathcal{T}(\vec{R}, \vec{C})$ can be feasibly routed. The reciprocal of throughput is the maximum link utilization in the network, so this problem is equivalent to computing an instance of two-phase routing for all matrices in $\mathcal{T}(\vec{R}, \vec{C})$ so as to minimize the maximum link utilization. Given a network with link capacities u_e and constraints R_i, C_i on the ingress-egress traffic at each node, the problem consists of computing the traffic split ratios α_i and routing of Phase 1 and Phase 2 paths so as to maximize the throughput.

Throughput is among the most common and important optimization metric for network routing. It is used in capacity planning decisions by ISPs, is directly related to other metrics like link congestion, and is useful for multi-period traffic planning when the traffic patterns scale (roughly) uniformly over time. 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).

We begin with a link flow based linear programming (LP) formulation for the

problem. We show how additional capacity constraints for router-to-OXC links can be incorporated for an IP-over-Optical network architecture. We bring out the need for and develop three combinatorial algorithms for the problem – the second and third algorithms improve the running time of the first through two different modifications. We also show how the combinatorial algorithms can handle a total cost constraint for maximum throughput two-phase routing. This can be used to solve the problem of minimum cost two-phase routing under given link capacities.

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 (link capacities and ingress-egress traffic capacities). The value of ϵ can be chosen to provide the desired degree of optimality for the solution. An FPTAS is said to be *strongly polynomial* when its running time does not depend on the size of the input numbers. The third combinatorial algorithm developed for the problem is strongly polynomial.

We compare the throughput of two-phase routing with that of the optimal scheme that is allowed to make the routing dependent on the traffic matrix. We show that when ingress-egress capacities are symmetric, the throughput of the optimal scheme is at most twice that of two-phase routing. The symmetry of the ingress-egress capacities is not a restrictive assumption in practice because network routers and switches have bidirectional ports (line cards), hence the ingress and egress capacities are equal. Computing the throughput of the optimal scheme was shown to be a hard optimization problem in Section 1.7.3. We develop heuristics for upper bounding the throughput of the optimal scheme. This will be useful in obtaining a posteriori bounds on the throughput of two-phase routing relative to that of the optimal scheme on actual problem instances.

Finally, we evaluate various aspects of two-phase routing on actual ISP network topologies collected for the Rocketfuel project [SMWH, SMW02, MSWA02]. For the evaluated topologies, the throughput of two-phase routing is within 6% of that of the optimal scheme. Thus, the throughput performance of two-phase routing is much better in practice than indicated by the theoretical result that it is within 50% of the optimal scheme.

4.1 Linear Programming Formulations

In this section, we describe linear programming formulations for throughput maximization in two-phase routing. Note that for the case of equal split ratios, i.e., $\alpha_i = \frac{1}{n}$ for all $i \in N$, the demand between nodes i and j is $\frac{R_i+C_j}{n}$, and the problem reduces to the *maximum concurrent flow problem* [SM90].

4.1.1 Link Flow Based LP Formulation

For routing all matrices in $\lambda \cdot \mathcal{T}(\vec{R}, \vec{C})$ using two-phase routing, the traffic demand from node i to node j is

$$\lambda(\alpha_j R_i + \alpha_i C_j) = (\lambda\alpha_j)R_i + (\lambda\alpha_i)C_j$$

Notice that this introduces bilinear terms $\lambda\alpha_j$ and $\lambda\alpha_i$ in the expression for the demand values, since λ is also a variable in the problem formulation. To tackle this, we use the transformation $\bar{\alpha}_i = \lambda\alpha_i$ for all i. Then, the demand from node i to node j can be written as $\bar{\alpha}_j R_i + \bar{\alpha}_i C_j$, which is linear in the $\bar{\alpha}_i$ variables. To obtain the throughput in terms of the $\bar{\alpha}_i$ variables, we have

$$\sum_{i \in N} \bar{\alpha}_i = \lambda \sum_{i \in N} \alpha_i = \lambda$$

since $\sum_{i \in N} \alpha_i = 1$. Thus, when the traffic split ratios α'_j are not constrained to sum to 1, the throughput λ is equal to the sum of the α'_i values. To simplify notation, we will drop the bars in the transformed split ratios $\bar{\alpha}_i$. Thus, in our modified problem, the traffic split ratios α_i sum to the throughput λ and can be divided by λ to obtain the normalized traffic split ratios that sum to 1.

Another way to look at this transformation is that if the traffic split ratios α_i sum to a quantity $q \neq 1$, then they can be divided by q (normalized) so that they sum to 1, in which case all matrices in $q \cdot \mathcal{T}(\vec{R}, \vec{C})$ can be feasibly routed. Thus, the appropriate measure of throughput λ is the quantity q which is the sum of the traffic split ratios α_i.

We adopt the standard network flow terminology from Ahuja, Magnanti, and Orlin [AMO93]. Let x_e^{ij} denote the flow value on link e for routing $\alpha_j R_i + \alpha_i C_j$ amount of flow from source node i to destination node j. Then, the problem of two-phase routing so as to maximize throughput can be expressed as the following link indexed linear program:

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

subject to

$$\sum_{e \in E^+(k)} x_e^{ij} - \sum_{e \in E^-(k)} x_e^{ij} = \begin{cases} \alpha_j R_i + \alpha_i C_j & \text{if } k = i \\ -\alpha_j R_i - \alpha_i C_j & \text{if } k = j \\ 0 & \text{otherwise} \end{cases} \qquad \forall\, i, j, k \in N \quad (4.1)$$

$$\sum_{i,j \in N} x_e^{ij} \leq u_e \quad \forall\, e \in E \quad (4.2)$$

$$\alpha_i \geq 0 \quad \forall\, i \in N \quad (4.3)$$

$$x_e^{ij} \geq 0 \quad \forall\, e \in E, \quad \forall\, i, j \in N \quad (4.4)$$

Constraints (4.1) corresponding to the routing of $\alpha_j R_i + \alpha_i C_j$ amount of flow from node i to node j. Constraints (4.2) are the link capacity constraints.

By using per-source flow variables x_e^i instead of per source-destination variables x_e^{ij}, the number of flow variables in the above linear program can be reduced by a factor of n.

Let α_i^* be the α_i values in an optimal solution of the above linear program and let λ^* denote the optimum objective function value (throughput), i.e., $\lambda^* = \sum_i \alpha_i^*$. If $\lambda^* \geq 1$, then the problem is feasible for the network. The α_i^* values can be reduced by a factor of λ^* to get the actual split ratios and the explicit paths along which demands are routed can be determined from the solution of the above linear program using flow decomposition along paths [AMO93]. If the value $\lambda^* < 1$, then the problem is infeasible. Under such circumstances, one of two things can be done:

- The ingress-egress capacities R_i, C_j have to be scaled down by $\frac{1}{\lambda^*}$ and will then be feasible for routing under the given link capacities, or
- The link capacities can be scaled up by $\frac{1}{\lambda^*}$ and will then be feasible for routing all matrices in $\mathcal{T}(\vec{R}, \vec{C})$.

The above linear program is of polynomial size and can be solved in polynomial time using a general linear programming algorithm [S86]. It is amenable for solution with LP solvers like CPLEX [CPLEX].

4.1.2 Incorporating Node Capacity Constraints in IP-over-Optical Networks

Consider the deployment of our routing scheme in IP-over-Optical networks as discussed in Section 2.2.1. The end-to-end IP traffic traverses router-to-OXC (Optical Cross-connect) links not only at the source and destination nodes but also at the intermediate nodes. This router-to-OXC traffic at a node is bounded by the aggregate

connectivity of the IP router to the OXC at that node. Thus, we need to model such node capacity constraints in our problem formulation.

This is done by transforming the graph representation of the network as follows. Split each node into two sub-nodes, one representing the IP router and another the OXC at that node. All links incident at each node in the original graph are now incident at the corresponding OXC sub-node. Add links in either direction connecting the router and OXC sub-nodes with capacity equal to the given connectivity between router and OXC (in each direction) at that node. Traffic originates and terminates at the router sub-nodes in this transformed graph. Transit traffic traverses the OXC sub-nodes only, except at the intermediates nodes where it uses the router-to-OXC links to enter and leave the router sub-nodes. With this graph transformation, we can use the linear programming formulation in Section 4.1.1 as well as the combinatorial algorithms in Section 4.3 in the context of IP-over-Optical networks.

For the linear programming formulation, a somewhat simpler approach models the node capacities as additional constraints involving the traffic splits ratios α_i. (These constraints cannot be handled by the combinatorial algorithms though). Let u_i^1 and u_i^2 denote the router-to-OXC and OXC-to-router link capacities respectively at node i. (These are equal in practice due to the bi-directional nature of router and OXC ports.) The traffic from the OXC to the router at node k can be divided into two categories:

- Traffic as a result of node k being an intermediate node in two-phase routing, which is $\alpha_k \sum_{i \neq k} R_i$.
- Traffic that exits the network at node k, which is $\lambda C_k = (\sum_{i \in N} \alpha_i) C_k$.

Thus, the router-to-OXC link capacity constraint at node k is

$$\alpha_k \sum_{i \neq k} R_i + (\sum_{i \in N} \alpha_i) C_k \leq u_k^1 \quad \forall\, k \in N$$

which can be written as

$$C_k \sum_{i \neq k} \alpha_i + (C_k + \sum_{i \neq k} R_i)\alpha_k \leq u_k^1 \quad \forall\, k \in N$$

Similarly, the constraint for the OXC-to-router link capacity at each node can be written as:

$$R_k \sum_{i \neq k} \alpha_i + (R_k + \sum_{i \neq k} C_i)\alpha_k \leq u_k^2 \quad \forall\, k \in N$$

4.2 Need for Combinatorial Algorithms

In the previous section, we developed a polynomial size linear programming formulation for maximum throughput two-phase routing. We now motivate the need for developing combinatorial algorithms for the problem through consideration of the following aspects:

- *Faster Algorithms:* It is well known that running times of general linear programming based algorithms for network problems do not scale well with increasing network size (the complexity of the linear program also affects this). The combinatorial algorithms that we develop use iterative shortest path computations, have fast running times, and can provide performance guarantees that are arbitrarily close to optimality.

- *Routing with Restoration:* In order to make two-phase routing resilient to network failures, we consider three restoration mechanisms in Chapter 7. The optimization problems for two of these schemes do not admit polynomial size linear programming formulations. However, they admit fast combinatorial algorithms that extend the approach used for the unprotected case in this chapter.

- *Handling Path Constraints:* The link flow based linear programming formulation developed does not deal explicitly with paths (link flows in a solution may

be converted to flows along paths using flow decomposition [AMO93]). Hence, it cannot accommodate constraints like bounding path delay or constraining the available paths between any two nodes to some subset (e.g., as may be required by ISP policy and administrative constraints). Because our combinatorial algorithms work explicitly with paths, both types of constraints can be efficiently handled within the optimization framework. We explain this in Section 4.3.6.

4.3 Combinatorial Algorithms

In this section, we develop fast combinatorial algorithms that compute the traffic split ratios and routing of Phase 1 and Phase 2 paths up to $(1+\epsilon)$-factor of the optimal objective function value (maximum throughput) for any $\epsilon > 0$. 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 (link capacities and ingress-egress traffic capacities). The value of ϵ can be chosen to provide the desired degree of optimality for the solution. We also develop strongly polynomial time FPTAS where the running time does not depend on the size of the input numbers.

Our algorithms use a primal-dual approach that is adapted from the technique in Garg and Könemann [GK98] for solving the maximum multicommodity flow problem, where flows are augmented in the primal solution and dual variables are updated in an iterative manner.

We begin with a path flow based version of the linear programming formulation in Section 4.1.1. This (primal) program and its dual will be used to develop the combinatorial algorithms in this section.

4.3.1 Path Flow Based LP Formulation

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

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

subject to

$$\sum_{P \in \mathcal{P}_{ij}} x(P) = \alpha_j R_i + \alpha_i C_j \quad \forall\, i, j \in N \tag{4.5}$$

$$\sum_{P \ni e} x(P) \leq u_e \quad \forall\, e \in E \tag{4.6}$$

$$\alpha_i \geq 0 \quad \forall\, i \in N \tag{4.7}$$

$$x(P) \geq 0 \quad \forall\, P \in \mathcal{P}_{ij}, \quad \forall\, i, j \in N \tag{4.8}$$

In the next section, we state the dual of the linear program. In general, a network can have an exponential number of paths (in the size of the network). Hence, this (primal) linear program can have possibly exponential number of variables and its dual can have an exponential number of constraints -- they are both not suitable for solving the problem on medium to large sized networks. The usefulness of the primal and dual formulation is in designing a fast combinatorial algorithm for the problem.

4.3.2 Dual of Path Flow Based LP Formulation

The dual formulation of the linear program in Section 4.3.1 associates a variable π_{ij} with each demand constraint in (4.5) and a non-negative variable $w(e)$ with each link capacity constraint in (4.6). The dual program can be written as:

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

subject to

$$\sum_{e \in P} w(e) \geq \pi_{ij} \quad \forall\, P \in \mathcal{P}_{ij}, \quad \forall\, i, j \in N \tag{4.9}$$

$$\sum_{i \in N, i \neq k} R_i \pi_{ik} + \sum_{j \in N, \neq k} C_j \pi_{kj} \geq 1 \quad \forall\, k \in N \tag{4.10}$$

$$w(e) \geq 0 \quad \forall\, e \in E \tag{4.11}$$

Because of the nature of constraints (4.10), we can assume that the variables π_{ij} attain the maximum possible value given by constraints (4.9) in any optimal solution. Then, we have

$$\pi_{ij} = \min_{P \in \mathcal{P}_{ij}} \sum_{e \in P} w(e) \quad \forall\, i, j \in N$$

This allows us to eliminate the dual variables π_{ij}. Thus, we can remove constraints (4.9) and write constraints (4.10) as

$$\sum_{i \in N, i \neq k} R_i \min_{P \in \mathcal{P}_{ik}} \sum_{e \in P} w(e) + \sum_{j \in N, j \neq k} C_j \min_{P \in \mathcal{P}_{kj}} \sum_{e \in P} w(e) \geq 1 \quad \forall\, k \in N$$

The simplified dual problem can be written as:

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

subject to

$$\sum_{i \in N, i \neq k} R_i \min_{P \in \mathcal{P}_{ik}} \sum_{e \in P} w(e) + \sum_{j \in N, j \neq k} C_j \min_{P \in \mathcal{P}_{kj}} \sum_{e \in P} w(e) \geq 1 \quad \forall\, k \in N \tag{4.12}$$

$$w(e) \geq 0 \quad \forall\, e \in E \tag{4.13}$$

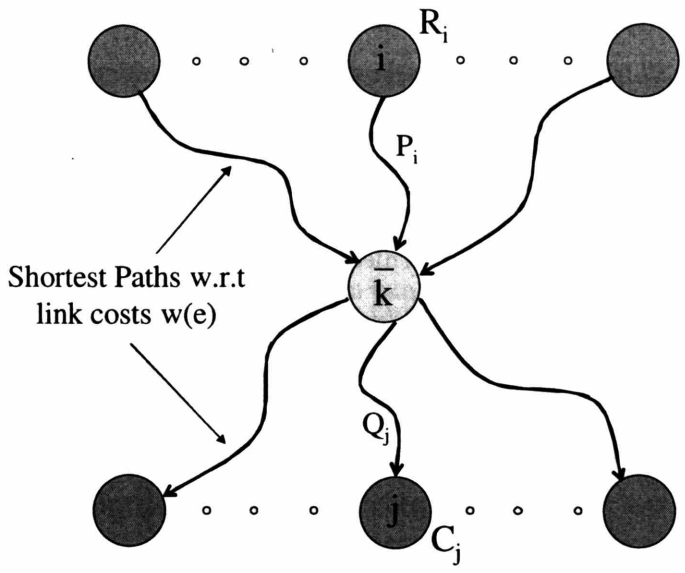


Figure 4-1: One Step in the Primal-Dual Computation for Maximum Throughput Two-Phase Routing.

4.3.3 Primal-Dual Scheme (FPTAS)

In this section, we use the primal and dual formulations of the path flow based linear program to develop a combinatorial algorithm for the problem.

For a given node k and weights $w(e)$, let $V(k)$ denote the left-hand-side (LHS) of constraint (4.12). Given the weights $w(e)$, note that $\min_{P \in \mathcal{P}_{ij}} \sum_{e \in P} w(e)$ is the cost of the shortest path from node i to node j under link costs $w(e)$. Thus, the values $V(k)$ for all $k \in N$ can be computed in polynomial time using a single all-pairs shortest path computation.

Given a set of weights $w(e)$, 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 $w(e) = \delta$ (the quantity δ depends on ϵ and is derived later). Repeat the following until the dual feasibility constraints (4.12) are satisfied:

1. Compute the node k for which $V(k)$ is minimum. This identifies a node $\bar{k}$ as well as paths P_i from node i to node $\bar{k}$ for all $i \neq \bar{k}$ and paths Q_j from node $\bar{k}$ to node j for all $j \neq \bar{k}$ (These are the corresponding shortest paths used in evaluating $V(\bar{k})$ as described above.) This is illustrated in Figure 4-1.

2. For a traffic split ratio of 1 for intermediate node $\bar{k}$, the traffic on path P_i is R_i for all $i \neq \bar{k}$ and the traffic on path Q_j is C_j for all $j \neq \bar{k}$. Using this, compute the traffic $f(e)$ on link e per unit split ratio $\alpha_{\bar{k}}$ for intermediate node $\bar{k}$ as

$$f(e) = \sum_{i \neq \bar{k}, P_i \ni e} R_i + \sum_{j \neq \bar{k}, Q_j \ni e} C_j \quad \forall\, e \in E \tag{4.14}$$

3. Compute the maximum value α for the traffic split ratio for intermediate node $\bar{k}$ that is consistent with (original) link capacity constraints for sending flow along paths P_i, Q_j as

$$\alpha = \min_{e \in E} \frac{u_e}{f(e)} \tag{4.15}$$

4. For this value α of the split ratio for intermediate node $\bar{k}$, send αR_i amount of flow from node i to node $\bar{k}$ along path P_i for all $i \neq \bar{k}$ and αC_j amount of flow from node $\bar{k}$ to node j along path Q_j for all $j \neq \bar{k}$. Compute the total flow on link e is $\Delta(e) = \alpha f(e)$ for all $e \in E$.

5. Update the weights $w(e)$ as

$$w(e) \leftarrow w(e)\left(1 + \frac{\epsilon \Delta(e)}{u_e}\right) \quad \forall\, e \in E$$

6. 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 flows and traffic 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.

The pseudo-code for the above procedure, called Algorithm MAX-THROUGHPUT-1, is provided below. Array $flow(e)$ keeps track of the flow sent on link e as the algorithm progresses. The variable G is initialized to 0 and remains less than 1 as long as the dual constraints are unsatisfied. After the **while** loop terminates, the maximum factor by which the capacity constraint on each link gets violated is computed into $scale\_fact$. Finally, the α_i values are divided by this factor and the resulting values are output.

Algorithm MAX-THROUGHPUT-1:

$\alpha_k \leftarrow 0 \quad \forall\, k \in N$;

$w(e) \leftarrow \delta \quad \forall\, e \in E$;

$flow(e) \leftarrow 0 \quad \forall\, e \in E$;

$G \leftarrow 0$;

while $G < 1$ **do**

For each $i, j \in N$, compute shortest path from i to j under link costs $w(e)$;

(Denote cost of shortest path from i to j by $SP(i,j)$.)

$V(k) \leftarrow \sum_{i \neq k} R_i SP(i,k) + \sum_{j \neq k} C_j SP(k,j) \quad \forall\, k \in N$;

$G \leftarrow \min_{k \in N} V(k)$;

$\bar{k} \leftarrow \arg\min_{k \in N} V(k)$;

if $G \geq 1$ **break** ;

(Denote shortest path from i to $\bar{k}$ by P_i for all $i \neq \bar{k}$

and shortest path from $\bar{k}$ to j by Q_j for all $j \neq \bar{k}$.)

$f(e) \leftarrow \sum_{i \neq \bar{k}, P_i \ni e} R_i + \sum_{j \neq \bar{k}, Q_j \ni e} C_j \quad \forall\, e \in E$;

$\alpha \leftarrow \min_{e \in E} \frac{u_e}{f(e)}$;

$\Delta(e) \leftarrow \alpha f(e) \quad \forall\, e \in E$;

$flow(e) \leftarrow flow(e) + \Delta(e) \quad \forall\, e \in E$;

$w(e) \leftarrow w(e)\left(1 + \frac{\epsilon \Delta(e)}{u_e}\right) \quad \forall\, e \in E$;

$\alpha_{\bar{k}} \leftarrow \alpha_{\bar{k}} + \alpha$;

end while

$scale\_fact \leftarrow \max_{e \in E} \frac{flow(e)}{u_e}$;

$\alpha_k \leftarrow \frac{\alpha_k}{scale\_fact}$ for all $k \in N$;

Output traffic split ratios α_k for all $k \in N$;

We next analyze the approximation guarantee and running time of Algorithm MAX-THROUGHPUT-1.

Analysis of Approximation Guarantee

We begin with some notation, then state some useful lemmas, and finally establish the main theorem for the approximation guarantee.

Let $M = \max_{k \in N}(\sum_{i \neq k} R_i + \sum_{j \neq k} C_j)$ and let M' denote the minimum non-zero value of the R_i's and C_j's.

Given a set of dual weights $w(e)$, let $D(w)$ denote the dual objective function value and let $\Gamma(w)$ denote the minimum value of the LHS of dual program constraint (4.12) over all nodes $k \in N$. Consider the dual problem. If $\Gamma(w) < 1$, then the weights $w(e)$ can be divided by $\Gamma(w)$ to make them feasible in which case the objective function value becomes $\frac{D(w)}{\Gamma(w)}$. If $\Gamma(w) > 1$, then the weights $w(e)$ remain feasible when they are divided by $\Gamma(w)$, but the objective function value decreases to $\frac{D(w)}{\Gamma(w)}$. Thus, solving the dual program is equivalent to finding a set of weights $w(e)$ such that $\frac{D(w)}{\Gamma(w)}$ is minimized (without the constraint $\Gamma(w) \geq 1$). Denote the optimal objective function

value of the latter by θ, i.e., $\theta = \min_w \frac{D(w)}{\Gamma(w)}$.

Let w_{t-1} denote the weight function at the beginning of iteration t of the **while** loop, and let A_{t-1} be the value of $\sum_{j \in N} \alpha_j$ (primal objective function) up to the end of iteration $t-1$. Suppose the algorithm terminates after iteration L. The following lemma upper bounds the value of $\Gamma(w)$ at the end of every iteration.

Lemma 4.3.1 *At the end of every iteration t, $1 \leq t \leq L$, of Algorithm MAX-THROUGHPUT-1, the following holds*

$$\Gamma(w_t) \leq n\delta M \prod_{j=1}^{t} [1 + \frac{\epsilon}{\theta}(A_j - A_{j-1})]$$

Proof: During iteration t, let $k = \bar{k}$ be the node for which $V(k)$ is minimum, let P_i, Q_j be the corresponding paths (as defined earlier) along which flow is augmented, and let α be the associated increment in $\alpha_{\bar{k}}$. Recall that the weights are updated as

$$w_t(e) = w_{t-1}(e)\left(1 + \frac{\epsilon \Delta(e)}{u_e}\right) \quad \forall\, e \in E$$

where $\Delta(e)$ is the total flow sent on link e during iteration t. Using this, we have

$$\begin{aligned}
D(w_t) &= \sum_{e \in E} u_e w_t(e) \\
&= \sum_{e \in E} u_e w_{t-1}(e) + \epsilon \sum_{e \in E} w_{t-1}(e)\Delta(e) \\
&= D(w_{t-1}) + \epsilon \sum_{e \in E} w_{t-1}(e)[\sum_{i \neq \bar{k}, P_i \ni e} \alpha R_i + \sum_{j \neq \bar{k}, Q_j \ni e} \alpha C_j] \\
&= D(w_{t-1}) + \epsilon\alpha \sum_{e \in E} w_{t-1}(e)[\sum_{i \neq \bar{k}. P_i \ni e} R_i + \sum_{j \neq \bar{k}, Q_j \ni e} C_j]
\end{aligned}$$

Interchanging the summations on the right-hand-side (RHS) of the above equation and first summing along links on paths P_i, Q_j, and then over i, j respectively, we can

rewrite the RHS of the above equation to obtain

$$\begin{aligned} D(w_t) &= D(w_{t-1}) + \epsilon\alpha[\sum_{i\neq\bar{k}} R_i \sum_{e\in P_i} w_{t-1}(e) + \sum_{j\neq\bar{k}} C_j \sum_{e\in Q_j} w_{t-1}(e)] \\ &= D(w_{t-1}) + \epsilon\alpha[\sum_{i\neq\bar{k}} R_i \min_{P\in\mathcal{P}_{i\bar{k}}} \sum_{e\in P} w_{t-1}(e) + \sum_{j\neq\bar{k}} C_j \min_{P\in\mathcal{P}_{kj}} \sum_{e\in P} w_{t-1}(e)] \qquad (4.16) \\ &= D(w_{t-1}) + \epsilon\alpha\Gamma(w_{t-1}) \\ &= D(w_{t-1}) + \epsilon(A_t - A_{t-1})\Gamma(w_{t-1}) \end{aligned}$$

The step leading to (4.16) follows from the fact that P_i, Q_j are shortest paths under link costs $w_{t-1}(e)$. The next step follows from the choice of node $k = \bar{k}$ for minimizing $V(k)$.

Using this last equation for each iteration down to the first one, we have

$$D(w_t) = D(w_0) + \epsilon \sum_{j=1}^{t} (A_j - A_{j-1})\Gamma(w_{j-1}) \qquad (4.17)$$

Now consider the weight function $w_t - w_0$. Note that $\Gamma(w_t - w_0)$ can be written as

$$\Gamma(w_t - w_0) = F(w_t) - F(w_0) \qquad (4.18)$$

where the function $F(w)$ is of the form of LHS of dual constraint (4.12), but not necessarily evaluated along shortest paths. Because any (simple) path in the network is at most $n-1$ hops in length, it follows that the quantity $\sum_{e\in P} w(e)$ is a sum of at most $(n-1)$ weights $w(e)$. Thus,

$$F(w_0) \leq \max_{k\in N}[\sum_{i\neq k}(n-1)\delta R_i + \sum_{j\neq k}(n-1)\delta C_j] < n\delta \max_{k\in N}(\sum_{i\neq k} R_i + \sum_{j\neq k} C_j) = n\delta M$$

Also, $F(w_t) \geq \Gamma(w_t)$. Using these in equation (4.18), we have

$$\Gamma(w_t - w_0) \geq \Gamma(w_t) - n\delta M \tag{4.19}$$

Since $D(w) = \sum_{e \in E} u_e w(e)$, we have $D(w_t - w_0) = D(w_t) - D(w_0)$. Since θ is the optimal dual objective function value, we have $\theta \leq \frac{D(w_t - w_0)}{\Gamma(w_t - w_0)}$. Using these and inequality (4.19), we have

$$D(w_t) - D(w_0) \geq \theta(\Gamma(w_t) - n\delta M)$$

Using this in equation (4.17), we have

$$\Gamma(w_t) \leq n\delta M + \frac{\epsilon}{\theta} \sum_{j=1}^{t} (A_j - A_{j-1}) \Gamma(w_{j-1}) \tag{4.20}$$

The property claimed in the lemma can now be proved using inequality (4.20) and mathematical induction on the iteration number t. For the induction basis case (iteration $t = 1$), we have $w_0(e) = \delta$ for all $e \in E$, hence

$$\Gamma(w_0) \leq \max_{k \in N} [\sum_{i \neq k} (n-1)\delta R_i + \sum_{j \neq k} (n-1)\delta C_j] = (n-1)\delta M < n\delta M$$

For the inductive step, suppose that the property is true for all iterations earlier than t. Then, we have

$$\begin{aligned} \Gamma(w_t) &\leq n\delta M + \frac{\epsilon}{\theta} \sum_{j=1}^{t} (A_j - A_{j-1}) n\delta M \prod_{i=1}^{j-1} [1 + \frac{\epsilon}{\theta}(A_i - A_{i-1})] \\ &= n\delta M \left(1 + \sum_{j=1}^{t} \frac{\epsilon}{\theta} (A_j - A_{j-1}) \prod_{i=1}^{j-1} [1 + \frac{\epsilon}{\theta}(A_i - A_{i-1})] \right) \end{aligned} \tag{4.21}$$

We now use the algebraic identity

$$\prod_{j=1}^{t}(1+a_j) = 1 + \sum_{j=1}^{t} a_j \prod_{i=1}^{j-1}(1+a_i)$$

Setting $a_j = \frac{\epsilon}{\theta}(A_j - A_{j-1})$ in inequality (4.21) and using the above identity, we obtain

$$\Gamma(w_t) \leq n\delta M \prod_{j=1}^{t}[1 + \frac{\epsilon}{\theta}(A_j - A_{j-1})]$$

This completes the inductive step. ■

We now estimate the factor by which the objective function value value A_L in the primal solution needs to be scaled when the algorithm terminates so as to ensure that link capacity constraints are not violated.

Lemma 4.3.2 *When Algorithm MAX-THROUGHPUT-1 terminates, the primal solution needs to be scaled by a factor of at most* $\log_{1+\epsilon} \frac{1+\epsilon}{\delta M'}$ *to ensure primal feasibility.*

Proof: We need to show that for every link e, the total flow on it is at most u_e when the primal solution is scaled by the above amount.

Consider any link e and associated weight $w(e)$. The value of $w(e)$ is updated when flow is augmented on link e. Let the sequence of *positive* flow augmentations (per iteration) on link e be $\Delta_1, \Delta_2, \ldots, \Delta_r$, where $r \leq L$. Let $\sum_{t=1}^{r} \Delta_t = \kappa u_e$, i.e., the total flow routed on link e exceeds its capacity by a factor of κ.

Because of the way in which α is chosen in accordance with equations (4.14)-(4.15), we have $\Delta_i \leq u_e$ for all i. Hence, the dual weight $w(e)$ is updated by a factor of at most $1+\epsilon$ after each iteration. Since the algorithm terminates when $\Gamma(w) \geq 1$, and since dual weights are updated by a factor of at most $1+\epsilon$ after each iteration, we have $\Gamma(w_L) < 1+\epsilon$. Note that just before each augmentation mentioned above, the weight $w(e)$, with coefficient *at least* M', is one of the summing components of

$\Gamma(w)$. Hence, $M' w_L(e) < 1 + \epsilon$. Also, the value of $w_L(e)$ is given by

$$w_L(e) = \delta \prod_{t=1}^{r} (1 + \frac{\Delta_t}{u_e} \epsilon)$$

Using the inequality $(1 + cx) \geq (1 + x)^c$ for all $x \geq 0$ and any $0 \leq c \leq 1$ and setting $x = \epsilon$ and $c = \frac{\Delta_t}{u_e} \leq 1$, we have

$$\begin{aligned} \frac{1+\epsilon}{M'} > w_L(e) &\geq \delta \prod_{t=1}^{r} (1+\epsilon)^{\Delta_t/u_e} \\ &= \delta (1+\epsilon)^{\sum_{t=1}^{r} \Delta_t/u_e} \\ &= \delta (1+\epsilon)^{\kappa} \end{aligned}$$

whence,

$$\kappa < \log_{1+\epsilon} \frac{1+\epsilon}{\delta M'}$$

■

The values of ϵ and δ are related, in the following theorem, to the approximation factor guarantee of Algorithm MAX-THROUGHPUT-1.

Theorem 4.3.3 *For any given $0 < \epsilon' \leq 0.5$, Algorithm MAX-THROUGHPUT-1 computes a solution with objective function value within $(1+\epsilon')$-factor of the optimum for*

$$\delta = \frac{1+\epsilon}{M'[(1+\epsilon)\frac{M}{M'}]^{1/\epsilon}} \quad \text{and} \quad \epsilon = \frac{\epsilon'}{2}$$

Proof: Using Lemma 4.3.1 and the inequality $1 + x \leq e^x$ for all $x \geq 0$, we have

$$\begin{aligned} \Gamma(w_t) &\leq n\delta M \prod_{j=1}^{t} e^{\frac{\epsilon}{\theta}(A_j - A_{j-1})} \\ &= n\delta M e^{\epsilon A_t/\theta} \end{aligned}$$

The simplification in the above step uses telescopic cancellation of the sum $(A_j - A_{j-1})$ over j. Since the algorithm terminates after iteration L, we must have $\Gamma(w_L) \geq 1$.

Thus,

$$1 \leq \Gamma(w_L) \leq n\delta M e^{\epsilon A_L/\theta}$$

whence,

$$\frac{\theta}{A_L} \leq \frac{\epsilon}{\ln \frac{1}{n\delta M}} \tag{4.22}$$

From Lemma 4.3.2, the objective function value of the feasible primal solution after scaling is at least

$$\frac{A_L}{\log_{1+\epsilon} \frac{1+\epsilon}{\delta M'}}$$

The approximation factor for the primal solution is at most the gap (ratio) between the dual and primal solutions. Using the lower bound on the primal solution and inequality (4.22), this is at most

$$\begin{aligned} \frac{\theta}{\frac{A_L}{\log_{1+\epsilon} \frac{1+\epsilon}{\delta M'}}} &\leq \frac{\epsilon \log_{1+\epsilon} \frac{1+\epsilon}{\delta M'}}{\ln \frac{1}{n\delta M}} \\ &= \frac{\epsilon}{\ln(1+\epsilon)} \frac{\ln \frac{1+\epsilon}{\delta M'}}{\ln \frac{1}{n\delta M}} \end{aligned}$$

The quantity $\ln \frac{1+\epsilon}{\delta M'} / \ln \frac{1}{n\delta M}$ equals $\frac{1}{1-\epsilon}$ for $\delta = \frac{1+\epsilon}{M'} / [(1+\epsilon)\frac{nM}{M'}]^{1/\epsilon}$. Using this value of δ, the approximation factor is upper bounded by $\frac{\epsilon}{(1-\epsilon)\ln(1+\epsilon)}$. This quantity is at most $1 + 2\epsilon$ for $\epsilon \leq 0.25$. Setting $\epsilon = \frac{\epsilon'}{2}$, we get the desired approximation ratio of $1 + \epsilon'$. ■

Analysis of Running Time

We now analyze the running time of Algorithm MAX-THROUGHPUT-1.

Theorem 4.3.4 *For any given $\epsilon > 0$ chosen to provide the desired approximation factor guarantee in accordance with Theorem 4.3.3, Algorithm MAX-THROUGHPUT-1 runs in time*

$$O\left(\frac{1}{\epsilon^2} nm(m + n \log n) \log \frac{nM}{M'}\right)$$

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

Proof: We first consider the running time of each iteration of the algorithm during which a node $\bar{k}$ and associated paths P_i, Q_j are chosen to augment flow. Selection of this node and the paths involves an all-pairs shortest path computation which can be implemented in $O(nm + n^2 \log n)$ time using Dijkstra's shortest path algorithm with Fibonacci heaps [AMO93]. All other operations within an iteration are absorbed (up to a constant factor) by the time taken for this all-pairs shortest path computation, leading to a total of $O(n(m + n \log n))$ time per iteration.

We next estimate the number of iterations before the algorithm terminates. Recall that in each iteration, flow is augmented along paths P_i, Q_j corresponding to the maximum value of intermediate node split ratio α such that the total flow $\Delta(e)$ sent on link e during that iteration is at most u_e. Thus, for at least one link e, $\Delta(e) = u_e$ and the weight $w(e)$ increases by a factor of $1 + \epsilon$. Accordingly, with each iteration, we can associate a weight $w(e)$ which increases by a factor of $1 + \epsilon$.

Consider the weight $w(e)$ for fixed $e \in E$. Since $w_0(e) = \delta$ and $w_L(e) < \frac{1+\epsilon}{M'}$ (as deduced in the proof of Lemma 4.3.2), the maximum number of times that this weight can be associated with any iteration is

$$\log_{1+\epsilon} \frac{1+\epsilon}{\delta M'} = \frac{1}{\epsilon}(1 + \log_{1+\epsilon} \frac{nM}{M'}) = O(\frac{1}{\epsilon} \log_{1+\epsilon} \frac{nM}{M'})$$

Since there are a total of m weights $w(e)$, hence the total number of iterations is upper bounded by $O(\frac{m}{\epsilon} \log_{1+\epsilon} \frac{nM}{M'})$. Multiplying this by the running time per iteration, we obtain the overall algorithm running time as $O(\frac{1}{\epsilon} nm(m + n \log n) \log_{1+\epsilon} \frac{nM}{M'})$.

Now, for any $0 < \epsilon < 1$, $\ln(1 + \epsilon) \geq \epsilon - \frac{1}{2}\epsilon^2 \geq \epsilon - \frac{1}{2}\epsilon = \frac{1}{2}\epsilon$. Also, $\ln(1 + \epsilon) \leq \epsilon$. This implies $\ln(1 + \epsilon) = \Theta(\epsilon)$, whence $\log_{1+\epsilon} \frac{nM}{M'} = O(\frac{1}{\epsilon} \log \frac{nM}{M'})$. Substituting this back into the earlier expression for running time, we obtain the running time in the theorem. Note that $\log \frac{nM}{M'}$ is polynomial in $\log n$ and the number of bits used to

represent the R_i and C_j values. ■

4.3.4 Speedup of Factor of n

We can reduce the running time of the Algorithm MAX-THROUGHPUT-1 by a factor of n by using a technique similar to that used by Fleischer [F00] for speeding up the maximum multicommodity flow algorithm in [GK98]. In Algorithm MAX-THROUGHPUT-1, the time for augmenting flow in each iteration is dominated by an all-pairs shortest path computation. We use this computation to identify an intermediate node k (and associated shortest paths P_i, Q_j) for which the LHS of dual constraint (4.12) is minimum.

The basic idea in the speedup is to keep augmenting flow associated with one intermediate node k so long as the value of LHS of (4.12) for that intermediate node is within a factor of $(1+\epsilon)$ of the minimum value of the same over all intermediate nodes. We then cycle through all intermediate nodes and repeat all over again until termination. Computing the value of LHS of (4.12) for one intermediate node k involves two single-source shortest path computations – one for the shortest path tree directed out of k, and the other for the shortest path tree directed towards k. Thus, each flow augmentation requires two single-source shortest path computations instead of an all-pairs shortest path computation – this is how the speedup is achieved. We now develop the details.

To check that the value of LHS of (4.12) for an intermediate node is within a factor of $(1+\epsilon)$ of the minimum value of the same over all intermediate nodes, we need to maintain a lower bound estimate $\gamma(w_t)$ of the latter, where t refers to the value of dual weights $w_t(e)$ after augmentation number t. Let $M'' = \min_{k \in N}(\sum_{i \neq k} R_i + \sum_{j \neq k} C_j)$. The initial lower bound estimate is $\gamma(w_0) = \delta \min_{k \in N}(\sum_{i \neq k} R_i + \sum_{j \neq k} C_j) = \delta M''$, since the shortest paths from intermediate node k to all other nodes and from all other nodes to node k are at least one hop in length.

The strategy for maintaining and updating the lower bound estimate $\gamma(w)$ is as

follows. We keep augmenting flow associated with intermediate node k and updating weights $w(e)$ in a multiplicative fashion as before so long as the value of LHS of (4.12) for node k is less than $\min(1, (1+\epsilon)\gamma(w))$. We repeat this for all other intermediate nodes. At this point, we know that the minimum value of LHS of (4.12) over all intermediate nodes is at least $(1+\epsilon)\gamma(w)$. Accordingly, we update $\gamma(w) \leftarrow (1+\epsilon)\gamma(w)$. We will use the term *phase* to mean the sequence of flow augmentations between successive updates to $\gamma(w)$ (i.e., during cycling once through all intermediate nodes). Each flow augmentation will be called an *iteration.*

In order to terminate the algorithm after dual constraint (4.12) is satisfied, it is sufficient to stop after phase ρ when $\delta M''(1+\epsilon)^\rho < 1+\epsilon$. Hence, the total number of phases is $\rho = \lfloor \log_{1+\epsilon} \frac{1+\epsilon}{\delta M''} \rfloor$.

The pseudo-code for the above procedure, called Algorithm MAX-THROUGHPUT-2, is provided below. Array $flow(e)$ keeps track of the traffic on link e as the algorithm progresses. The variable G is initialized to 0 and remains less than 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 are output.

Algorithm MAX-THROUGHPUT-2:

$\alpha_k \leftarrow 0 \quad \forall\, k \in N$;

$w(e) \leftarrow \delta \quad \forall\, e \in E$;

$flow(e) \leftarrow 0 \quad \forall\, e \in E$;

for $i = 1$ to $\lfloor \log_{1+\epsilon} \frac{1+\epsilon}{\delta M''} \rfloor$ **do**

for $k = 1$ to n **do**

while true do

Compute shortest paths from i to k for all $i \neq$k and from k to j for all $j \neq k$ under link costs $w(e)$;

(Denote by $SP(i,j)$ the cost of the shortest path from i to j.)

$V(k) \leftarrow \sum_{i \neq k} R_i SP(i,k) + \sum_{j \neq k} C_j SP(k,j)$;

if $V(k) \geq \min(1, \delta M''(1+\epsilon)^i)$ **break** ;

(Denote shortest path from i to k by P_i for all $i \neq k$

and shortest path from k to j by Q_j for all $j \neq k$.) ;

$f(e) \leftarrow \sum_{i \neq \bar{k}, P_i \ni e} R_i + \sum_{j \neq \bar{k}, Q_j \ni e} C_j \quad \forall\, e \in E$;

$\alpha \leftarrow \min_{e \in E} \frac{u_e}{f(e)}$;

$\Delta(e) \leftarrow \alpha f(e) \quad \forall\, e \in E$;

$flow(e) \leftarrow flow(e) + \Delta(e) \quad \forall\, e \in E$;

$w(e) \leftarrow w(e)\left(1 + \frac{\epsilon \Delta(e)}{u_e}\right) \quad \forall\, e \in E$;

$\alpha_k \leftarrow \alpha_k + \alpha$;

end while

end for

end for

$scale(e) \leftarrow \frac{flow(e)}{u_e}$ for all $e \in E$;

$scale\_max \leftarrow \max_{e \in E} scale(e)$;

$\alpha_k \leftarrow \frac{\alpha_k}{scale\_max}$ for all $k \in N$;

Output traffic split ratios α_k for all $k \in N$;

We next establish the approximation guarantee and running time of Algorithm MAX-THROUGHPUT-2. The approach is similar to that for Algorithm MAX-THROUGHPUT-1.

Analysis of Approximation Guarantee

We use the same notation as in Section 4.3.3. Let L be the total number of flow augmentations (or, iterations). Recall that the number of phases is $\rho = \lfloor \log_{1+\epsilon} \frac{1+\epsilon}{\delta M''} \rfloor$. The following lemma upper bounds the value of $D(w)$ at the end of every iteration.

Lemma 4.3.5 *At the end of flow augmentation (or, iteration) number t, $1 \leq t \leq L$, of Algorithm MAX-THROUGHPUT-2, the following holds*

$$\Gamma(w_t) \leq n\delta M \prod_{j=1}^{t} [1 + \frac{\epsilon(1+\epsilon)}{\theta}(A_j - A_{j-1})]$$

Proof: For the flow augmented during iteration t, let k be the intermediate node, let P_i, Q_j be the corresponding paths along which flow is augmented, and let α be the associated increment in α_k. Starting as in the proof of Lemma 4.3.1, we first establish equation (4.16) which is

$$D(w_t) = D(w_{t-1}) + \epsilon\alpha[\sum_{i \neq k} R_i \min_{P \in \mathcal{P}_{ik}} \sum_{e \in P} w_{t-1}(e) + \sum_{j \neq k} C_j \min_{P \in \mathcal{P}_{kj}} \sum_{e \in P} w_{t-1}(e)] \quad (4.23)$$

Because the value of LHS of dual constraint (4.12) for intermediate node k is within $(1+\epsilon)$ of the minimum value $\Gamma(w_{t-1})$ of LHS of (4.12) over all intermediate nodes, we have

$$\sum_{i \neq k} R_i \min_{P \in \mathcal{P}_{ik}} \sum_{e \in P} w_{t-1}(e) + \sum_{j \neq k} C_j \min_{P \in \mathcal{P}_{kj}} \sum_{e \in P} w_{t-1}(e) \leq (1+\epsilon)\Gamma(w_{t-1})$$

Using this in (4.23), we have

$$\begin{aligned} D(w_t) &\leq D(w_{t-1}) + \epsilon(1+\epsilon)\alpha\Gamma(w_{t-1}) \\ &= D(w_{t-1}) + \epsilon(1+\epsilon)(A_t - A_{t-1})\Gamma(w_{t-1}) \end{aligned}$$

Using this for each iteration down to the first one, we have

$$D(w_t) \leq D(w_0) + \epsilon(1+\epsilon) \sum_{j=1}^{t} (A_j - A_{j-1})\Gamma(w_{j-1}) \quad (4.24)$$

This is of the same form as equation (4.17) except that ϵ is replaced by $\epsilon(1+\epsilon)$ and there is inequality instead of equality. By following the same steps from here onwards

as in Lemma 4.3.1, we obtain the property claimed in this lemma. ■

We now estimate the factor by which the objective function value value A_L in the primal solution needs to be scaled when the algorithm terminates so as to ensure that link capacity constraints are not violated. Because this algorithm augments flows and updates dual weights in a manner similar to that of Algorithm MAX-THROUGHPUT-1 and also terminates when dual feasibility constraints are satisfied, the proof of the lemma below is identical to that for Lemma 4.3.2.

Lemma 4.3.6 *When Algorithm MAX-THROUGHPUT-2 terminates, the primal solution needs to be scaled by a factor of at most* $\log_{1+\epsilon} \frac{1+\epsilon}{\delta M'}$ *to ensure primal feasibility.*

The values of ϵ and δ are related, in the following theorem, to the approximation factor guarantee of Algorithm MAX-THROUGHPUT-2.

Theorem 4.3.7 *For any given* $0 < \epsilon' \leq 0.42$, *Algorithm MAX-THROUGHPUT-2 computes a solution with objective function value within* $(1+\epsilon')$*-factor of the optimum for*

$$\delta = \frac{1+\epsilon}{M'[(1+\epsilon)\frac{M}{M'}]^{1/\epsilon}} \quad \textit{and} \quad \epsilon = \frac{\epsilon'}{3}$$

Proof: The main difference from the proof of Theorem 4.3.3 is that we use the two lemmas established in this section. Using Lemma 4.3.5 and the inequality $1 + x \leq e^x$ for all $x \geq 0$, we have

$$\begin{aligned} \Gamma(w_t) &\leq n\delta M \prod_{j=1}^{t} e^{\frac{\epsilon(1+\epsilon)}{\theta}(A_j - A_{j-1})} \\ &= n\delta M e^{\epsilon(1+\epsilon)A_t/\theta} \end{aligned}$$

Since the algorithm terminates after iteration L, we must have $\Gamma(w_L) \geq 1$. Thus,

$$1 \leq \Gamma(w_L) \leq n\delta M e^{\epsilon(1+\epsilon)A_L/\theta}$$

whence,

$$\frac{\theta}{A_L} \leq \frac{\epsilon(1+\epsilon)}{\ln \frac{1}{n\delta M}} \tag{4.25}$$

From Lemma 4.3.6, the objective function value of the feasible primal solution after scaling is at least

$$\frac{A_L}{\log_{1+\epsilon} \frac{1+\epsilon}{\delta M'}}$$

The approximation factor for the primal solution is at most the gap (ratio) between the dual and primal solutions. Using the lower bound on the primal solution and inequality (4.25), this is at most

$$\begin{aligned} \frac{\theta}{\frac{A_L}{\log_{1+\epsilon} \frac{1+\epsilon}{\delta M'}}} &\leq \frac{\epsilon(1+\epsilon)\log_{1+\epsilon} \frac{1+\epsilon}{\delta M'}}{\ln \frac{1}{n\delta M}} \\ &= \frac{\epsilon(1+\epsilon)}{\ln(1+\epsilon)} \frac{\ln \frac{1+\epsilon}{\delta M'}}{\ln \frac{1}{n\delta M}} \end{aligned}$$

The quantity $\ln \frac{1+\epsilon}{\delta M'} / \ln \frac{1}{n\delta M}$ equals $\frac{1}{1-\epsilon}$ for $\delta = \frac{1+\epsilon}{M'} / [(1+\epsilon)\frac{nM}{M'}]^{1/\epsilon}$. Using this value of δ, the approximation factor is upper bounded by $\frac{\epsilon(1+\epsilon)}{(1-\epsilon)\ln(1+\epsilon)}$. This is at most $1+3\epsilon$ for $\epsilon \leq 0.14$. Setting $\epsilon = \frac{\epsilon'}{3}$, we get the desired approximation ratio of $1+\epsilon'$. ■

Analysis of Running Time

We now establish the speedup of factor of n for the running time of Algorithm MAX-THROUGHPUT-2.

Theorem 4.3.8 *For any $\epsilon > 0$ chosen to provide the desired approximation factor guarantee in accordance with Theorem 4.3.7, Algorithm MAX-THROUGHPUT-2 runs in time*

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

which is a speedup of factor of n compared to Algorithm MAX-THROUGHPUT-1.

Proof: We first consider the time taken for each flow augmentation. For a given intermediate node k, computation of shortest paths P_i, Q_j that minimize the LHS of dual constraint (4.12) for node k involves two single-source shortest path computations – one for the shortest path tree directed out of k, and the other for the shortest path tree directed towards k. This can be implemented in $O(m + n \log n)$ time using Dijkstra's shortest path algorithm with Fibonacci heaps [AMO93]. All other operations associated with a flow augmentation are absorbed (up to a constant factor) by the time taken for two single-source shortest path computations, leading to a total of $O(m + n \log n)$ time per flow augmentation.

The number of flow augmentations before the algorithm terminates is the same as that for Algorithm MAX-THROUGHPUT-1 and was shown in the proof of Theorem 4.3.4 to be upper bounded by $O(\frac{1}{\epsilon} m \log_{1+\epsilon} \frac{nM}{M'})$. Multiplying this by the time per flow augmentation, we obtain the total time for all flow augmentations as $O(\frac{1}{\epsilon} m(m + n \log n) \log_{1+\epsilon} \frac{nM}{M'})$. Since $\ln(1+\epsilon) = \Theta(\epsilon)$, this is $O(\frac{1}{\epsilon^2} m(m + n \log n) \log \frac{nM}{M'})$.

It remains to account for the single-source shortest path computations that do not lead to flow augmentations. This happens once per intermediate node per phase right before we move on to the next intermediate node in that phase. Thus, the total time spent on such single-source shortest path computations is of the order of

$$\begin{aligned}
\rho n(m + n \log n) &= n(m + n \log n) \log_{1+\epsilon} \frac{1+\epsilon}{\delta M''} \\
&= n(m + n \log n) \log_{1+\epsilon} \frac{M'}{M''} \left(\frac{(1+\epsilon)nM}{M'} \right)^{1/\epsilon} \\
&= n(m + n \log n) \left(\log_{1+\epsilon} \frac{M'}{M''} + \frac{1}{\epsilon}(1 + \log_{1+\epsilon} \frac{nM}{M'}) \right) \\
&= n(m + n \log n) \log_{1+\epsilon} \frac{M'}{M''} + O\left(\frac{1}{\epsilon} n(m + n \log n) \log_{1+\epsilon} \frac{nM}{M'} \right)
\end{aligned}$$

Since the quantity M'' contains at least one non-zero value of R_i, C_j, we have $M'' \geq M'$. Hence, $\log_{1+\epsilon} \frac{M'}{M''} < 0$. Hence, the total time spent on single-source shortest path computations that do not lead to flow augmentations is $O\left(\frac{1}{\epsilon} n(m + n \log n) \log_{1+\epsilon} \frac{nM}{M'}\right)$.

This is subsumed by the total time for flow augmentations. This completes the proof.

■

4.3.5 Strongly Polynomial Time FPTAS

A strongly polynomial time algorithm is a polynomial time algorithm that has no dependence on the size of the input numbers of the problem. In our case, the input numbers are the link capacities and ingress-egress traffic capacities. The primal dual scheme of Section 4.3.3 can be made strongly polynomial time by terminating when the dual objective function value becomes greater than or equal to 1 (instead of when the dual solution becomes feasible). This is used as a terminating condition for primal-dual schemes for some multicommodity flow and fractional packing problems in [GK98]. The modifications to Algorithm MAX-THROUGHPUT-1 for making its running time strongly polynomial are as follows:

- Terminate when the dual objective function value becomes greater than or equal to 1 (instead of when the dual solution becomes feasible),
- Use $\delta = \frac{1+\epsilon}{[(1+\epsilon)m]^{1/\epsilon}}$, and
- Initialize dual weights $w(e) = \frac{\delta}{u_e}$ for all $e \in E$.

The analysis of the approximation guarantee and running time follows the same approach as that for Algorithm MAX-THROUGHPUT-1.

Analysis of Approximation Guarantee

We use the same notation as in Section 4.3.3. Let the total number of iterations be L. The following lemma upper bounds the value of $D(w)$ at the end of every iteration.

Lemma 4.3.9 *At the end of every iteration t, $1 \leq t \leq L$, of Algorithm MAX-THROUGHPUT-3, the following holds*

$$D(w_t) \leq m\delta \prod_{j=1}^{t} [1 + \frac{\epsilon}{\theta}(A_j - A_{j-1})]$$

Proof: Starting as in the proof of Lemma 4.3.1, we first establish equation (4.17) which is

$$D(w_t) = D(w_0) + \epsilon \sum_{j=1}^{t} (A_j - A_{j-1})\Gamma(w_{j-1}) \tag{4.26}$$

From the definition of θ, we have $\theta \leq \frac{D(w_{j-1})}{\Gamma(w_{j-1})}$, whence $\Gamma(w_{j-1}) \leq \frac{1}{\theta} D(w_{j-1})$. Also, $D(w_0) = m\delta$. Using these in equation (4.26), we have

$$D(w_t) \leq m\delta + \frac{\epsilon}{\theta} \sum_{j=1}^{t} (A_j - A_{j-1}) D(w_{j-1}) \tag{4.27}$$

The property claimed in the lemma can now be proved using inequality (4.27) and mathematical induction on the iteration number t. The method is similar to that used in the proof of Lemma 4.3.1. ■

We now estimate the factor by which the objective function value value A_L in the primal solution needs to be scaled when the algorithm terminates so as to ensure that link capacity constraints are not violated.

Lemma 4.3.10 *When Algorithm MAX-THROUGHPUT-3 terminates, the primal solution needs to be scaled by a factor of at most* $\log_{1+\epsilon} \frac{1+\epsilon}{\delta}$ *to ensure primal feasibility.*

Proof: We need to show that for every link e, the total flow on it is at most u_e when the primal solution is scaled by the above amount.

Consider any link e and let the sequence of flow augmentations (per iteration) on link e be $\Delta_1, \Delta_2, \ldots, \Delta_L$. Let $\sum_{t=1}^{L} \Delta_t = \kappa u_e$, i.e., the total flow routed on link e exceeds its capacity by a factor of κ.

Since the algorithm terminates when $D(w) \geq 1$, and since dual weights are updated by a factor of at most $1+\epsilon$ after each iteration, we have $D(w_L) < 1+\epsilon$. Since the weight $w(e)$, with coefficient u_e, is one of the summing components of $D(w)$, we have $u_e w_L(e) < 1+\epsilon$. Also, the value of $w_L(e)$ is given by

$$w_L(e) = \frac{\delta}{u_e} \prod_{t=1}^{L} (1 + \frac{\Delta_t}{u_e}\epsilon)$$

With the above changes, we complete the proof using the same argument as in Lemma 4.3.2. Using the inequality $(1+cx) \geq (1+x)^c$ for all $x \geq 0$ and any $0 \leq c \leq 1$ and setting $x = \epsilon$ and $c = \frac{\Delta_t}{u_e} \leq 1$, we have

$$\begin{aligned} \frac{1+\epsilon}{u_e} > w_L(e) &\geq \frac{\delta}{u_e} \prod_{t=1}^{L} (1+\epsilon)^{\Delta_t/u_e} \\ &= \frac{\delta}{u_e} (1+\epsilon)^{\sum_{t=1}^{L} \Delta_t/u_e} \\ &= \frac{\delta}{u_e} (1+\epsilon)^{\kappa} \end{aligned}$$

whence,

$$\kappa < \log_{1+\epsilon} \frac{1+\epsilon}{\delta}$$

■

The values of ϵ and δ are related, in the following theorem, to the approximation factor guarantee of Algorithm MAX-THROUGHPUT-3.

Theorem 4.3.11 *For any given $0 < \epsilon' \leq 0.5$, Algorithm MAX-THROUGHPUT-3 computes a solution with objective function value within $(1+\epsilon')$-factor of the optimum for*

$$\delta = \frac{1+\epsilon}{[(1+\epsilon)m]^{1/\epsilon}} \quad \textit{and} \quad \epsilon = \frac{\epsilon'}{2}$$

Proof: The main difference from the proof of Theorem 4.3.3 is the use of the two lemmas proved in this section. Using Lemma 4.3.9 and the inequality $1 + x \leq e^x$ for

all $x \geq 0$, we have

$$\begin{aligned} D(w_t) &\leq m\delta \prod_{j=1}^{t} e^{\frac{\epsilon}{\theta}(A_j - A_{j-1})} \\ &= m\delta e^{\epsilon A_t/\theta} \end{aligned}$$

Since the algorithm terminates after iteration L, we must have $D(w_L) \geq 1$. Thus,

$$1 \leq D(w_L) \leq m\delta e^{\epsilon A_L/\theta}$$

whence,

$$\frac{\theta}{A_L} \leq \frac{\epsilon}{\ln \frac{1}{m\delta}} \tag{4.28}$$

From Lemma 4.3.10, the objective function value of the feasible primal solution after scaling is at least

$$\frac{A_L}{\log_{1+\epsilon} \frac{1+\epsilon}{\delta}}$$

The approximation factor for the primal solution is at most the gap (ratio) between the dual and primal solutions. Using the lower bound on the primal solution and inequality (4.28), this is at most

$$\begin{aligned} \frac{\theta}{\frac{A_L}{\log_{1+\epsilon} \frac{1+\epsilon}{\delta}}} &\leq \frac{\epsilon \log_{1+\epsilon} \frac{1+\epsilon}{\delta}}{\ln \frac{1}{m\delta}} \\ &= \frac{\epsilon}{\ln(1+\epsilon)} \frac{\ln \frac{1+\epsilon}{\delta}}{\ln \frac{1}{m\delta}} \end{aligned}$$

The quantity $\ln \frac{1+\epsilon}{\delta} / \ln \frac{1}{m\delta}$ equals $\frac{1}{1-\epsilon}$ for $\delta = (1+\epsilon)/[(1+\epsilon)m]^{1/\epsilon}$. Using this value of δ, the approximation factor is upper bounded by $\frac{\epsilon}{(1-\epsilon)\ln(1+\epsilon)}$. This quantity is at most $1 + 2\epsilon$ for $\epsilon \leq 0.25$. Setting $\epsilon = \frac{\epsilon'}{2}$, we get the desired approximation ratio of $1 + \epsilon'$. ■

Analysis of Running Time

We now show that the running time of Algorithm MAX-THROUGHPUT-3 is strongly polynomial.

Theorem 4.3.12 *For any given $\epsilon > 0$ chosen to provide the desired approximation factor guarantee in accordance with Theorem 4.3.11, Algorithm MAX-THROUGHPUT-3 runs in time*

$$O\left(\frac{1}{\epsilon^2} nm(m + n\log n)\log m\right)$$

which is strongly polynomial.

Proof: The proof is similar to that for Theorem 4.3.4. As before, the running time of each iteration of the algorithm during which a node $\bar{k}$ and associated paths P_i, Q_j are chosen to augment flow is $O(n(m + n\log n))$. Also, with each iteration, we can associate a weight $w(e)$ which increases by a factor of $1 + \epsilon$.

Consider the weight $w(e)$ for fixed $e \in E$. Since $w_0(e) = \frac{\delta}{u_e}$ and $w_L(e) < \frac{1+\epsilon}{u_e}$ (as deduced in the proof of Lemma 4.3.10), the maximum number of times that this weight can be associated with any iteration is

$$\log_{1+\epsilon} \frac{1+\epsilon}{\delta} = \frac{1}{\epsilon}(1 + \log_{1+\epsilon} m) = O(\frac{1}{\epsilon}\log_{1+\epsilon} m)$$

Since there are a total of m weights $w(e)$, hence the total number of iterations is upper bounded by $O(\frac{1}{\epsilon} m \log_{1+\epsilon} m)$. Multiplying this by the running time per iteration, we obtain the overall algorithm running time as $O(\frac{1}{\epsilon} nm(m+n\log n)\log_{1+\epsilon} m) = O(\frac{1}{\epsilon^2} nm(m + n\log n)\log m)$. ■

4.3.6 Handling Path Constraints

We addressed the issue of increase in end-to-end path delay in two-phase routing compared to shortest path routing in Section 2.3.2. For delay sensitive applications that need to meet strict delay bounds, routing with end-to-end delay guarantees

needs to be accommodated within the optimization framework. An end-to-end delay guarantee of at most d in two-phase routing can be enforced by constraining the Phase 1 and Phase 2 paths to each have delays of at most $\frac{d}{2}$. This can be accommodated in our primal dual framework as follows.

We are given the delay associated with each link in the network. The delay associated with a path is the sum of the delay values on all its links. In our primal dual framework, we replace the shortest path computation from node i to node j by a *delay constrained shortest path computation* with delay at most $\frac{d}{2}$. While this is an $\mathcal{NP}$-hard problem in general [GJ79], it can be solved exactly in pseudo-polynomial time (for example, using dynamic programming). We can also obtain approximate solutions in polynomial time by either (i) relaxing the optimality of the path cost by computing a delay constrained path with cost at most $(1+\epsilon)$-times that of the optimum, or (ii) relaxing the delay constraint and computing a shortest cost path with delay at most $(1+\epsilon)$ times the allowed delay, for any given $\epsilon > 0$. Fully polynomial time approximation schemes for (i) appear in [H92, P93] and for (ii) in [GRKL01].

Our combinatorial algorithms can also handle other types of constraints on the selection of Phase 1 and Phase 2 paths. For example, ISPs can restrict the available paths between any two nodes in the network based on administrative and policy constraints. These constraints could be dictated by Service Level Agreements (SLAs) of the ISP with its customers. These constraints can be easily incorporated into our combinatorial algorithms by making the set $\mathcal{P}_{ij}$ contain only the allowed paths from node to i to j.

4.4 Maximum Throughput Routing with Cost Bound

In this section, we consider adding a cost constraint to the maximum throughput routing problem for two-phase routing. We are given link costs c_e per unit traffic on each link e and a total cost bound C. The problem consists of computing a maximum

throughput solution for two-phase routing that has cost at most C. Once we have an efficient algorithm for this problem, we can use binary search on the cost bound C to obtain a minimum cost solution for two-phase routing that has a throughput of (at least) 1. (Because of the use of binary search on the cost bound C, a trade-off needs to be made between the computation time and the degree of precision of the obtained minimum cost.) This solves the capacitated version of minimum cost two-phase routing.

We begin with a linear programming formulation for the problem by adding an additional cost constraint to the linear program of Section 4.1.1. We then extend the primal-dual approach for the strongly polynomial time combinatorial algorithm in Section 4.3.5 to handle this additional cost constraint.

4.4.1 Link Flow Based LP Formulation

Let x_e^{ij} denote the flow value on link e for routing $\alpha_j R_i + \alpha_i C_j$ amount of flow from source node i to destination node j. As before, the appropriate measure of throughput is the quantity $\lambda = \sum_{i \in N} \alpha_i$ *when the traffic split ratios* α_j *are not constrained to sum to* 1. The problem of maximum throughput two-phase routing subject to the constraint that the total cost of the solution is C can be expressed as the following link indexed linear program:

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

subject to

$$\sum_{e \in E^+(k)} x_e^{ij} - \sum_{e \in E^-(k)} x_e^{ij} = \begin{cases} \alpha_j R_i + \alpha_i C_j & \text{if } k = i \\ -\alpha_j R_i - \alpha_i C_j & \text{if } k = j \\ 0 & \text{otherwise} \end{cases} \qquad \forall\, i, j, k \in N \quad (4.29)$$

$$\sum_{i,j \in N} x_e^{ij} \leq u_e \quad \forall\, e \in E \quad (4.30)$$

$$\sum_{e \in E} c_e \sum_{i,j \in N} x_e^{ij} \leq C \tag{4.31}$$

$$\alpha_i \geq 0 \quad \forall\, i \in N \tag{4.32}$$

$$x_e^{ij} \geq 0 \quad \forall\, e \in E, \quad \forall\, i,j \in N \tag{4.33}$$

Constraints (4.29) corresponding to the routing of $\alpha_j R_i + \alpha_i C_j$ amount of flow from node i to node j. Constraints (4.30) are the link capacity constraints. Constraint (4.31) requires that the total cost of the solution is at most C.

By using per-source flow variables x_e^i instead of per source-destination variables x_e^{ij}, the number of flow variables in the above linear program can be reduced by a factor of n.

4.4.2 Path Flow Based LP Formulation

We convert the above link flow based linear program into a path flow based linear program. This program and its dual will be used to develop the combinatorial algorithm for the problem.

Let $\mathcal{P}_{ij}$ denote the set of all (simple) paths from node i to node j. Let $x(P)$ denote the traffic on path P. Then, the problem of maximum throughput two-phase routing with a total cost of at most C can be expressed as the following path indexed linear program:

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

subject to

$$\sum_{P \in \mathcal{P}_{ij}} x(P) = \alpha_j R_i + \alpha_i C_j \quad \forall\, i,j \in N \tag{4.34}$$

$$\sum_{P \ni e} x(P) \leq u_e \quad \forall\, e \in E \tag{4.35}$$

$$\sum_{e \in E} c_e \sum_{P \ni e} x(P) \leq C \tag{4.36}$$

$$\alpha_i \geq 0 \ \forall \ i \in N \tag{4.37}$$

$$x(P) \geq 0 \ \forall \ P \in \mathcal{P}_{ij}, \ \forall \ i, j \in N \tag{4.38}$$

4.4.3 Dual of Path Flow Based LP Formulation

The dual formulation of the path indexed linear program associates a variable π_{ij} with each demand constraint in (4.34), a non-negative variable $w(e)$ with each link capacity constraint in (4.35), and a non-negative variable y with constraint (4.36). The dual program can be written as:

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

subject to

$$\sum_{e \in P} w(e) + y \sum_{e \in P} c_e \geq \pi_{ij} \ \forall \ P \in \mathcal{P}_{ij}, \ \forall \ i, j \in N \tag{4.39}$$

$$\sum_{i \in N, i \neq k} R_i \pi_{ik} + \sum_{j \in N, \neq k} C_j \pi_{kj} \geq 1 \ \forall \ k \in N \tag{4.40}$$

$$w(e) \geq 0 \ \forall \ e \in E \tag{4.41}$$

$$y \geq 0 \tag{4.42}$$

Because of the nature of constraints (4.40), we can assume that the variables π_{ij} attain the maximum possible value given by constraints (4.39) in any optimal solution. Then, we have

$$\pi_{ij} = \min_{P \in \mathcal{P}_{ij}} \left(\sum_{e \in P} w(e) + y \sum_{e \in P} c_e \right)$$

$$= \min_{P \in \mathcal{P}_{ij}} \sum_{e \in P} (w(e) + yc_e) \quad \forall\, i, j \in N$$

This allows us to eliminate the dual variables π_{ij}. The simplified dual problem can be written as:

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

subject to

$$\sum_{i \neq k} R_i \min_{P \in \mathcal{P}_{ik}} \sum_{e \in P} (w(e) + yc_e) + \sum_{j \neq k} C_j \min_{P \in \mathcal{P}_{kj}} \sum_{e \in P} (w(e) + yc_e) \geq 1 \quad \forall\, k \in N \tag{4.43}$$

$$w(e) \geq 0 \quad \forall\, e \in E \tag{4.44}$$

4.4.4 Combinatorial Algorithm

In this section, we use the primal and dual formulations of the path flow based linear program to develop a combinatorial algorithm for the problem.

For a given node k and weights $w(e), y$, let $V(k)$ denote the LHS of constraint (4.43). Given the weights $w(e), y$, note that $\min_{P \in \mathcal{P}_{ij}} \sum_{e \in P} (w(e) + yc_e)$ is the cost of the shortest path from node i to node j under link costs $c(e) = w(e) + yc_e$ for all $e \in E$. Thus, the values $V(k)$ for all $k \in N$ can be computed in polynomial time using a single all-pairs shortest path computation.

The algorithm works as follows. Start with initial weights $y = \frac{\delta}{C}$ and $w(e) = \frac{\delta}{u_e}$ for all $e \in E$ (the quantity δ depends on ϵ and is derived later). Repeat the following until the dual objective function value is greater than or equal to 1:

1. Compute the node k for which $V(k)$ is minimum. This identifies a node $\bar{k}$ as well as paths P_i from node i to node $\bar{k}$ for all $i \neq \bar{k}$ and paths Q_j from node

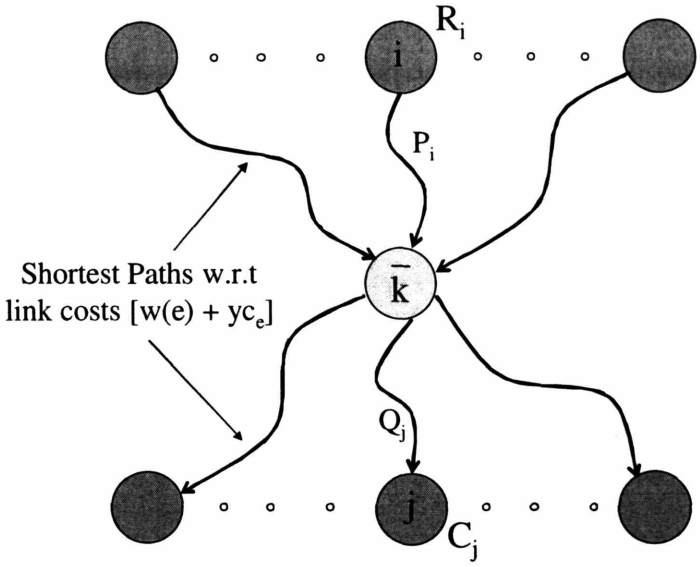


Figure 4-2: One Step in the Primal-Dual Computation for Maximum Throughput Two-Phase Routing with Cost Bound.

$\bar{k}$ to node j for all $j \neq \bar{k}$ (These are the corresponding shortest paths used in evaluating $V(\bar{k})$ as described above.) This is illustrated in Figure 4-2.

2. For a traffic split ratio of 1 for intermediate node $\bar{k}$, the traffic on path P_i is R_i for all $i \neq \bar{k}$ and the traffic on path Q_j is C_j for all $j \neq \bar{k}$. Using this, compute the traffic $f(e)$ on link e per unit split ratio $\alpha_{\bar{k}}$ for intermediate node $\bar{k}$ as

$$f(e) = \sum_{i \neq \bar{k}, P_i \ni e} R_i + \sum_{j \neq \bar{k}, Q_j \ni e} C_j \quad \forall\, e \in E \tag{4.45}$$

3. Compute the maximum value α for the traffic split ratio for intermediate node $\bar{k}$ subject to (i) original link capacity constraints for sending flow along paths P_i, Q_j are obeyed, *and* (ii) cost of the flow is at most C. This is given by

$$\alpha = \min\left(\min_{e \in E} \frac{u_e}{f(e)}, \frac{C}{\sum_{e \in E} c_e f(e)}\right) \tag{4.46}$$

4. For this value α of the traffic split ratio for intermediate node $\bar{k}$, send αR_i amount of flow from node i to node $\bar{k}$ along path P_i for all $i \neq \bar{k}$ and αC_j amount of flow from node $\bar{k}$ to node j along path Q_j for all $j \neq \bar{k}$. Compute the total flow on link e is $\Delta(e) = \alpha f(e)$ for all $e \in E$.

5. Update the weights $w(e)$ as

$$w(e) \leftarrow w(e)\left(1 + \frac{\epsilon\Delta(e)}{u_e}\right) \quad \forall\, e \in E$$

6. Update the weight y as

$$y \leftarrow y\left(1 + \frac{\epsilon \sum_{e \in E} c_e \Delta(e)}{C}\right)$$

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

When the above procedure terminates, link capacity constraints and the total cost constraint in the primal program will be violated, since we were working with the original link capacities and total cost bound at each stage. To remedy this, we scale down the flows traffic split ratios α_i uniformly so that these primal 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.

The pseudo-code for the above procedure, called Algorithm COST-BOUND-MAX-THROUGHPUT, is provided below. Array $flow(e)$ keeps track of the flow sent on link e as the algorithm progresses. The variable D is initialized to 0 and remains less than 1 as long as the dual objective function value is less than 1. After the **while** loop terminates, the factor by which the primal solution needs to be scaled down to

make it feasible is computed into *scale_fact*. Finally, the α_i values are divided by this factor and the resulting values are output.

Algorithm COST-BOUND-MAX-THROUGHPUT:

$\alpha_k \leftarrow 0 \quad \forall\, k \in N$;

$w(e) \leftarrow \frac{\delta}{u_e} \quad \forall\, e \in E$;

$y \leftarrow \frac{\delta}{C}$;

$flow(e) \leftarrow 0 \quad \forall\, e \in E$;

$D \leftarrow 0$;

while $D < 1$ **do**

For each $i, j \in N$, compute shortest path from i to j under link costs $w(e) + yc_e$;

(Denote cost of shortest path from i to j by $SP(i,j)$.)

$V(k) \leftarrow \sum_{i \neq k} R_i SP(i,k) + \sum_{j \neq k} C_j SP(k,j) \quad \forall\, k \in N$;

$\bar{k} \leftarrow \arg\min_{k \in N} V(k)$;

(Denote shortest path from i to $\bar{k}$ by P_i for all $i \neq \bar{k}$

and shortest path from $\bar{k}$ to j by Q_j for all $j \neq \bar{k}$.) ;

$f(e) \leftarrow \sum_{i \neq \bar{k}, P_i \ni e} R_i + \sum_{j \neq \bar{k}, Q_j \ni e} C_j \quad \forall\, e \in E$;

$\alpha \leftarrow \min\left(\min_{e \in E} \frac{u_e}{f(e)}, \frac{C}{\sum_{e \in E} c_e f(e)}\right)$;

$\Delta(e) \leftarrow \alpha f(e) \quad \forall\, e \in E$;

$flow(e) \leftarrow flow(e) + \Delta(e) \quad \forall\, e \in E$;

$w(e) \leftarrow w(e)\left(1 + \frac{\epsilon \Delta(e)}{u_e}\right) \quad \forall\, e \in E$;

$y \leftarrow y\left(1 + \frac{\epsilon \sum_{e \in E} c_e \Delta(e)}{C}\right)$;

$\alpha_{\bar{k}} \leftarrow \alpha_{\bar{k}} + \alpha$;

$D \leftarrow \sum_{e \in E} u_e w(e) + Cy$;

end while

$scale\_fact \leftarrow \max\left(\max_{e \in E} \frac{flow(e)}{u_e}, \frac{\sum_{e \in E} c_e flow(e)}{C}\right)$;

$\alpha_k \leftarrow \frac{\alpha_k}{scale\_fact}$ for all $k \in N$;

Output traffic split ratios α_k for all $k \in N$;

The analysis of the approximation guarantee and running time follows the same approach as that for Algorithm MAX-THROUGHPUT-3 in Section 4.3.5.

Analysis of Approximation Guarantee

Given a set of dual weights $w(e), y$, let $D(w, y)$ denote the dual objective function value and let $\Gamma(w, y)$ denote the minimum value of the LHS of dual program constraint (4.43) over all nodes $k \in N$. Then, solving the dual program is equivalent to finding a set of weights $w(e), y$ such that $\frac{D(w,y)}{\Gamma(w,y)}$ is minimized. Denote the optimal objective function value of the latter by θ, i.e., $\theta = \min_{w,y} \frac{D(w,y)}{\Gamma(w,y)}$. Let w_{t-1} and y_{t-1} denote the weight function w and weight value y respectively at the beginning of iteration t of the **while** loop, and let A_{t-1} be the value of $\sum_{j \in N} \alpha_j$ (primal objective function) up to the end of iteration $t-1$. Suppose the algorithm terminates after iteration L. The following lemma upper bounds the value of $D(w, y)$ at the end of every iteration.

Lemma 4.4.1 *At the end of every iteration t, $1 \leq t \leq L$, of Algorithm COST-BOUND-MAX-THROUGHPUT, the following holds*

$$D(w_t, y_t) \leq (m+1)\delta \prod_{j=1}^{t} [1 + \frac{\epsilon}{\theta}(A_j - A_{j-1})]$$

Proof: During iteration t, let $k = \bar{k}$ be the node for which $V(k)$ is minimum, let P_i, Q_j be the corresponding paths (as defined earlier) along which flow is augmented, and let α be the associated increment in $\alpha_{\bar{k}}$. Recall that the weights are updated as

$$w_t(e) = w_{t-1}(e)\left(1 + \frac{\epsilon \Delta(e)}{u_e}\right) \quad \forall\, e \in E$$

$$y_t = y_{t-1}\left(1 + \frac{\epsilon \sum_{e \in E} c_e \Delta(e)}{C}\right)$$

where $\Delta(e)$ is the total flow sent on link e during iteration t. Using this, we have

$$\begin{aligned}
D(w_t, y_t) &= \sum_{e \in E} u_e w_t(e) + C y_t \\
&= \sum_{e \in E} u_e w_{t-1}(e) + \epsilon \sum_{e \in E} w_{t-1}(e)\Delta(e) + C y_{t-1} + \epsilon y_{t-1} \sum_{e \in E} c_e \Delta(e) \\
&= D(w_{t-1}, y_{t-1}) + \epsilon \sum_{e \in E} (w_{t-1}(e) + y_{t-1} c_e)\Delta(e) \\
&= D(w_{t-1}, y_{t-1}) + \epsilon \sum_{e \in E} (w_{t-1}(e) + y_{t-1} c_e)[\sum_{i \neq \bar{k}, P_i \ni e} \alpha R_i + \sum_{j \neq \bar{k}, Q_j \ni e} \alpha C_j] \\
&= D(w_{t-1}) + \epsilon\alpha \sum_{e \in E} (w_{t-1}(e) + y_{t-1} c_e)[\sum_{i \neq \bar{k}, P_i \ni e} R_i + \sum_{j \neq \bar{k}, Q_j \ni e} C_j]
\end{aligned}$$

Interchanging the summations on the RHS of the above equation and first summing along links on paths P_i, Q_j, and then over i, j respectively, we can rewrite the RHS of the above equation to obtain

$$\begin{aligned}
D(w_t) &= D(w_{t-1}) + \epsilon\alpha[\sum_{i \neq \bar{k}} R_i \sum_{e \in P_i} (w_{t-1}(e) + y_{t-1} c_e) + \sum_{j \neq \bar{k}} C_j \sum_{e \in Q_j} (w_{t-1}(e) + y_{t-1} c_e)] \\
&= D(w_{t-1}) + \epsilon\alpha[\sum_{i \neq \bar{k}} R_i \min_{P \in \mathcal{P}_{i\bar{k}}} \sum_{e \in P} (w_{t-1}(e) + y_{t-1} c_e) + \\
&\quad \sum_{j \neq \bar{k}} C_j \min_{P \in \mathcal{P}_{\bar{k}j}} \sum_{e \in P} (w_{t-1}(e) + y_{t-1} c_e)] \qquad (4.47) \\
&= D(w_{t-1}) + \epsilon\alpha \Gamma(w_{t-1}, y_{t-1}) \\
&= D(w_{t-1}) + \epsilon(A_t - A_{t-1})\Gamma(w_{t-1}, y_{t-1})
\end{aligned}$$

The step leading to (4.47) follows from the fact that P_i, Q_j are shortest paths under link costs $(w_{t-1}(e) + y_{t-1} c_e)$. The next step follows from the choice of node $k = \bar{k}$ for minimizing $V(k)$.

Using this last equation for each iteration down to the first one, we have

$$D(w_t, y_t) = D(w_0, y_0) + \epsilon \sum_{j=1}^{t} (A_j - A_{j-1})\Gamma(w_{j-1}, y_{t-1}) \qquad (4.48)$$

From the definition of θ, we have $\theta \leq \frac{D(w_{j-1},y_{j-1})}{\Gamma(w_{j-1},y_{j-1})}$, whence $\Gamma(w_{j-1},y_{j-1}) \leq \frac{1}{\theta}D(w_{j-1},y_{j-1})$. Also, $D(w_0,y_0) = (m+1)\delta$. Using these in equation (4.48), we have

$$D(w_t,y_t) \leq (m+1)\delta + \frac{\epsilon}{\theta}\sum_{j=1}^{t}(A_j - A_{j-1})D(w_{j-1},y_{j-1}) \tag{4.49}$$

The property claimed in the lemma can now be proved using inequality (4.49) and mathematical induction on the iteration number t. The method is similar to that used in the proof of Lemma 4.3.1. ■

We now estimate the factor by which the objective function value value A_L in the primal solution needs to be scaled when the algorithm terminates so as to ensure that link capacity constraints are not violated.

Lemma 4.4.2 *When Algorithm COST-BOUND-MAX-THROUGHPUT terminates, the primal solution needs to be scaled by a factor of at most* $\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$ *to ensure primal feasibility.*

Proof: We need to show that link capacity constraints (4.35) and cost constraint (4.36) in the primal linear program are obeyed after the primal solution is scaled by the above factor. The link capacity constraints can be shown to be obeyed as in the proof of Lemma 4.3.10. We show here that the cost of the solution is at most C after scaling by the above factor.

Let the cost of the flow sent during iteration t be B_t for $t = 1, 2, \ldots, L$. Let $\sum_{t=1}^{L} B_t = \kappa C$, i.e., the cost of the unscaled solution exceeds the cost bound C by a factor of κ.

Because of the way in which α is chosen in accordance with equations (4.45)-(4.46), the flow sent on each link e during an iteration is at most u_e and the cost of this flow is at most C. Hence, dual weights $w(e), y$ are updated by a factor of at most $1+\epsilon$ after each iteration. Since the algorithm terminates when $D(w,y) \geq 1$, and since dual weights are updated by a factor of at most $1+\epsilon$ after each iteration, we

have $D(w_L, y_L) < 1+\epsilon$. Since the weight y, with coefficient C, is one of the summing components of $D(w, y)$, we have $Cy_L < 1+\epsilon$. Also, the value of y_L is given by

$$y_L = \frac{\delta}{C}\prod_{t=1}^{L}(1+\frac{B_t}{C}\epsilon)$$

Using the inequality $(1+cx) \geq (1+x)^c$ for all $x \geq 0$ and any $0 \leq c \leq 1$ and setting $x = \epsilon$ and $c = \frac{B_t}{C} \leq 1$, we have

$$\begin{aligned}\frac{1+\epsilon}{C} > y_L &\geq \frac{\delta}{C}\prod_{t=1}^{L}(1+\epsilon)^{B_t/C} \\ &= \frac{\delta}{C}(1+\epsilon)^{\sum_{t=1}^{L} B_t/C} \\ &= \frac{\delta}{C}(1+\epsilon)^{\kappa}\end{aligned}$$

whence,

$$\kappa < \log_{1+\epsilon}\frac{1+\epsilon}{\delta}$$

■

The approximation guarantee for Algorithm COST-BOUND-MAX-THROUGHPUT is established in the following theorem. The proof is similar to that for Theorem 4.3.11. The main difference is in the use of the two lemmas established in this section.

Theorem 4.4.3 *For any given $0 < \epsilon' \leq 0.5$, Algorithm COST-BOUND-MAX-THROUGHPUT computes a solution with objective function value within $(1+\epsilon')$-factor of the optimum for*

$$\delta = \frac{1+\epsilon}{[(1+\epsilon)(m+1)]^{1/\epsilon}} \quad \textit{and} \quad \epsilon = \frac{\epsilon'}{2}$$

Analysis of Running Time

We now show that the running time of Algorithm COST-BOUND-MAX-THROUGHPUT is strongly polynomial.

Theorem 4.4.4 *For any given $\epsilon > 0$ chosen to provide the desired approximation factor guarantee in accordance with Theorem 4.3.11, Algorithm COST-BOUND-MAX-THROUGHPUT runs in time*

$$O\left(\frac{1}{\epsilon^2} nm(m + n\log n)\log m\right)$$

which is strongly polynomial.

Proof: The proof is similar to that for Theorem 4.3.12. As before, the running time of each iteration of the algorithm during which a node $\bar{k}$ and associated paths P_i, Q_j are chosen to augment flow is $O(n(m + n\log n))$.

We next estimate the number of iterations before the algorithm terminates. Recall that in each iteration, traffic is sent along primary-backup path pairs P_i, Q_j corresponding to the maximum value of intermediate node split ratio α such that the flow on link e is at most u_e *and* the cost of the flow sent is at most C. Thus, either (or both) of the following is true: (i) the flow sent on some link e is exactly u_e, or (ii) the cost of the flow sent is exactly C. For (i), the weight $w(e)$ increases by a factor of $1 + \epsilon$, and for (ii), the weight y increases by a factor of $1 + \epsilon$. Accordingly, with each iteration, we can associate a dual weight which increases by a factor of $1 + \epsilon$.

As shown in the proof of Theorem 4.3.12, the maximum number of times that the weight $w(e)$ (for a given link e) can be associated with any iteration is $O(\frac{1}{\epsilon}\log_{1+\epsilon} m)$. Using the same reasoning, it follows that the maximum number of times that the weight y can be associated with any iteration is the same quantity.

Since there are a total of $m+1$ weights $w(e), y$, hence the total number of iterations is upper bounded by $O(\frac{1}{\epsilon}(m + 1)\log_{1+\epsilon} m)$. Multiplying this by the running time

per iteration, we obtain the overall algorithm running time as $O(\frac{1}{\epsilon}n(m+1)(m+n\log n)\log_{1+\epsilon} m) = O(\frac{1}{\epsilon^2}nm(m+n\log n)\log m)$. ■

4.5 How Optimal is Two-Phase Routing?

Two-phase routing specifies ratios for splitting traffic among intermediate nodes and Phase 1 and Phase 2 paths for routing them. Thus, two-phase routing is one form of fixed path routing. However, as explained in Section 2.2, it has the desirable property of static provisioning that a general solution of fixed path routing (e.g., direct source-destination path routing) may not have. Moreover, the scheme does not require a packet's final destination to be known as the source, an indirection property that is required of specialized service overlays like i3.

With reference to the above, we would like to investigate the following question: *Do the desirable properties of two-phase routing come with any resource (throughput) overhead compared to (i) direct source-destination path routing, and (ii) optimal scheme among the class of all schemes that are allowed to make the routing dynamically dependent on the traffic matrix?* We address this question in the following ways.

First, using the algorithms developed for two-phase routing in this chapter, we compare the throughput of two-phase routing with that of the optimal scheme on actual ISP topologies in Section 4.6. As we discussed in Section 1.7.3, the throughput of the optimal scheme is *co*$\mathcal{NP}$-hard to compute. In Section 4.5.3, we discuss heuristics to upper bound the throughput of the optimal scheme. This enables us to obtain a posteriori lower bounds on the performance of two-phase routing with respect to the optimal scheme.

Secondly, using the algorithms developed for a generalized version of two-phase routing and for direct source-destination path routing scheme in Chapter 5, we compare the throughput of two-phase routing with that of direct source-destination path

routing on actual ISP topologies in Section 5.5.

Thirdly, we analyze the throughput requirements of two-phase routing from a theoretical perspective and establish a 2-optimal bound in this section. That is, the throughput of two-phase routing is at least $\frac{1}{2}$ that of the best possible scheme in which the *routing can be dependent on the traffic matrix.* We would like to emphasize the generality of this result – it compares two-phase routing with the *most general class of schemes* for routing hose traffic.

4.5.1 Characterization of Optimal Scheme

Consider the class of schemes for routing all matrices in $\mathcal{T}(\vec{R}, \vec{C})$ where the routing can be made dependent on the traffic matrix. For any scheme $\mathcal{A}$, let $A(e, T)$ be the traffic on link e when matrix T is routed by $\mathcal{A}$. Then, the throughput λ_A of scheme $\mathcal{A}$ is given by

$$\lambda_A = \min_{e \in E} \frac{u_e}{\max_{T \in \mathcal{T}(\vec{R}, \vec{C})} A(e, T)}$$

The optimal scheme is the one that achieves the maximum throughput λ_{OPT} among all schemes. This is given by

$$\lambda_{OPT} = \max_{\mathcal{A}} \lambda_A$$

In the following lemma, the throughput of the optimal scheme is expressed in another way. For each $T \in \mathcal{T}(\vec{R}, \vec{C})$, let $\lambda(T)$ be the maximum throughput achievable for routing the single matrix T.

Lemma 4.5.1 *The throughput of the optimal scheme is given by*

$$\lambda_{OPT} = \min_{T \in \mathcal{T}(\vec{R}, \vec{C})} \lambda(T)$$

Proof: For any matrix $T \in \mathcal{T}(\vec{R}, \vec{C})$, since the optimal scheme has to route it, the quantity λ_{OPT} is upper bounded by $\lambda(T)$. Thus, $\lambda_{OPT} \leq \min_{T \in \mathcal{T}(\vec{R}, \vec{C})} \lambda(T)$. This

minimum throughput can indeed be achieved by routing every matrix in a way that maximizes its throughput. Hence equality holds in the above upper bound. ■

At first glance, the optimal scheme that maximizes throughput appears to be hard to specify because it can route each traffic matrix differently, of which there are infinitely many in $\mathcal{T}(\vec{R}, \vec{C})$. However, because the link capacities are given in our throughput maximization model, (an) the optimal scheme can be characterized in a simple way from the proof of Lemma 4.5.1. Given a traffic matrix as input, route it in a manner that maximizes its throughput. Routing a single matrix so as to maximize its throughput is also known as the *maximum concurrent flow problem* [SM90] and is solvable in polynomial time. Clearly, the routing is dependent on the traffic matrix and can be different for different matrices. (An) The optimal scheme for minimum cost network design does not appear to have a simple characterization like the above for maximum throughput network routing.

4.5.2 2-Optimality of Two-Phase Routing

The 2-optimal bound for two-phase routing that we prove next establishes that two-phase routing provides a 2-approximation to the optimal scheme for maximum throughput network routing. Even though this theoretical result shows that the throughput of two-phase routing, in the *worst case*, can be as low as $\frac{1}{2}$ that of the optimal scheme, the experiments in Section 4.6 indicate that two-phase routing performs much better in practice - *for the evaluated topologies, the throughput of two-phase routing is within 6% of that of the optimal scheme and matches that of the optimal scheme in some cases.*

We assume that $R_i = C_i$ for all nodes i. Note that this is not a restrictive assumption in practice because network routers and switches have bidirectional ports (line cards), hence the ingress and egress capacities are equal.

Theorem 4.5.2 *Let $R_i = C_i$ for all nodes i, and $R = \sum_{i \in N} R_i$. Then, the throughput*

of the optimal scheme is at most

$$2\left(1-\frac{1}{R}\min_{i\in N} R_i\right)$$

times that of two-phase routing.

Proof: Let α_i be the traffic split ratios associated with each intermediate node i in two-phase routing. Set $\alpha_i = \frac{R_i}{R}$ for all $i \in N$. Then, the demand matrix $D = [d_{ij}]$ as a result of two phase routing is given by

$$\begin{aligned} d_{ij} &= \alpha_j R_i + \alpha_i C_j \\ &= \alpha_j R_i + \alpha_i R_j \\ &= 2\frac{R_i R_j}{R} \end{aligned}$$

for all $i \neq j$ and $d_{ii} = 0$ for all i.

Now consider the traffic matrix $T = [t_{ij}]$ where

$$t_{ij} = \frac{R_i R_j}{R}$$

for all $i \neq j$ and $t_{ii} = 0$ for all i. Let β be the maximum multiplier such that $\beta T \in \mathcal{T}(\vec{R}, \vec{C})$. Then, we must have

$$\begin{aligned} \beta \sum_{j\in N, j\neq i} t_{ij} &\leq R_i \quad \forall\, i \in N \\ \beta \sum_{j\in N, j\neq i} \frac{R_i R_j}{R} &\leq R_i \quad \forall\, i \in N \\ \beta \frac{R_i(R-R_i)}{R} &\leq R_i \quad \forall\, i \in N \\ \beta &\leq \frac{R}{R-R_i} \quad \forall\, i \in N \end{aligned}$$

whence,

$$\beta = \frac{R}{R - \min_{i \in N} R_i}$$

Since $D = 2T$, hence $\beta T = \frac{\beta}{2} D$. Since the optimal scheme must route the matrix $\beta T \in \mathcal{T}(\vec{R}, \vec{C})$, its throughput λ_{OPT} is at most the throughput $\lambda(\beta T)$ for routing matrix βT (using Lemma 4.5.1). Hence,

$$\lambda_{OPT} \leq \lambda(\beta T) = \lambda(\frac{\beta}{2} D) = \frac{2}{\beta} \lambda(D)$$

The last step uses the property that the throughput of c times a given matrix is equal to $\frac{1}{c}$ times the throughput of the original matrix. Since D is the demand matrix for two-phase routing, we conclude that the throughput of the optimal scheme is at most

$$\frac{2}{\beta} = 2\left(1 - \frac{1}{R} \min_{i \in N} R_i\right)$$

times that of two-phase routing. ■

4.5.3 Upper Bounding Throughput of Optimal Scheme

In this section, we discuss some methods for upper bounding the throughput of the optimal scheme that can possibly reconfigure the routing with changes in the traffic matrix. Because computing the throughput of the optimal scheme is a hard optimization problem (see Section 1.7.3), an upper bound will be useful in comparing the throughput of two-phase routing with that of the optimal scheme.

Lemma 4.5.1 states that $\lambda_{OPT} = \min_{T \in \mathcal{T}(\vec{R}, \vec{C})} \lambda(T)$. Thus, we would a like to identify a matrix $T \in \mathcal{T}(\vec{R}, \vec{C})$ for which $\lambda(T)$ is minimum. We know that this problem is hard. Suppose that we take any single matrix $T \in \mathcal{T}(\vec{R}, \vec{C})$ and compute its throughput $\lambda(T)$ – the maximum throughput for routing a single matrix under given link capacities can be solved using the maximum concurrent flow problem [SM90]. This certainly gives an upper bound on λ_{OPT}, since $\lambda_{OPT} \leq \lambda(T)$. We describe

heuristic approaches to find a matrix $T \in \mathcal{T}(\vec{R}, \vec{C})$ that gives tight upper bounds. We first consider general ingress-egress capacities and then the special case of equal ingress-egress capacities.

General Ingress-Egress Capacities

Consider a matrix T with throughput $\lambda(T)$ whose maximum throughput routing uses x_e capacity on link e. Since $\lambda(T)x_e \leq u_e$ for all e, we have $\sum_{e \in E} \lambda(T)x_e \leq \sum_{e \in E} u_e$, whence

$$\lambda_{OPT} \leq \lambda(T) \leq \frac{\sum_{e \in E} u_e}{\sum_{e \in E} x_e}$$

Let $B(T)$ be the minimum bandwidth required to route matrix T. Then, $\sum_{e \in E} x_e \geq B(T)$, whence $\lambda_{OPT} \leq \frac{\sum_{e \in E} u_e}{B(T)}$. Thus, the least upper bound obtained in this manner is given by

$$\lambda_{OPT} \leq \frac{\sum_{e \in E} u_e}{\max_{T \in \mathcal{T}(\vec{R}, \vec{C})} B(T)} \tag{4.50}$$

The matrix $T \in \mathcal{T}(\vec{R}, \vec{C})$ that takes the highest bandwidth to route can be computed in polynomial time as follows. The minimum bandwidth routing must route all demands *along shortest hop paths.* Let d_{ij} denote the hop count of a shortest path from node i to j for all $i, j \in N$. Then, the problem of determining the traffic matrix $T = [t_{ij}] \in \mathcal{T}(\vec{R}, \vec{C})$ that takes the maximum bandwidth to route can be formulated as the following linear program:

$$\text{maximize} \quad \sum_{i,j \in N} d_{ij} t_{ij}$$

subject to

$$\sum_{j \in N, j \neq i} t_{ij} \leq R_i \quad \forall\, i \in N \tag{4.51}$$

$$\sum_{i \in N, i \neq j} t_{ij} \leq C_j \quad \forall\, j \in N \tag{4.52}$$

$$t_{ij} \geq 0 \ \forall \ i, j \in N \tag{4.53}$$

The required bandwidth $B(T)$ is the objective function of the linear program and the ingress-egress traffic capacities that define $\mathcal{T}(\vec{R}, \vec{C})$ form the constraints.

Let the optimum solution to this linear program be the matrix T^*. The value of $B(T^*) = \max_{T \in \mathcal{T}(\vec{R}.\vec{C})} B(T)$ thus obtained gives us an upper bound on λ_{OPT} using inequality (4.50) above. Note that maximum throughput routing does not necessarily route along shortest paths. Hence, we can actually compute the throughput $\lambda(T^*)$ of the matrix T^* and check if that gives a better (lower) upper bound (since $\lambda_{OPT} \leq \lambda(T^*)$). In all experiments, the latter gave a better upper bound.

In some experiments, the above bound improved when we approximated T^* using a greedy heuristic as follows: Initialize all t_{ij} values to zero. Compute the source-destination pair (i, j) for which $d_{ij} \min(R_i, C_j)$ is maximum. Increment t_{ij} by $t = \min(R_i, C_j)$, decrement both R_i and C_j by t, and recurse.

Equal Ingress-Egress Capacities

Consider the special case when all ingress-egress capacities are equal. Without any loss of generality, we can assume that $R_i = C_i = 1$ for all $i \in N$. We can characterize the structure of the polytope $\mathcal{T}(\vec{R}, \vec{C})$ in this case in a simple manner.

It is well known that the set of non-negative matrices with unit row and column sums (also called doubly stochastic) can be expressed as convex combinations of permutation matrices, i.e., the vertices of the polytope formed by such matrices, also known as the Birkhoff polytope, are precisely the permutation matrices [Z95]. Since the matrices in $\mathcal{T}(\vec{R}, \vec{C})$ have zero diagonal entries, it follows, by an argument similar to that for the Birkhoff polytope, that the vertices of $\mathcal{T}(\vec{R}, \vec{C})$ are precisely the matrices corresponding to *derangement permutations*. (A derangement permutation is one in which no element maps to itself.)

The maximum throughput routing for any matrix T leads to a routing in which all links have utilization of at most $\frac{1}{\lambda(T)}$ (with the upper bound reached by at least one link). Thus, if $\mu(T)$ is the minimum value of the maximum link utilization for routing matrix T, then $\mu(T) = \frac{1}{\lambda(T)}$. We next establish the convex nature of the minimum link utilization function $\mu(T)$.

Lemma 4.5.3 *Consider any two matrices T_1 and T_2. For any scalar $x \in [0,1]$, let $T = xT_1 + (1-x)T_2$. Then,*

$$\mu(T) \leq x\mu(T_1) + (1-x)\mu(T_2)$$

Proof: Let $\mu_1 = \mu(T_1)$ and $\mu_2 = \mu(T_2)$. The matrix xT_1 can be routed with link utilization at most $x\mu_1$. The matrix $(1-x)T_1$ can be routed with link utilization at most $(1-x)\mu_2$. Taking the sum of the routing for the two cases, we obtain a routing for matrix $T = xT_1 + (1-x)T_2$ with link utilization at most $x\mu_1 + (1-x)\mu_2$. Thus, the routing for matrix T that minimizes the maximum link utilization must have a link utilization of at most this quantity. ■

Using Lemma 4.5.1 and the reciprocal relation of throughput and minimum link utilization, we have

$$\lambda_{OPT} = \min_{T \in \mathcal{T}(\vec{R},\vec{C})} \frac{1}{\mu(T)} = \frac{1}{\max_{T \in \mathcal{T}(\vec{R},\vec{C})} \mu(T)}$$

Since the function $\mu(T)$ is convex, its maximum value must occur at one of the vertices of the polytope $\mathcal{T}(\vec{R},\vec{C})$. Thus, it suffices to consider the derangement permutation matrices. We can generate these matrices at random and compute the throughput in an effort to improve the upper bound on λ_{OPT} obtained using the previous methods when the ingress-egress capacities are all equal.

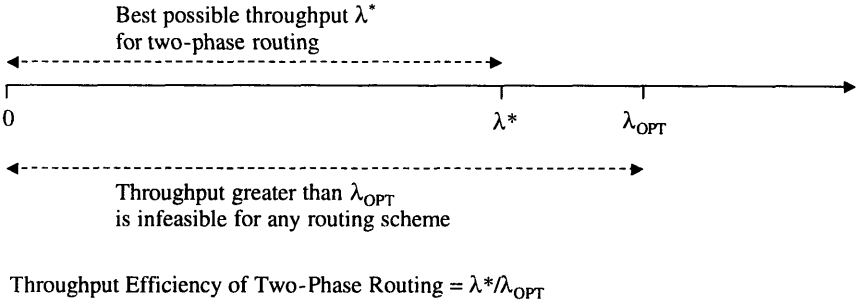


Figure 4-3: Schematic Illustrating Throughput Efficiency of Two-Phase Routing.

4.6 Evaluation on ISP Topologies

In this section, we evaluate the performance of two-phase routing. We first define a quantity called *throughput efficiency* that will be used to measure the effectiveness of two-phase routing with respect to the optimal scheme.

4.6.1 Throughput Efficiency

Given a network with link capacities and bounds R_i, C_j on ingress-egress traffic, an output λ^* of the problem formulation for two-phase routing in Section 4.1 provides a guarantee that all matrices in $\lambda^* \cdot \mathcal{T}(\vec{R}, \vec{C})$ can be routed by two-phase routing. The highest possible throughput λ_{OPT} is admitted by the optimal scheme. This is illustrated in Figure 4-3. We use the ratio $\frac{\lambda^*}{\lambda_{OPT}}$ to define the *throughput efficiency* of two-phase routing.

> **Definition (Throughput Efficiency):** Under given link capacities and ingress-egress bounds on the traffic matrix, the *throughput efficiency* of two-phase routing is given by the quantity $\frac{\lambda^*}{\lambda_{OPT}}$ (≤ 1).

From Theorem 4.5.2, it follows that the throughput efficiency of two-phase routing is at least 0.5 (or, 50%) when the ingress-egress capacities are symmetric, i.e., $R_i = C_i$ for all i. The latter assumption holds for all the ISP topologies we use

in our experiments because network routers and switches have bidirectional ports (line cards). We will see that the throughput efficiency of two-phase routing on the evaluated topologies is significantly better than the theoretical lower bound of 50%.

The value λ_{OPT} was shown to be $co\mathcal{NP}$-hard to compute in Section 1.7.3. We use the methods discussed in Section 4.5.3 to compute a "good" upper bound $\hat{\lambda}_{OPT}$ for λ_{OPT}. Since $\lambda_{OPT} \leq \hat{\lambda}_{OPT}$, we have

$$\frac{\lambda^*}{\hat{\lambda}_{OPT}} \leq \frac{\lambda^*}{\lambda_{OPT}} \leq 1$$

Thus, the quantity $\frac{\lambda^*}{\hat{\lambda}_{OPT}}$ is a lower bound on the throughput efficiency of two-phase routing. We will use this lower bound as a conservative estimate of throughput efficiency of two-phase routing in all the experiments.

4.6.2 Topologies and Link/Ingress-Egress Capacities

We use the six ISP maps from the Rocketfuel dataset which had accompanying (deduced) OSPF/IS-IS weights [SMWH, SMW02, MSWA02]. These topologies list multiple intra-PoP (Point of Presence) routers and/or multiple intra-city PoPs as individual nodes. We coalesced such nodes so that nodes correspond to cities and the topology represents geographical PoP-to-PoP ISP topologies. Some data about the original topologies and their coalesced versions is listed in Table 4.1.

The Rocketfuel topologies are router-level (IP layer) topologies. The PoP-to-PoP topologies we obtained as above all have average node degrees less than 4. Physical WDM topologies of ISPs are characterized by small average node degrees (typically less than 4). Assuming that all physical WDM links appear in the IP topology, it is conceivable that "most" of the links in the PoP-to-PoP Rocketfuel topologies correspond to physical WDM links (instead of multi-hop physical layer paths). We make this assumption for our experiments with IP-over-Optical networks. ISPs regard their topologies as proprietary information – we are not aware of other credible sources

Topology	Routers (original)	Links (inter-router)	PoPs (coalesced)	Links (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 4.1: Rocketfuel topologies with AS number and name. The table lists the original number of routers and inter-router links, and the number of coalesced PoPs and inter-PoP links.

for information about actual ISP topologies.

The topologies provided by Rocketfuel did not include the capacities of the links, which were needed for our study. The Rocketfuel maps did include derived OSPF/ISIS weights of links, which were computed to match observed routes. In the absence of any other information on capacities, we need a way to deduce the link capacities from the weights. For this purpose, we assumed that the given link weights are the Cisco default setting for OSPF weights, i.e., inversely-proportional to the link capacities [Cisco97]. The link capacities obtained in this manner turned out to be symmetric, i.e., $u_{ij} = u_{ji}$ for all $(i, j) \in E$.

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 $R_i (= C_i) \propto \sum_{e \in E^+(i)} u_e$.

Topology	Throughput Efficiency of Two-Phase Routing (Lower Bound)	Throughput Efficiency of Point-to-Point Pipe Model (Lower Bound)
Telstra (Australia) 1221	100%	5.39%
Sprintlink (US) 1239	97.71%	3.76%
Ebone (Europe) 1755	98.90%	7.33%
Tiscali (Europe) 3257	95.65%	5.97%
Exodus (Europe) 3967	100%	13.15%
Abovenet (US) 6461	94.82%	10.44%

Table 4.2: Throughput Efficiency (lower bound) of Two-Phase Routing and Point-to-Point Pipe model.

4.6.3 Experiments and Results

To obtain the maximum throughput for two-phase routing for purposes of comparison with that of the optimal scheme, we used the exact linear programming formulation from Section 4.1. The linear program size was reduced by a factor of n by using per source flow variables instead of per source-destination flow variables.

Throughput Efficiency

In Table 4.2, we give the throughput efficiency of two-phase routing for the six Rocketfuel topologies. We compare this with the throughput efficiency of the *point-to-point pipe provisioning model* in which a fixed demand of $\min(R_i, C_j)$ is provisioned from node i to node j for all $i, j \in N$ to handle the maximum possible traffic from i and j under the given ingress-egress capacities. Similar to that for two-phase routing, the throughput efficiency of the point-to-point pipe model is measured relative to the throughput of the optimal scheme.

Table 4.2 clearly shows that the throughput of two-phase routing is very close to that of the best possible scheme for routing with traffic variability on all six Rocketfuel topologies. Thus, two-phase routing, surprisingly, is able to meet the requirements of Section 1.5 without any appreciable decrease in throughput compared to the optimal scheme. Table 4.2 also brings out the poor throughput performance of the point-to-

Topology	Number of Intermediate Nodes
Telstra (Australia) 1221	1
Sprintlink (US) 1239	5
Ebone (Europe) 1755	4
Tiscali (Europe) 3257	7
Exodus (Europe) 3967	3
Abovenet (US) 6461	7

Table 4.3: Number of Intermediate nodes in Two-Phase Routing.

point pipe model, the throughput efficiency of which is in the range of 3-14%.

Number of Intermediate Nodes

In Table 4.3, we list the number of intermediate nodes i with $\alpha_i > 0$ for maximum throughput two-phase routing on the six Rocketfuel topologies. Interestingly, the number of such intermediate nodes, especially for the larger topologies, is small compared to the total number of nodes. This may have favorable implications in the adaptation of the scheme to specialized service overlays and middlebox routing as explained in Section 2.2. In these two application scenarios, the intermediate nodes are sites for locating overlay routing servers and middleboxes respectively.

As explained in Section 2.2.1, a small number of intermediate nodes (compared to the total number of network nodes) is desirable for IP-over-Optical networks also, since that makes the number of IP layer links linear in the number of network nodes – this leads to a more scalable network architecture from an ISP deployment perspective.

Equal vs. Unequal Traffic Split Ratios

For the two-phase routing scheme, we denote the throughput for equal traffic split ratios by λ_{equal} and the throughput for our general problem formulation that allows unequal traffic split ratios by $\lambda_{unequal}$. It is easy to see that $\lambda_{unequal} \geq \lambda_{equal}$. In

Topology	$\lambda_{unequal}$	λ_{equal}	$\frac{\lambda_{unequal}-\lambda_{equal}}{\lambda_{equal}}$
Telstra (Australia) 1221	1.0	0.7756	28.93%
Sprintlink (US) 1239	1.0	0.3978	151.38%
Ebone (Europe) 1755	1.0	0.6137	62.95%
Tiscali (Europe) 3257	1.0	0.6625	50.95%
Exodus (Europe) 3967	1.0	0.8908	12.26%
Abovenet (US) 6461	1.0	0.7098	40.89%

Table 4.4: Throughput of Two-Phase Routing with unequal and equal traffic split ratios.

Table 4.4, we give the throughput of two-phase routing with equal and unequal split ratios. The percentage increase in throughput $\frac{\lambda_{unequal}-\lambda_{equal}}{\lambda_{equal}}$ when we go from equal to unequal split ratios is also shown. 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 unequal traffic split ratios case is $\lambda_{unequal} = 1.0$.

The results clearly bring out the increase in network throughput when the split ratios α_i are allowed to be unequal. The average savings for the six Rocketfuel topologies is 57.89% and the range is from 12% to as high as 152%. We conclude that by allowing the traffic split ratios to be unequal, network throughput for two-phase routing can be increased significantly over the equal traffic split ratios case.

Router-to-OXC Link Capacity Constraints in IP-over-Optical Networks

We investigate the effect of router-to-OXC link capacity constraints in IP-over-Optical networks on the throughput and number of intermediate nodes for the two-phase routing scheme. For this purpose, we set the router-to-OXC link capacity at every node to be a fixed fraction of the total capacity of all network links incident at that node and vary this fraction for the experiments, starting from 0.05 and increasing to 1.0.

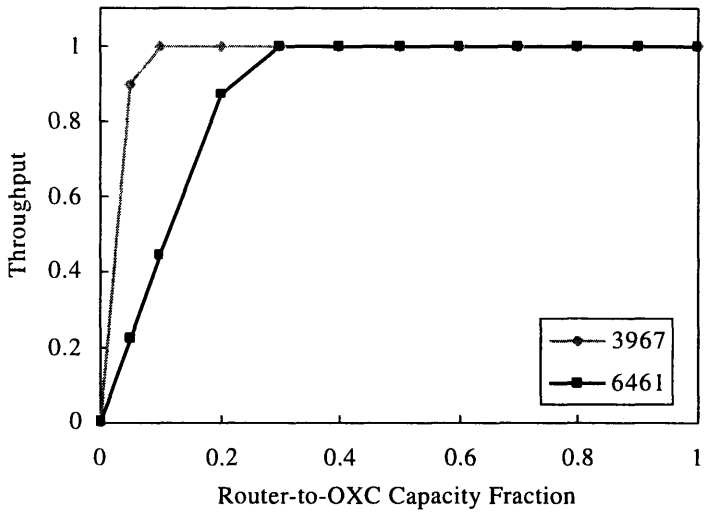


Figure 4-4: Throughput for Two-Phase Routing in IP-over-Optical networks with varying router-to-OXC capacity. Rocketfuel topology number is indicated in legend.

The throughput values for two Rocketfuel topologies, Exodus 3967 and Abovenet 6461, are plotted in Figure 4-4. As before, we have normalized the throughput values so that the throughput when the router-to-OXC link capacities are sufficiently large is 1.0. (When the router-to-OXC link capacities are sufficiently large, the throughput is determined by the capacities of the network links.) In Figure 4-5, we plot the number of intermediate nodes i with $\alpha_i > 0$.

Because intermediate node processing requires packets to traverse router-to-OXC links at that node, it can be expected that limiting this capacity will lead to traffic being split across more intermediate nodes. This is consistent with our experimental observation. As the router-to-OXC link capacity fraction is increased, the throughput increases and ultimately flattens to the (normalized) value of 1.0 - this happens at router-to-OXC link capacity fraction values of 0.1 and 0.3 for the Exodus 3967 and Abovenet 6461 topologies respectively. Similarly, the number of intermediate nodes decreases and ultimately flattens to the values consistent with Table 4.3 for the corresponding topologies - this happens at router-to-OXC link capacity fraction

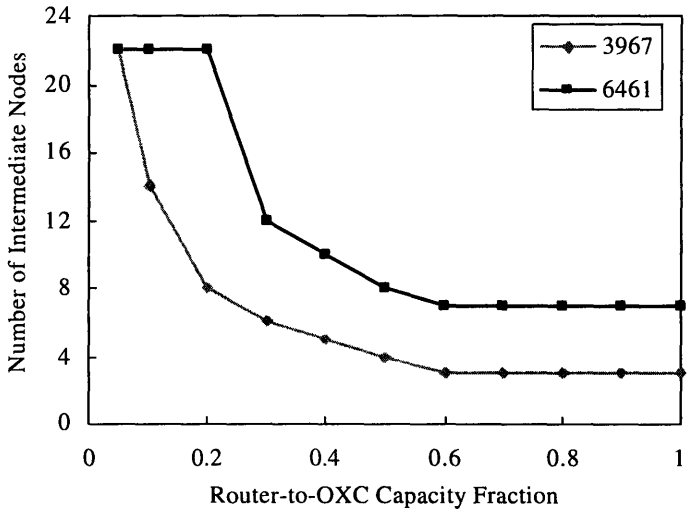


Figure 4-5: Number of Intermediate Nodes in Two-Phase Routing in IP-over-Optical networks with varying router-to-OXC capacity. Rocketfuel topology number is indicated in legend.

value of 0.6 for both the topologies.

Chapter 5

Generalized Traffic Split Ratios

In our description and problem formulation for two-phase routing so far in this thesis, the traffic split ratios α_i are dependent on the respective intermediate nodes only. In this chapter, we generalize the traffic split ratios to depend on source and destination of traffic. This is conceivably the most general form of two-phase routing. It is motivated by the fact that source nodes should not be required to split traffic through intermediate nodes that are distant from them or the destination nodes of traffic. This generalization has the potential of reducing network cost or increasing network throughput.

We first derive an expression for the demand between any two nodes as a result of two-phase routing with generalized traffic split ratios. For each of the optimization models of minimum cost network design and maximum throughput network routing, we first provide a linear programming formulation with an infinite number of constraints and a polynomial time separation oracle (another linear program) that is suitable for solution using the ellipsoid method. By taking the dual of the separation oracle and combining it with the main linear program, we reduce the number of constraints to polynomial size, thus significantly reducing the running time. For the minimum cost network design problem, we use an upper bound on the demand values to obtain a simplified linear programming formulation that can be interpreted

as a fraction Steiner forest problem (this problem admits combinatorial algorithms).

Using a technique similar to that used for two-phase routing, we give a polynomial size linear programming formulation for maximizing throughput for direct source-destination routing of variable traffic along fixed paths. Using this, we compare the throughput requirement of two-phase routing (with generalized traffic split ratios) with that of direct source-destination path routing on actual ISP topologies collected for the Rocketfuel project [SMWH].

5.1 Generalized Traffic Split Ratios and Demand Values

The traffic split ratios α_i can be generalized to depend on *source or destination nodes* of the traffic, or both. We consider the latter version here. While this generalization does not meet the indirection requirement of specialized service overlays like i3, it can potentially decrease the resource requirements of the two-phase routing scheme for other application scenarios like IP-over-Optical networks.

Suppose that a fraction α_k^{ij} of the traffic that originates at node i whose destination is node j is routed to node k in Phase 1. The traffic split ratios associated with any source-destination pair must sum to unity, i.e., $\sum_{k \in N} \alpha_k^{ij} = 1$ for all $i, j \in N$. Let us compute the total demand that is needed between nodes a and b to route Phase 1 and Phase 2 paths. Let the current traffic matrix be $T = [t_{ij}] \in \mathcal{T}(\vec{R}, \vec{C})$. In the first phase, a fraction α_b^{ak} of the traffic t_{ak} originating at node a and destined for node k is sent to intermediate node b. Thus, the demand from node a to node b for Phase 1 traffic is $\sum_{k \in N} \alpha_b^{ak} t_{ak}$. A fraction α_a^{kb} of the traffic t_{kb} originating at node k and destined for node b is sent to intermediate node a in Phase 1 and needs to be routed to node b in the second phase. Thus, the demand from node a to node b for Phase 2 traffic is $\sum_{k \in N} \alpha_a^{kb} t_{kb}$. Therefore, the total demand τ_{ab} that needs to be statically provisioned from node a to node b is the maximum value, taken over all

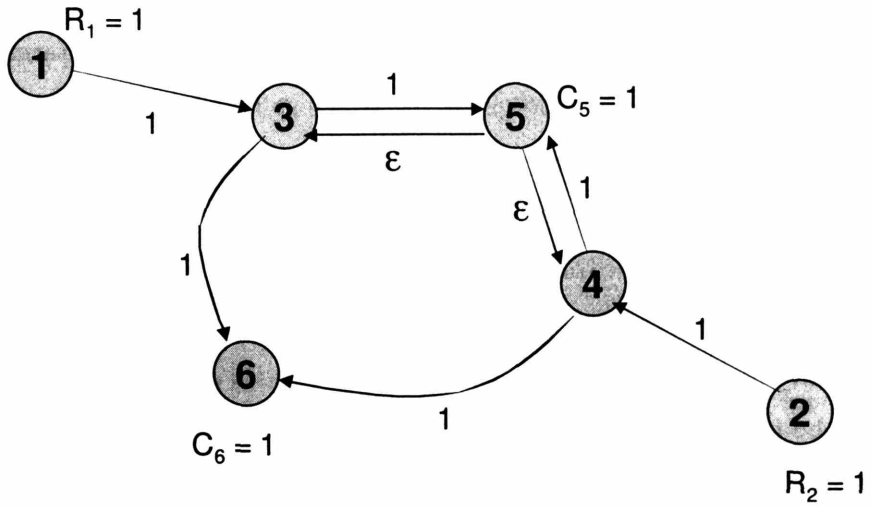


Figure 5-1: 6-node network to illustrate throughput improvement with generalized traffic split ratios for Two-Phase Routing.

traffic matrices $T \in \mathcal{T}(\vec{R}, \vec{C})$, of the sum of the above two quantities, that is,

$$\tau_{ab} = \max_{[t_{ij}] \in \mathcal{T}(\vec{R},\vec{C})} \left[\sum_{k \in N} \alpha_b^{ak} t_{ak} + \sum_{k \in N} \alpha_a^{kb} t_{kb} \right] \tag{5.1}$$

The quantity above appears to involve bilinear terms but can be nicely accommodated into a linear program for the problems of both minimum cost network design and maximum throughput network routing as we show in the next two sections.

Before proceeding further, we give an example to illustrate the improvement in throughput when we generalize the traffic split ratios as above. Consider the 6-node network shown in Figure 5-1. Here $R_1 = R_2 = 1$ and $C_5 = C_6 = 1$. All other R_i, C_j values are zero. The capacities of links $(5, 3)$ and $(5, 4)$ are each equal to some small quantity $\epsilon > 0$. All other links shown have unit capacity.

Observe that node 1 has a unit capacity path to node 3 but the capacity of the path to node 4 is small $(= \epsilon)$. Similarly, node 2 has a unit capacity path to node 4 but the capacity of the path to node 3 is small $(= \epsilon)$. Thus, when maximizing

throughput, node 4 is not a good choice for serving as intermediate node for the traffic originating at node 1. Similarly, node 3 is not a good choice for serving as intermediate node for the traffic originating at node 2. If the traffic split ratios are dependent on intermediate nodes *only* (and not on source or destination of traffic), the throughput of two-phase routing will be small. By making the traffic split ratios dependent on the source of traffic also, two-phase routing can completely avoid routing along the links with small capacities. In fact, the gap between the throughputs of two-phase routing with intermediate node dependent traffic split ratios α_k and generalized traffic split ratios α_k^{ij} can be made arbitrarily large by making the value of ϵ arbitrarily small.

For network cost, a similar example where the links $(5,3)$ and $(5,4)$ have large costs can be used to show that the gap between the costs of two-phase routing with intermediate node dependent traffic split ratios and generalized traffic split ratios can be arbitrarily large.

However, in view of the 2-optimality results for two-phase routing with respect to both network cost and network throughput (in Sections 3.2 and 4.5) that use only intermediate node dependent traffic split ratios and assume $R_i = C_i$ for all i, it follows that such pathological examples where the cost or throughput improvement with generalized split ratios is arbitrarily large (or, even greater than 2) do not exist when ingress-egress capacities are symmetric.

5.2 Minimum Cost Network Design for Two-Phase Routing

Given a network with link costs c_e and constraints R_i, C_j on the ingress-egress traffic, we consider the problem of two-phase routing with generalized traffic split ratios so as to minimize the network cost.

We begin with a linear programming formulation with an infinite number of constraints and a polynomial size separation oracle linear program for it, and then com-

bine the two into a polynomial size linear program that can be solved in polynomial time using a general linear programming algorithm [S86].

5.2.1 LP with Infinite Constraints and Separation Oracle

The routing of τ_{ab} amount of traffic, as given by equation (5.1). for each source-destination pair (a, b) must be along the shortest cost path in an optimal solution. Let d_{ij} be the cost of the shortest path from node i to node j under link costs c_e. Then, the network cost is given by $\sum_{i,j \in N} d_{ij}\tau_{ij}$. The linear programming for minimizing network cost is as follows:

$$\text{minimize} \quad \sum_{i,j \in N} d_{ij}\tau_{ij}$$

subject to

$$\tau_{ab} \geq \sum_{k \in N} \alpha_b^{ak} t_{ak} + \sum_{k \in N} \alpha_a^{kb} t_{kb} \quad \forall\, [t_{ij}] \in \mathcal{T}(\vec{R}, \vec{C}), \quad \forall\, a, b \in N \tag{5.2}$$

$$\sum_{k \in N} \alpha_k^{ab} = 1 \quad \forall\, a, b \in N \tag{5.3}$$

$$\alpha_k^{ab} \geq 0 \quad \forall\, k, a, b \in N \tag{5.4}$$

Constraints (5.2) correspond to the value of the demand τ_{ab} from node a to node b as given in equation (5.1). Constraints (5.3) correspond to the traffic split ratios summing to 1 for each source-destination pair. The quantities t_{ij} in the RHS of (5.2) are constants and hence the constraints are linear. Note that there are an infinite set of constraints in (5.2), since there are $n(n-1)$ constraints for each $[t_{ij}] \in \mathcal{T}(\vec{R}, \vec{C})$.

The above linear program can be solved in polynomial time by the ellipsoid algorithm [S86] provided we can find a polynomial time separation oracle for the constraints (5.2). Given a set of values for the variables in the above linear program, the

separation oracle needs to identify at least one constraint that is violated (if any), or indicate otherwise. Clearly, constraint (5.3) can be verified in polynomial time.

To determine if the constraints in (5.3) are violated for any link, we need to either identify a source-destination pair (a, b) and a traffic matrix $[t_{ij}] \in \mathcal{T}(\vec{R}, \vec{C})$ such that the corresponding constraint is violated, or determine that all such constraints are satisfied. This can be done by verifying that for each source-destination pair (a, b), the linear program below, with variables t_{ij} for all $i, j \in N$, has optimum objective function value at most τ_{ab}. If not, the traffic matrix $[t_{ij}]$ obtained in the optimal solution of the linear program identifies the corresponding violating constraint in (5.2).

$$\text{maximize} \quad \sum_{k \in N} \alpha_b^{ak} t_{ak} + \sum_{k \in N} \alpha_a^{kb} t_{kb}$$

subject to

$$\sum_{j \in N, j \neq i} t_{ij} \leq R_i \quad \forall\, i \in N \tag{5.5}$$

$$\sum_{i \in N, i \neq j} t_{ij} \leq C_j \quad \forall\, j \in N \tag{5.6}$$

$$t_{ij} \geq 0 \quad \forall\, i, j \in N \tag{5.7}$$

The ellipsoid method is primarily a theoretical tool for proving polynomial-time solvability – its running time is not feasible for practical implementations. Hence, the motivation for designing a polynomial size linear program for the above problem. Such an LP can be directly fed into LP solvers like CPLEX [CPLEX] for solution.

5.2.2 Polynomial Size LP

In developing the polynomial size LP, we first take the dual of the separation oracle linear program above. For a given source-destination pair (a, b), the dual linear

program has non-negative variables $r(i,a,b)$ corresponding to each constraint in (5.5) and non-negative variables $c(j,a,b)$ corresponding to each constraint in (5.6).

$$\text{minimize} \quad \sum_{i \in N} R_i r(i,a,b) + \sum_{j \in N} C_j c(j,a,b)$$

subject to

$$\begin{aligned}
r(a,a,b) + c(b,a,b) &\geq \alpha_a^{ab} + \alpha_b^{ab} && (5.8)\\
r(a,a,b) + c(k,a,b) &\geq \alpha_b^{ak} \quad \forall\, k \in N, k \neq b && (5.9)\\
r(k,a,b) + c(b,a,b) &\geq \alpha_a^{kb} \quad \forall\, k \in N, k \neq a && (5.10)\\
r(i,a,b), c(i,a,b) &\geq 0 \quad \forall\, i \in N && (5.11)
\end{aligned}$$

It follows directly from strong duality of linear programming [S86] that for each source-destination pair (a,b), the primal (separation oracle) linear program has an optimum objective function value of at most τ_{ab} if and only if the dual linear program has a feasible solution with objective function value at most τ_{ab}. The requirement that the dual linear programs, for all $a,b \in N$, have feasible solutions with objective function value at most τ_{ab} can be modeled as the following constraint:

$$\sum_{i \in N} R_i r(i,a,b) + \sum_{j \in N} C_j c(j,a,b) \leq \tau_{ab} \quad \forall\, a,b \in N$$

This allows us to remove the infinite set of constraints in (5.2) and add the above constraint and constraints (5.8)-(5.11) from the dual linear programs to obtain the following polynomial size linear program for our problem:

$$\text{minimize} \quad \sum_{i,j \in N} d_{ij} \tau_{ij}$$

subject to

$$\sum_{k \in N} \alpha_k^{ab} = 1 \quad \forall\, a, b \in N \tag{5.12}$$

$$\sum_{i \in N} R_i r(i,a,b) + \sum_{j \in N} C_j c(j,a,b) \leq \tau_{ab} \quad \forall\, a, b \in N \tag{5.13}$$

$$r(a,a,b) + c(b,a,b) \geq \alpha_a^{ab} + \alpha_b^{ab} \quad \forall\, a, b \in N \tag{5.14}$$

$$r(a,a,b) + c(k,a,b) \geq \alpha_b^{ak} \quad \forall\, k, a, b \in N, k \neq b \tag{5.15}$$

$$r(k,a,b) + c(b,a,b) \geq \alpha_a^{kb} \quad \forall\, k, a, b \in N, k \neq a \tag{5.16}$$

$$\alpha_k^{ab} \geq 0 \quad \forall\, k, a, b \in N \tag{5.17}$$

$$r(i,a,b), c(i,a,b) \geq 0 \quad \forall\, i, a, b \in N \tag{5.18}$$

This linear program has $n(n-1)$ constraints each in (5.12)-(5.14), $n(n-1)(n-2)$ constraints each in (5.15)-(5.16), $n^2(n-1)$ constraints in (5.17), and $2n^2(n-1)$ constraints in (5.18), for a total of $O(n^3)$ constraints. The number of variables is $n(n-1) + n^2(n-1) + 2n^2(n-1) = O(n^3)$.

5.2.3 Simplification Using an Upper Bound on the Demands

In this section, we derive an upper bound on the demand from node a to node b for two-phase routing with generalized traffic split ratios, as given by equation (5.1). We then use this upper bound as the demand value and obtain two different formulations for the problem that can give faster running times. The first is a linear programming formulation with about $2n^3$ fewer variables. The second is a fractional Steiner forest formulation that admits a combinatorial algorithm. It should be noted that the only approximation involved in the problem formulations is in the use of the derived upper bound on the demand values as the actual demand values – the formulations are exact for these demand values.

For a traffic matrix $T = [t_{ij}] \in \mathcal{T}(\vec{R}, \vec{C})$, the demand from node a to node b for

Phase 1 traffic is

$$\sum_{k \in N} \alpha_b^{ak} t_{ak} \leq \max_{k \in N} \alpha_b^{ak} \sum_{k \in N} t_{ak} \leq \max_{k \in N} \alpha_b^{ak} R_a$$

and the demand from node a to node b for Phase 2 traffic is

$$\sum_{k' \in N} \alpha_a^{k'b} t_{k'b} \leq \max_{k' \in N} \alpha_a^{k'b} \sum_{k' \in N} t_{k'b} \leq \max_{k' \in N} \alpha_a^{k'b} C_b$$

Using the above two inequalities, the upper bound on the total demand τ_{ab} from node a to node b as a result of two-phase routing is

$$\tau_{ab} \leq \max_{k \in N} \alpha_b^{ak} R_a + \max_{k' \in N} \alpha_a^{k'b} C_b \quad \forall\, a, b \in N \tag{5.19}$$

Let τ'_{ab} denote the upper bound on τ_{ab} given by the RHS of (5.19). We will use this upper bound τ'_{ab} instead of the exact value τ_{ab} given by (5.1).

$$\tau'_{ab} = R_a \max_{k \in N} \alpha_b^{ak} + C_b \max_{k' \in N} \alpha_a^{k'b} \quad \forall\, a, b \in N \tag{5.20}$$

Linear Programming Formulation

The quantity τ'_{ab} is equal to the minimum value that satisfies the following linear constraints:

$$\tau'_{ab} \geq R_a \alpha_b^{ak} + C_b \alpha_a^{k'b} \quad \forall\, k, k' \in N$$

We replace the infinite set of constraints (5.2) in the linear programming formulation in Section 5.2.1 to obtain the following polynomial size linear program:

$$\text{minimize} \quad \textstyle\sum_{i,j \in N} d_{ij} \tau'_{ij}$$

subject to

$$\tau'_{ab} \geq R_a \alpha_b^{ak} + C_b \alpha_a^{k'b} \quad \forall\, k, k', a, b \in N \tag{5.21}$$

$$\sum_{k \in N} \alpha_k^{ab} = 1 \quad \forall\, a, b \in N \tag{5.22}$$

$$\alpha_k^{ab} \geq 0 \quad \forall\, k, a, b \in N \tag{5.23}$$

Compared to the polynomial size linear program in Section 5.2.2, the above linear program has $2n^2(n-1)$ fewer variables. However, the number of constraints increases from $O(n^3)$ to $O(n^4)$ because of the $n^3(n-1)$ constraints in (5.21). The above linear program has a much simpler structure than the one in Section 5.2.2 and can be interpreted in terms of fractional solutions to a well-studied problem in combinatorial optimization. We discuss this next.

Fractional Steiner Forest Formulation

We show that the linear programming formulation that we just developed is equivalent to the fractional Steiner forest problem with certain restrictions on the choice of paths. In the Steiner forest problem problem, we are given a graph with non-negative edge costs and a set of source-destination node pairs. The objective is to select a minimum cost set of edges such that the induced subgraph contains a path connecting each source-destination pair. For the undirected version of the problem, the subgraph formed by the selected edges in an optimal solution is a forest, since otherwise we can remove a link from a cycle without increasing the cost of the solution. However, this is not true in general for the directed version of the problem. Because we use a directed graph in our problem for two-phase routing, we will be concerned with the directed version of the Steiner forest problem.

The fractional version of the Steiner forest problem can be obtained by considering it in terms of network flow as follows – the path connecting a source-destination pair corresponds to sending integer flow of unit value and the usage on a link is the *maximum flow* it carries for any source-destination path (which, in the integer version, is 0 or 1). The fractional relaxation is obtained by allow the unit flow between every

source-destination pair to be *splittable*. The total cost is obtained by multiplying the cost of a link by its usage and summing over all links. Thus, the main difference from the standard multicommodity flow problem is that the flow on a link is the maximum (and, not the sum) of the usage associated with all source-destination pairs.

Coming back to our minimum cost network design problem for two-phase routing, recall that the Phase 1 and Phase 2 paths are routed along shortest cost paths in an optimal solution. Hence, it is helpful to work on a complete graph whose links represent shortest paths in the given topology graph G. Accordingly, let $G' = (N, E')$ be the complete graph on n nodes where the link (i, j) corresponds to a shortest path (of cost d_{ij}) from node i to node j. (Recall that d_{ij} is the cost of the shortest path from node i to node j under link costs c_e.)

For Phase 1 paths, the capacity that needs to be allocated on the logical link (i, j) in G' is $R_i \max_{k \in N} \alpha_j^{ik}$, incurring a cost of $d_{ij} R_i \max_{k \in N} \alpha_j^{ik}$. Likewise, for Phase 2 paths, the capacity that needs to be allocated on the logical link (i, j) in G' is $C_j \max_{k \in N} \alpha_i^{kj}$, incurring a cost of $d_{ij} C_j \max_{k \in N} \alpha_i^{kj}$.

To develop the Steiner forest formulation, we actually need to make two copies of the each link (i, j) in G'. Denote the resulting graph by $G'' = (N, E'')$. The first copy of a link is used for carrying Phase 1 traffic, and the second copy is used for carrying Phase 2 traffic. Thus, in G'', each link carries either Phase 1 traffic or Phase 2 traffic but not both.

We now interpret the traffic split ratios α_k^{ij} as flow variables for sending unit amount of flow from node i to node j in graph G''. This flow is split across paths of (at most) two hops that pass through any intermediate node k. The flow on the path $i - k - j$ is α_k^{ij}. The first hop corresponds to Phase 1 traffic and the second hop corresponds to Phase 2 traffic. This is illustrated in Figure 5-2. When $k = i$, this is a one hop path corresponding to Phase 2 traffic. When $k = j$, this is a two hop path corresponding to Phase 1 traffic. For all other values of k, this is a two hop path. The total flow sent from node i to node j is unity since $\sum_{k \in N} \alpha_k^{ij} = 1$.

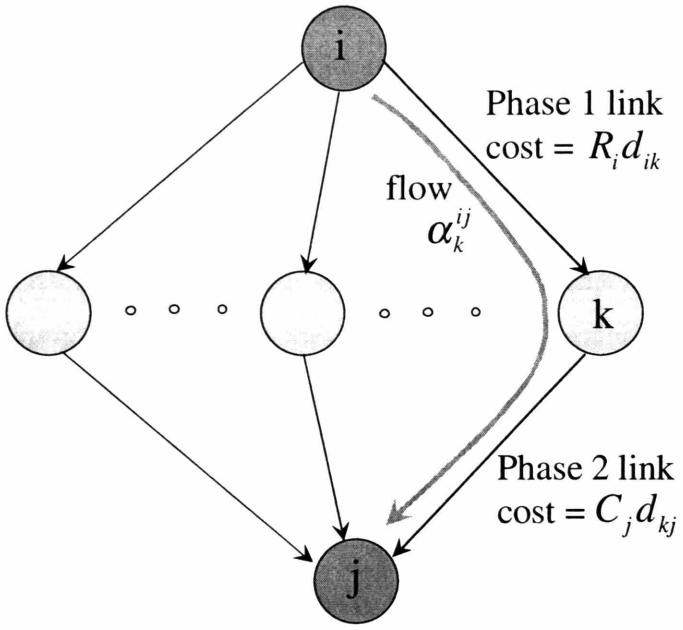


Figure 5-2: Fractional Steiner forest formulation for minimum cost Two-Phase Routing with generalized traffic split ratios.

Using this interpretation of flow variables, the *maximum flow for any source-destination pair* on the first copy of link (i, j) (for Phase 1 traffic) in G'' is $\max_{k \in N} \alpha_j^{ik}$. Thus, the value of Phase 1 traffic on a link is consistent with the capacity allocated on a link in the fractional Steiner forest problem. By making the cost of the first copy of link (i, j) (for Phase 1 traffic) equal to $d_{ij} R_i$, we obtain the cost associated with Phase 1 traffic on this link as $(d_{ij} R_i) \max_{k \in N} \alpha_j^{ik}$. Similarly, the flow on the second copy of link (i, j) (for Phase 2 traffic) is $\max_{k \in N} \alpha_i^{kj}$. By making the cost of the second copy of link (i, j) (for Phase 2 traffic) equal to $d_{ij} C_j$, we obtain the cost associated with Phase 2 traffic on this link as $(d_{ij} C_j) \max_{k \in N} \alpha_i^{kj}$. The link costs are shown in Figure 5-2. Thus, the cost of the solution for the fractional Steiner forest formulation is equal to

$$\sum_{i,j \in N} (d_{ij} R_i) \max_{k \in N} \alpha_j^{ik} + \sum_{i,j \in N} (d_{ij} C_j) \max_{k \in N} \alpha_i^{kj} \quad = \quad \sum_{i,j \in N} d_{ij} \left(R_i \max_{k \in N} \alpha_j^{ik} + C_j \max_{k \in N} \alpha_i^{kj} \right)$$

$$= \sum_{i,j \in N} d_{ij} \tau'_{ij} \quad \text{(using (5.20))}$$

which is equal to the objective function of our linear programming problem formulation using the upper bound τ'_{ij} on the demand values. This establishes the fractional Steiner forest formulation for our problem. Note that in our formulation, there are exactly n paths along which flow can be (splittably) routed for each source-destination pair (two of these paths have a single hop and the remaining have two hops).

Garg and Khandekar [GK02] give a fast combinatorial algorithm for computing a solution to the fractional Steiner forest problem up to $(1+\epsilon)$-factor of the optimum for any given $\epsilon > 0$. They develop a primal-dual scheme using a *path indexed* linear programming formulation for the problem. Hence, their formulation allows the set of paths for each source-destination pair to be restricted, which meets the requirement of our formulation. On an instance of our problem, where the graph G'' has n nodes and $2n(n-1)$ links and a set of n available paths (of at most two hops) for each source-destination pair, their combinatorial algorithm runs in $O(\frac{1}{\epsilon^2} n^5 \log^2 n)$ time. Since our linear programming formulation has $O(n^3)$ variables and $O(n^4)$ constraints, the combinatorial algorithm is asymptotically faster than a general linear programming algorithm for the problem.

5.2.4 Note about Undirected Graphs with Symmetric Ingress-Egress Capacities

In Section 3.3, we showed that for undirected graphs and symmetric ingress-egress capacities, the optimum solution for minimum cost two-phase routing with α_i split ratios is identical in structure and cost to that for the minimum cost tree solution for direct source-destination path routing. It is conjectured that for direct source-destination path routing along fixed paths in undirected graphs, the cost of the optimum tree solution is identical to that for the optimum multi-path routing (fractional) solution when ingress-egress capacities are symmetric. Experimental evidence for this is re-

ported in [ER04]. Recently, the conjecture was shown to be true when the graph is a cycle [HKS05]. If the conjecture is true for general graphs, then it follows that for undirected graphs and symmetric ingress-egress capacities, the cost of optimal two-phase routing with α_i split ratios, as obtained in Section 3.1, is identical to that for optimum multi-path routing when ingress-egress capacities are symmetric – in which case it suffices to use α_i split ratios for two-phase routing (using generalized split ratios α_k^{ij} does not help in reducing cost).

5.3 Maximum Throughput Two-Phase Routing

Given a network with link capacities u_e and constraints R_i, C_j on the ingress-egress traffic, we consider the problem of two-phase routing with generalized traffic split ratios so as to maximize network throughput. The throughput is the maximum multiplier λ such that all matrices in $\lambda \cdot \mathcal{T}(\vec{R}, \vec{C})$ can be routed under given link capacities.

We use the same approach as for minimum cost network design in Section 5.2 to obtain a polynomial size linear program for the problem. We begin with a linear programming formulation with an infinite number of constraints and a polynomial size separation oracle linear program for it, and then combine the two into a polynomial size linear program.

5.3.1 LP with Infinite Constraints and Separation Oracle

The routing of τ_{ab} amount of traffic, as given by equation (5.1), for each source-destination pair (a, b) can be specified by a set of flow variables x_e^{ab}, where x_e^{ab} denotes the amount of traffic from node a to node b that traverses link e in the network. Let μ denote the maximum utilization of any link in the network. Then, all matrices in $\frac{1}{\mu}\mathcal{T}(\vec{R}, \vec{C})$ can also be feasibly routed. Thus, the throughput λ is the reciprocal of the maximum link utilization μ – maximizing λ is equivalent to minimizing μ. The

linear programming formulations is as follows:

$$\text{minimize } \mu$$

subject to

$$\tau_{ab} \geq \sum_{k \in N} \alpha_b^{ak} t_{ak} + \sum_{k \in N} \alpha_a^{kb} t_{kb} \quad \forall\, [t_{ij}] \in \mathcal{T}(\vec{R}, \vec{C}), \quad \forall\, a, b \in N \tag{5.24}$$

$$\sum_{e \in E^+(k)} x_e^{ij} - \sum_{e \in E^-(k)} x_e^{ij} = \begin{cases} +\tau_{ij} & \text{if } k = i \\ -\tau_{ij} & \text{if } k = j \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i, j, k \in N \tag{5.25}$$

$$\sum_{i,j \in N} x_e^{ij} \leq \mu u_e \quad \forall\, e \in E \tag{5.26}$$

$$\sum_{k \in N} \alpha_k^{ab} = 1 \quad \forall\, a, b \in N \tag{5.27}$$

$$\alpha_k^{ab} \geq 0 \quad \forall\, k, a, b \in N \tag{5.28}$$

$$x_e^{ij} \geq 0 \quad \forall\, e \in E, \quad \forall\, i, j \in N \tag{5.29}$$

Constraints (5.24) correspond to the value of the demand τ_{ab} from node a to node b as given in equation (5.1). Constraints (5.25) correspond to routing of flows between each source-destination pair of the required value. Constraints (5.26) are the maximum utilization constraints for each link. Constraints (5.27) correspond to the traffic split ratios summing to 1 for each source-destination pair. The quantities t_{ij} in the RHS of (5.24) are constants and hence the constraints are linear. Note that there are an infinite set of constraints in (5.24), since there are $n(n-1)$ constraints for each $[t_{ij}] \in \mathcal{T}(\vec{R}, \vec{C})$.

The above linear program can be solved in polynomial time by the ellipsoid algorithm [S86] provided we can find a polynomial time separation oracle for the constraints (5.24). Given a set of values for the variables in the above linear program, the separation oracle needs to identify at least one constraint that is violated (if any),

or indicate otherwise. Clearly, constraints (5.25)-(5.27) can be verified in polynomial time.

To determine if the constraints in (5.24) are violated for any link, we need to either identify a source-destination pair (a, b) and a traffic matrix $[t_{ij}] \in \mathcal{T}(\vec{R}, \vec{C})$ such that the corresponding constraint is violated, or determine that all such constraints are satisfied. This can be done by verifying that for each source-destination pair (a, b), the linear program below, with variables t_{ij} for all $i, j \in N$, has optimum objective function value at most τ_{ab}. If not, the traffic matrix $[t_{ij}]$ obtained in the optimal solution of the linear program identifies the corresponding violating constraint in (5.24).

$$\text{maximize} \quad \sum_{k \in N} \alpha_b^{ak} t_{ak} + \sum_{k \in N} \alpha_a^{kb} t_{kb}$$

subject to

$$\sum_{j \in N, j \neq i} t_{ij} \leq R_i \quad \forall\, i \in N \tag{5.30}$$

$$\sum_{i \in N, i \neq j} t_{ij} \leq C_j \quad \forall\, j \in N \tag{5.31}$$

$$t_{ij} \geq 0 \quad \forall\, i, j \in N \tag{5.32}$$

5.3.2 Polynomial Size LP

In developing the polynomial size linear program, we first take the dual of the separation oracle linear program above. For a given source-destination pair (a, b), the dual linear program has non-negative variables $r(i, a, b)$ corresponding to each constraint in (5.30) and non-negative variables $c(j, a, b)$ corresponding to each constraint in (5.31).

$$\text{minimize} \quad \sum_{i \in N} R_i r(i,a,b) + \sum_{j \in N} C_j c(j,a,b)$$

subject to

$$\begin{aligned}
r(a,a,b) + c(b,a,b) &\geq \alpha_a^{ab} + \alpha_b^{ab} && (5.33)\\
r(a,a,b) + c(k,a,b) &\geq \alpha_b^{ak} \quad \forall\, k \in N, k \neq b && (5.34)\\
r(k,a,b) + c(b,a,b) &\geq \alpha_a^{kb} \quad \forall\, k \in N, k \neq a && (5.35)\\
r(i,a,b), c(i,a,b) &\geq 0 \quad \forall\, i \in N && (5.36)
\end{aligned}$$

It follows directly from strong duality of linear programming [S86] that for each source-destination pair (a,b), the primal (separation oracle) linear program has an optimum objective function value of at most τ_{ab} if and only if the dual linear program has a feasible solution with objective function value at most τ_{ab}. The requirement that the dual linear program, for all $a, b \in N$, have feasible solutions with objective function value at most τ_{ab} can be modeled as the following constraint:

$$\sum_{i \in N} R_i r(i,a,b) + \sum_{j \in N} C_j c(j,a,b) \leq \tau_{ab} \quad \forall\, a,b \in N$$

This allows us to remove the infinite set of constraints in (5.24) and add the above constraint and constraints (5.33)-(5.36) from the dual linear programs to obtain the following polynomial size linear program for our problem:

$$\text{minimize} \quad \mu$$

subject to

$$\sum_{e \in E^+(k)} x_e^{ij} - \sum_{e \in E^-(k)} x_e^{ij} = \begin{cases} +\tau_{ij} & \text{if } k = i \\ -\tau_{ij} & \text{if } k = j \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i,j,k \in N \quad (5.37)$$

$$\sum_{i,j \in N} x_e^{ij} \leq \mu u_e \quad \forall\, e \in E \tag{5.38}$$

$$\sum_{k \in N} \alpha_k^{ab} = 1 \quad \forall\, a, b \in N \tag{5.39}$$

$$\sum_{i \in N} R_i r(i,a,b) + \sum_{j \in N} C_j c(j,a,b) \leq \tau_{ab} \quad \forall\, a, b \in N \tag{5.40}$$

$$r(a,a,b) + c(b,a,b) \geq \alpha_a^{ab} + \alpha_b^{ab} \quad \forall\, a, b \in N \tag{5.41}$$

$$r(a,a,b) + c(k,a,b) \geq \alpha_b^{ak} \quad \forall\, k, a, b \in N, k \neq b \tag{5.42}$$

$$r(k,a,b) + c(b,a,b) \geq \alpha_a^{kb} \quad \forall\, k, a, b \in N, k \neq a \tag{5.43}$$

$$\alpha_k^{ab} \geq 0 \quad \forall\, k, a, b \in N \tag{5.44}$$

$$x_e^{ij} \geq 0 \quad \forall\, e \in E, \quad \forall\, i, j \in N \tag{5.45}$$

$$r(i,a,b), c(i,a,b) \geq 0 \quad \forall\, i, a, b \in N \tag{5.46}$$

This linear program has $n^2(n-1)$ constraints in (5.37), m constraints in (5.38), $n(n-1)$ constraints each in (5.39)-(5.41), $n(n-1)(n-2)$ constraints each in (5.42)-(5.43), $n^2(n-1)$ constraints in (5.44), $mn(n-1)$ constraints in (5.45), and $2n^2(n-1)$ constraints in (5.46), for a total of $O(mn^2)$ constraints. The number of variables is $n^2(n-1) + n(n-1) + mn(n-1) + 2n^2(n-1) + 1 = O(mn^2)$.

By using per-source flow variables x_e^i instead of per source-destination variables x_e^{ij}, the number of variables and constraints in the above linear program can be reduced to $O(n^3)$.

5.4 Maximum Throughput Direct Source-Destination Path Routing

As discussed in Section 1.6, 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. [DGGMR99] and Kumar et al. [KRSY01]. In order to make throughput

comparisons with two-phase routing, we consider the multi-path version of direct source-destination routing where the traffic from a source to a destination can be split along multiple paths -- both the paths and the ratios in which traffic is split among them is fixed a priori. An instance of this scheme is completely described by specifying how a unit flow is (splittably) routed between each source-destination pair in the network.

Erlebach and Rüegg [ER04] consider the problem of minimum cost direct source-destination (multi-)path routing of hose traffic under given link costs. They give a linear program with an infinite number of constraints (and, a polynomial size separation oracle linear program) that is suitable for solving using the ellipsoid method [S86]. Because the ellipsoid method gives running times that are not feasible for practical implementations, the authors in [ER04] also give a cutting-plane heuristic for solving the infinite size linear program and obtain reasonable running times for the experiments reported. However, this cutting-plane heuristic can have exponential running times in the worst case.

In this section, we develop a *polynomial size linear program* for maximum throughput multi-path routing of hose traffic under given link capacities. Our technique can be used to obtain a polynomial size linear program for the minimum cost version of the problem also, thus improving the result in [ER04].

Given a network with link capacities u_e and constraints R_i, C_j on the ingress-egress traffic, we consider the problem of direct source-destination path routing so as to maximize the network throughput. We use the same approach used earlier in this chapter for the minimum cost network design and maximum throughput network routing problems for two-phase routing with generalized traffic split ratios. We begin with a linear programming formulation with an infinite number of constraints and a polynomial size separation oracle linear program for it, and then combine the two into a polynomial size linear program.

5.4.1 LP with Infinite Constraints and Separation Oracle

The fixed path routing for each source-destination pair (i, j) can be specified by a set of *unit flow variables* f_e^{ij}, where f_e^{ij} denotes the fraction of traffic from i to j that traverses link e in the network. Let μ denote the maximum utilization of any link in the network. As explained earlier, maximizing the throughput λ is equivalent to minimizing the maximum link utilization μ. The linear programming formulation is as follows:

$$\text{minimize } \mu$$

subject to

$$\sum_{e \in E^+(k)} f_e^{ij} - \sum_{e \in E^-(k)} f_e^{ij} = \begin{cases} +1 & \text{if } k = i \\ -1 & \text{if } k = j \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i, j, k \in N \tag{5.47}$$

$$\sum_{i,j \in N} t_{ij} f_e^{ij} \leq \mu u_e \quad \forall\, e \in E, \quad \forall\, [t_{ij}] \in \mathcal{T}(\vec{R}, \vec{C}) \tag{5.48}$$

$$f_e^{ij} \geq 0 \quad \forall\, e \in E, \quad \forall\, i, j \in N \tag{5.49}$$

Constraints (5.47) correspond to routing of unit flows between each source-destination pair for determining the fixed paths. Constraints (5.48) are the maximum utilization constraints for each link. The quantities t_{ij} in the LHS of (5.48) are constants and hence the constraints are linear. Note that there is an infinite set of constraints in (5.48), since there are m constraints for each $[t_{ij}] \in \mathcal{T}(\vec{R}, \vec{C})$.

The above linear program can be solved in polynomial time by the ellipsoid algorithm [S86] provided we can find a polynomial time separation oracle for the constraints (5.48). Given a set of values for the variables in the above linear program, the separation oracle needs to identify at least one constraint that is violated (if any), or indicate otherwise. Clearly, constraint (5.47) can be verified in polynomial time.

To determine if the constraints in (5.48) are violated for any link, we need to either identify a link e and a traffic matrix $[t_{ij}] \in \mathcal{T}(\vec{R}, \vec{C})$ such that the corresponding constraint is violated, or determine that all such constraints are satisfied. This can be done by verifying that for each link $\ell \in E$, the linear program below, with variables t_{ij} for all $i, j \in N$, has optimum objective function value at most μ. If not, the traffic matrix $[t_{ij}]$ obtained in the optimal solution of the linear program identifies the corresponding violating constraint in (5.48).

$$\text{maximize} \quad \sum_{i,j \in N} \frac{f_\ell^{ij} t_{ij}}{u_\ell}$$

subject to

$$\sum_{j \in N, j \neq i} t_{ij} \leq R_i \quad \forall\, i \in N \tag{5.50}$$

$$\sum_{i \in N, i \neq j} t_{ij} \leq C_j \quad \forall\, j \in N \tag{5.51}$$

$$t_{ij} \geq 0 \quad \forall\, i, j \in N \tag{5.52}$$

5.4.2 Polynomial Size LP

In developing the polynomial size linear program, we first take the dual of the separation oracle linear program above. For a given link ℓ, the dual linear program has non-negative variables $r(i, \ell)$ corresponding to each constraint in (5.50) and non-negative variables $c(j, \ell)$ corresponding to each constraint in (5.51).

$$\text{minimize} \quad \sum_{i \in N} R_i r(i, \ell) + \sum_{j \in N} C_j c(j, \ell)$$

subject to

$$r(i, \ell) + c(j, \ell) \geq \frac{f_\ell^{ij}}{u_\ell} \quad \forall\, i, j \in N \tag{5.53}$$

$$r(i,\ell), c(i,\ell) \geq 0 \ \forall\, i \in N \tag{5.54}$$

It follows directly from strong duality of linear programming [S86] that for each link $\ell \in E$, the primal (separation oracle) linear program has an optimum objective function value of at most μ if and only if the dual LP has a feasible solution with objective function value at most μ.

The requirement that the dual linear programs, for all $\ell \in E$, have feasible solutions with objective function value at most μ can be modeled as the following constraint:

$$\sum_{i \in N} R_i r(i,\ell) + \sum_{j \in N} C_j c(j,\ell) \leq \mu \ \ \forall\, \ell \in E$$

This allows us to remove the infinite set of constraints in (5.48) and add the above constraint and constraints (5.53)-(5.54) from the dual LPs to obtain the following polynomial size LP for our problem:

---

minimize μ

subject to

$$\sum_{e \in E^+(k)} f_e^{ij} - \sum_{e \in E^-(k)} f_e^{ij} = \begin{cases} +1 & \text{if } k = i \\ -1 & \text{if } k = j \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i,j,k \in N \tag{5.55}$$

$$r(i,\ell) + c(j,\ell) \geq \frac{f_\ell^{ij}}{u_\ell} \ \ \forall\, \ell \in E, \ \ \forall\, i,j \in N \tag{5.56}$$

$$\sum_{i \in N} R_i r(i,\ell) + \sum_{j \in N} C_j c(j,\ell) \leq \mu \ \ \forall\, \ell \in E \tag{5.57}$$

$$r(i,\ell), c(i,\ell) \geq 0 \ \ \forall\, i \in N, \ \ \forall\, \ell \in E \tag{5.58}$$

$$f_e^{ij} \geq 0 \ \ \forall\, e \in E, \ \ \forall\, i,j \in N \tag{5.59}$$

This linear program has $n^2(n-1)$ constraints in (5.55), $mn(n-1)$ constraints in (5.56), m constraints in (5.57), $2mn$ constraints in (5.58), and $mn(n-1)$ constraints in (5.59), for a total of $O(mn^2)$ constraints. The number of variables is $mn(n-1) + 2mn + 1 = O(mn^2)$.

5.5 Throughput Comparisons on ISP Topologies

In this section, we compare the throughput performance of three schemes for routing hose traffic, namely, (i) two-phase routing with intermediate node dependent traffic split ratios α_k, (ii) two-phase routing with generalized traffic split ratios α_k^{ij}, and (iii) direct source-destination path routing, For (i), we use the linear programming formulation from Section 4.1.1. For (ii) and (iii), we use the linear programming formulations developed in this chapter. We use CPLEX [CPLEX] to solve all linear programs.

5.5.1 Topologies and Link/Ingress-Egress Capacities

We use the six ISP maps from the Rocketfuel dataset which had accompanying (deduced) OSPF/IS-IS weights [SMWH, SMW02, MSWA02]. These topologies list multiple intra-PoP (Point of Presence) routers and/or multiple intra-city PoPs as individual nodes. We coalesced such nodes so that nodes correspond to cities and the topology represents geographical PoP-to-PoP ISP topologies. Some data about the original topologies and their coalesced versions is listed in Table 5.1.

The topologies provided by Rocketfuel did not include the capacities of the links, which were needed for our study. The Rocketfuel maps did include derived OSPF/ISIS weights of links, which were computed to match observed routes. In the absence of any other information on capacities, we need a way to deduce the link capacities from the weights. For this purpose, we assumed that the given link weights are the Cisco default setting for OSPF weights, i.e., inversely-proportional to the link capacities

Topology	Routers (original)	Links (inter-router)	PoPs (coalesced)	Links (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 5.1: Rocketfuel topologies with AS number and name. The table lists the original number of routers and inter-router links, and the number of coalesced PoPs and inter-PoP links.

[Cisco97]. The link capacities obtained in this manner turned out to be symmetric, i.e., $u_{ij} = u_{ji}$ for all $(i,j) \in E$.

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 $R_i(= C_i) \propto \sum_{e \in E^+(i)} u_e$.

5.5.2 Experiments and Results

We denote the throughput values for the three different schemes as follows: (i) λ_{TPR} for two-phase routing with intermediate node dependent traffic split ratios, (ii) λ_{GTPR} for two-phase routing with generalized traffic split ratios, and (iii) λ_{FPR} for direct source-destination routing along fixed paths. Clearly, $\lambda_{TPR} \leq \lambda_{GTPR} \leq \lambda_{DPR} \leq \lambda_{OPT}$, where λ_{OPT} is the throughput of the optimal scheme.

Experiments on the six Rocketfuel topologies for maximum throughput two-phase routing in Section 4.6 showed that the throughput performance of two-phase routing with intermediate node dependent split ratios α_i is within 6% of that of the optimal

scheme. (For two of the topologies, it actually matches that of the optimal scheme.) Hence, for these topologies, there is clearly not much room for throughput improvement when we generalize the traffic split ratios in two-phase routing or even consider arbitrary fixed path routing solutions.

For the Tiscali 3257 topology, the CPLEX processes for solving the linear programs for λ_{GTPR} and λ_{FPR} ran out of memory and were killed on a 2.4GHz Dual Xeon machine with 1GB of RAM and running Linux. This was the fastest machine with the highest RAM that we had access to for running CPLEX. For the Exodus 3967 and Telstra 1221 topologies, the throughput of two-phase routing with traffic split ratios α_i matches that of the optimal scheme (as reported in Section 4.6), hence $\lambda_{TPR} = \lambda_{GTPR} = \lambda_{DPR}$. The latter was observed to be the case with the remaining three Rocketfuel topologies also.

Thus, on five of the six Rocketfuel topologies, the *throughput of two-phase routing with α_i traffic split ratios equals that with generalized traffic split ratios and matches the throughput of direct source-destination routing along fixed paths.* (Recall that the pathological example for the improvement in throughput of two-phase routing with generalized traffic split ratios in Section 5.1 exploited $R_i \neq C_i$ for some nodes i and asymmetric link capacities – both of these are not present in the Rocketfuel topologies.) Given the identical throughput performance of the two versions of two-phase routing, the simpler version with intermediate node-dependent traffic split ratios α_i is preferred because of its ability to support indirection in specialized service overlay models like i3.

These experiments on actual ISP topologies indicate that two-phase routing achieves its robustness to traffic variation without compromising throughput performance compared to previous approaches like direct source-destination path routing. Its throughput performance is within 6% of that of the optimal scheme on the evaluated topologies. Thus, two-phase routing is able to handle highly variable traffic in a capacity efficient manner and provide the desirable properties of (i) static provisioning at the

optical layer in IP-over-Optical networks, and (ii) supporting indirection in specialized service overlay networks. Direct source-destination routing does not meet these requirements.

Chapter 6

Protecting Against Router Failures in IP-over-Optical Networks

In this chapter, we focus on two-phase routing in IP-over-Optical networks where routers are interconnected over a switched optical backbone, also called IP-over-OTN (Optical Transport Network). In this architecture, the first and second phase paths are realized at the optical layer with the routers at intermediate nodes being responsible for (de)multiplexing traffic to its final destination. Studies like Labovitz et al. [LAJ98] indicate that IP routers are 200 times more unreliable than traditional carrier-grade switches and average 1219 minutes of down time per year. Given this unreliability of routers, we consider how two-phase routing in IP-over-OTN can be made resilient against router node failures. If the router at a node goes down, the node ceases to perform its intermediate node functionality. Thus, the traffic split ratio corresponding to this node has to be redistributed to other intermediate nodes.

We propose two different schemes for provisioning the optical layer to handle router node failures – one that is failure node independent and static (called *failure independent provisioning*), and another that is failure node dependent and dynamic (called *failure dependent provisioning*). For both mechanisms, we consider sharing bandwidth across single router failure scenarios. For failure independent provisioning,

the optical layer is statically provisioned a priori so as to handle any single router failure scenario. For failure independent provisioning, the paths in the optical layer along which traffic is redistributed to other intermediate nodes are provisioned after failure – this allows sharing of optical layer bandwidth at the link level and may lead to lower restoration capacity overhead compared to the first scheme.

We explain why a simple choice of redistribution ratios proportional to the original traffic split ratios does not lead to optimal throughput. Hence, our problem formulations must accommodate arbitrary redistribution of traffic split ratios after failure. We develop linear programming formulations for both schemes and a fast combinatorial algorithm for the second scheme so as to maximize network throughput. In each case, we determine (i) the optimal distribution of traffic to various intermediate routers for both normal (no-failure) and failure conditions, and (ii) provisioning of optical layer circuits to provide the needed inter-router links.

For failure independent provisioning, we prove that the throughput is at most $\frac{n-1}{n}$ times that for the unprotected case, where n is the number of nodes. For the six Rocketfuel topologies, the achieved throughput is within 2% of this theoretical upper bound. Also, the throughput for failure dependent provisioning is less than 1% more than that for failure independent provisioning on the evaluated topologies. Hence, given the static optical layer provisioning property of failure independent provisioning, it might be the preferred scheme for protecting against router node failures in IP-over-Optical networks.

We assume a single router failure model under which bandwidth can be shared across different router node failure scenarios. The focus on shared backup bandwidth allocation is because of its reduced cost and the increased complexity of the optimization problems that arises from sharing backup bandwidth.

6.1 Two-Phase Routing in IP-over-OTN

Core IP networks are often deployed by interconnecting routers over a switched optical backbone, also called IP-over-OTN (Optical Transport Network). Some aspects of this architecture were discussed in Section 1.5.1.

The application of two-phase routing to IP-over-OTN was described in Section 2.2.1. Two-phase routing, as envisaged for IP-over-OTN, establishes the fixed bandwidth Phase 1 and Phase 2 tunnels at the optical layer. Thus, the *optical layer is statically provisioned* and does not need to be reconfigured in response to traffic changes. IP packets are routed end-to-end with *IP layer processing at a single intermediate node only.* While in transit at the optical layer inside either Phase 1 or Phase 2 paths, packets do enter the router but appear as transit traffic at the Optical Cross-Connect (OXC) only.

The IP layer packet processing at an intermediate node works as follows. The optical layer circuit is dropped at the IP router at the node (through OXC-to-router links), wherein the packets are multiplexed back to the OXC (through router-to-OXC links) to be routed through direct optical layer circuits to their final destinations. This is illustrated in Figure 2-3. Thus, if the route at a node fails, then that node ceases to perform the functionality of an intermediate node in two-phase routing. We consider how to make two-phase routing resilient to such IP router failures.

6.2 Making Two-Phase Routing Resilient to Router Failures

We consider extending the two-phase routing scheme for protecting against *router node failures.* In the term "router node failure", node refers to a PoP, hence it includes the failure of all routers in a PoP. When a router at a node f fails, any other node i cannot split any portion of its originating traffic to intermediate node

f. Hence, it must redistribute the traffic split ratio α_f among other nodes $j \neq f$. Accordingly, let β_{jf} denote the portion of α_f that is redistributed to node j when router node f fails. Then, we must have

$$\sum_{j \in N, j \neq f} \beta_{jf} = \alpha_f \quad \forall\, f \in N \tag{6.1}$$

Note that since only the router (and not the OXC) at node f fails, this node can continue to be on the Phase 1 and Phase 2 paths for optical layer switching. However, no traffic enters or leaves the network at node f.

We propose two different schemes for provisioning the optical layer in IP-over-OTN in order to handle the redistribution of traffic split ratios after router node failures. We discuss these next. Algorithms for computing the traffic split ratios and their redistribution after failure and the routing of Phase 1 and Phase 2 paths so as to maximize throughput under the two protection schemes are presented in Sections 6.3 and 6.4 respectively.

6.2.1 Failure Independent Provisioning

In the first scheme, called "Failure Independent Provisioning", the restoration demand from node i to node j at the optical layer, for each $i, j \in N$, is provisioned *a priori* so as to handle the "worst case" router node failure scenario. The maximum additional traffic split ratio that node j needs to handle is $\max_{f \neq j} \beta_{jf}$. Thus, the modified split ratio α'_j associated with each node j is

$$\alpha'_j = \alpha_j + \max_{f \in N, f \neq j} \beta_{jf} \quad \forall\, j \in N \tag{6.2}$$

The demand that needs to be statically provisioned at the optical between nodes i and j so as to protect against any single router node failure is $\alpha'_j R_i + \alpha'_i C_j$. Since this does not depend on which router node fails, hence the name of the scheme.

In equation (6.2), the value of the traffic split ratio α'_j for intermediate node j is determined by the node f that achieves the maximum value of β_{jf}. For different nodes j, the maximum value of β_{jf} could be achieved by different (failure) nodes f. Thus, for a given router node failure, all the split ratios α'_j may not be required to be as high as the maximum value given by RHS of (6.2). Also, in this scheme, optical layer bandwidth is shared at the connection/path level (Phase 1 and Phase 2 demands between node pairs) and not at the link level across different failure scenarios. Because of these reasons, failure independent provisioning may not achieve the most capacity efficient sharing of restoration bandwidth across different router node failure scenarios. However, the advantage of the scheme is that it preserves the static nature of the original two-phase routing scheme.

6.2.2 Failure Dependent Provisioning

In the second scheme, called "Failure Dependent Provisioning", the restoration demand from node i to node j at the optical layer depends on the node f which failed and is provisioned in a *reactive* manner after the node f fails, the value of the demand being $\beta_{jf}R_i + \beta_{if}C_j$, which could be different for different failed nodes f. By "reactive", we mean that cross-connects need to be setup for redistributing traffic to other intermediate nodes after failure – this is because backup bandwidth is shared at the link level in the optical layer. The scheme allows increased sharing of restoration bandwidth across different router node failure scenarios and may lead to increased throughput. As we shall show in Section 6.4.4, the maximum throughput routing problem for this scheme also admits a fast combinatorial algorithm (FPTAS).

The cross-connects on the Phase 1 and Phase 2 paths that originate or terminate at the failed router node are kept intact (after failure) even though they carry no traffic during the duration of failure of the respective router node. This has the advantage that when the failed router node comes back up, traffic can be immediately reverted back to that intermediate node without waiting for the Phase 1 and Phase 2 paths

to/from this node to be reprovisioned. This is also consistent with how IP layer failures are handled in practice in IP-over-Optical networks – optical layer circuits that carry traffic under normal (no-failure) conditions but are not needed after an IP layer failure are kept in place for use when the IP layer failure is repaired.

6.3 Maximizing Throughput for Failure Independent Provisioning

Given a network with link capacities u_e and constraints R_i, C_j on the ingress-egress traffic, we consider the problem of routing with protection against router node failures through *failure independent provisioning* so as to maximize throughput. The throughput is the maximum multiplier λ such that all matrices in $\lambda \cdot \mathcal{T}(\vec{R}, \vec{C})$ can be feasibly routed with protection against router node failures under the scheme.

We first consider the straightforward approach of redistributing traffic for the failed intermediate router node to other intermediate nodes in proportion to that of the original traffic split ratios obtained by solving the maximum throughput routing problem for the unprotected case. We explain why this approach does not lead to the best throughput. Next, we consider a linear programming formulation that accommodates arbitrary redistribution of traffic split ratios after failure.

6.3.1 Redistributing Traffic in Proportion to Split Ratios for the Unprotected Case

Under this approach, we would solve the maximum throughput routing problem for the unprotected case and then make the failure redistribution ratios $\beta_{jf} \propto \alpha_j$. Using equation (6.1), this gives

$$\beta_{jf} = \frac{\alpha_j \alpha_f}{\sum_{i \neq f} \alpha_i} = \frac{\alpha_j \alpha_f}{1 - \alpha_f} \quad \forall\, j \neq f$$

Substituting this back into equation (6.2), we have

$$\alpha'_j = \alpha_j \left(1 + \max_{f \in N, f \neq j} \frac{\alpha_f}{1 - \alpha_f}\right) = \alpha_j \left(1 + \frac{\max_{f \neq j} \alpha_f}{1 - \max_{f \neq j} \alpha_f}\right) \quad \forall\, j \in N$$

Thus, the maximum relative increase in split ratio compared to the unprotected case is determined by the intermediate node f with the largest split ratio α_f for the unprotected case, i.e., the node f whose failure affects the largest fraction of traffic in the network. If λ_{UNP} denotes the throughput for the unprotected case, then the throughput obtained for failure independent provisioning using this method will be approximately

$$\frac{\lambda_{UNP}}{1 + \frac{\max_{f \neq j} \alpha_f}{1 - \max_{f \neq j} \alpha_f}}$$

Thus, to achieve higher throughputs for two-phase routing with protection against router node failures, it is not desirable for any intermediate node to carry a large proportion of traffic compared to the average of $\frac{1}{n}$. Otherwise, in the event of a route node failure at any of these intermediate nodes, a relatively large fraction of the traffic needs to be restored, thus increasing the resources reserved for restoration (and decreasing the throughput).

However, as we observed in the experiments in Section 4.6, a small number of intermediate nodes is preferred for the unprotected case – these nodes are presumably located at geographical center(s) of the network and provide the best opportunities for serving as intermediate nodes without increasing the length of end-to-end paths or decreasing throughput significantly. A small number of intermediates nodes is associated with a relatively large traffic split ratio for each such node – hence the above method will not lead to the maximum throughput in such cases.

This points to the need for an optimization approach that takes router node failure explicitly into account rather than redistributing traffic for the failed intermediate node to other nodes in ratios proportional to the original traffic split ratios obtained by solving the unprotected version of the problem.

6.3.2 Linear Programming Formulation

In this section, we develop a linear programming formulation for failure independent provisioning that accommodates arbitrary redistribution of traffic split ratios after failure. We first derive an expression for throughput in terms of the modified traffic split ratios α'_j.

Because other intermediate nodes should be able to take up the traffic split ratios of the failed node f, a necessary and sufficient condition for the α'_j values to be feasible is

$$\sum_{j \in N, j \neq f} \alpha'_j \geq 1 \quad \forall\, f \in N \tag{6.3}$$

It suffices to have the minimum of these n sums exactly equal to 1. Thus, in the case that this is not so, the traffic split ratios can be divided by

$$\lambda = \min_{f \in N} \sum_{j \in N, j \neq f} \alpha'_j \tag{6.4}$$

(normalized) to achieve the desired result – in which case all traffic matrices in $\lambda \cdot \mathcal{T}(\vec{R}, \vec{C})$ can be feasibly routed. Thus, the appropriate measure of throughput in this case is the quantity λ as defined above.

We adopt the standard network flow terminology from [AMO93]. Let x_e^{ij} denote the flow value on link e for routing the demand of $\alpha'_j R_i + \alpha'_i C_j$ from node i to node j. Then, the problem of two-phase routing with failure independent provisioning so as to maximize the network throughput can be formulated as the following linear program:

---

maximize λ

subject to

$$\lambda \leq \sum_{j \in N, j \neq f} \alpha'_j \quad \forall\, f \in N \tag{6.5}$$

$$\sum_{e \in E^+(k)} x_e^{ij} - \sum_{e \in E^-(k)} x_e^{ij} = \begin{cases} \alpha'_j R_i + \alpha'_i C_j & \text{if } k = i \\ -\alpha'_j R_i - \alpha'_i C_j & \text{if } k = j \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i, j, k \in N \tag{6.6}$$

$$\sum_{i,j \in N} x_e^{ij} \leq u_e \quad \forall\, e \in E \tag{6.7}$$

$$\alpha'_j \geq 0 \quad \forall\, j \in N \tag{6.8}$$

$$x_e^{ij} \geq 0 \quad \forall\, e \in E, \quad \forall\, i, j \in N \tag{6.9}$$

Constraints (6.5) correspond to the definition of throughput in equation (6.4). Constraints (6.6) correspond to the routing of $\alpha'_j R_i + \alpha'_i C_j$ amout of flow from node i to node j. The link capacity constraints are in (6.7).

By using per-source flow variables x_e^i instead of per source-destination variables x_e^{ij}, the number of variables in the above LP can be reduced to $O(nm)$.

This linear program can be solved in polynomial time using a general linear programming algorithm [S86]. The traffic specified by the network ingress-egress capacities is feasible for the network if the throughput λ in the optimal solution is at least 1. Assuming this is so and given the values α'_i in an optimal solution of the above linear program, the (normalized) traffic split ratios α_i under normal (no-failure) conditions can be chosen to satisfy (i) $\sum_{i \in N} \alpha_i = 1$, and (ii) $\alpha_i \leq \alpha'_i$ for all i, but they may not be determined uniquely. One solution is to make the α_i's proportional to the α'_i's, i.e., $\alpha_i = \frac{\alpha'_i}{\sum_{i \in N} \alpha'_i}$. The failure redistribution ratios β_{if} are also not determined uniquely, but can be chosen in a feasible manner so as to satisfy equation (6.1) because if the optimal throughput is at least 1, then constraints (6.5) imply necessary and sufficient condition (6.3).

6.3.3 An Upper Bound on the Throughput Relative to the Unprotected Case

Let λ_{UNP} and λ_{FIP} denote the maximum throughputs respectively for two-phase routing for the unprotected case and for router node failure protection using failure independent provisioning. The following theorem establishes an upper bound on λ_{FIP} relative to λ_{UNP}.

Theorem 6.3.1 *The throughput λ_{FIP} of two-phase routing with failure independent provisioning to protect against router node failures is at most $\frac{n-1}{n}$ times the throughput λ_{UNP} for the unprotected case.*

Proof: Let α'_i be the traffic split ratios in an optimal solution for the linear program for maximizing throughput in failure independent provisioning. Summing constraints (6.5) for all $f \in N$, we have

$$n\lambda_{FIP} \leq (n-1)\sum_{i \in N} \alpha'_i \tag{6.10}$$

Now recall that in the linear programming formulation in Section 4.1.1 for maximizing throughput for the unprotected case, the throughput is the sum of the traffic split ratios (when they are not constrained to sum to 1). Also, the linear program for failure independent provisioning differs from that for the unprotected case only in the definition of throughput (in the former case, the throughput is the minimum value of the n possible sums of any $n-1$ of the traffic split ratios). Thus, we must have

$$\sum_{i \in N} \alpha'_i \leq \lambda_{UNP}$$

Using this in (6.10), we obtain the upper bound claimed in the theorem. ■

As we will see in Section 6.5, the ratio $\frac{\lambda_{FIP}}{\lambda_{UNP}}$ for all evaluated Rocketfuel topologies is within 2% to this theoretical upper bound.

We provide an intuitive explanation that gives the quantity $\frac{n-1}{n}$ as a rough theoretical estimate of the ratio $\frac{\lambda_{FIP}}{\lambda_{UNP}}$. The average split ratio per node under normal (no-failure) conditions is $\frac{1}{n}$. When a router node fails, each of the other $n-1$ nodes takes up on the average an additional split ratio of $\frac{1}{n(n-1)}$. Thus, the estimate for the ratio $\frac{\lambda_{FIP}}{\lambda_{UNP}}$ is

$$\frac{\frac{1}{n}}{\frac{1}{n}+\frac{1}{n(n-1)}} = \frac{n-1}{n}$$

6.4 Maximizing Throughput for Failure Dependent Provisioning

Given a network with link capacities u_e and constraints R_i, C_j on the ingress-egress traffic, we consider the problem of routing with protecting against router node failures through *failure dependent provisioning* so as to maximize throughput. The throughput is the maximum multiplier λ such that all matrices in $\lambda \cdot \mathcal{T}(\vec{R}, \vec{C})$ can be feasibly routed with protection against router node failures under the scheme.

For failure dependent provisioning, we work explicitly with both normal (no-failure) traffic split ratios α_j and failure redistribution ratios β_{jf}. Suppose we relax the requirement that the traffic split ratios α_j sum to 1 in a feasible solution of the problem. Consider the sum

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

The traffic split ratios α_j can be divided by λ (normalized) so that they sum to 1, in which case all matrices in $\lambda \cdot \mathcal{T}(\vec{R}, \vec{C})$ can be feasibly routed. (The failure redistribution ratios β_{jf} values are scaled by the same amount.) Thus, the appropriate measure of throughput in this case is the quantity λ above *when the traffic split ratios α_j are not constrained to sum to* 1.

6.4.1 Link Flow Based LP Formulation

Let x_e^{ij} denote the flow value on link e for routing the demand of $\alpha_j R_i + \alpha_i C_j$ from node i to node j under normal (no-failure) conditions. Let y_{ef}^{ij} be the *restoration flow* that appears on link e for routing the demand of $\beta_{jf} R_i + \beta_{if} C_j$ from node i to node j after failure of node f. Then, the problem of two-phase routing with failure dependent provisioning so as to maximize the network throughput can be formulated as the following link-indexed linear program:

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

subject to

$$\sum_{e \in E^+(k)} x_e^{ij} - \sum_{e \in E^-(k)} x_e^{ij} = \begin{cases} \alpha_j R_i + \alpha_i C_j & \text{if } k = i \\ -\alpha_j R_i - \alpha_i C_j & \text{if } k = j \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i,j,k \in N \qquad (6.11)$$

$$\sum_{e \in E^+(k)} y_{ef}^{ij} - \sum_{e \in E^-(k)} y_{ef}^{ij} = \begin{cases} \beta_{jf} R_i + \beta_{if} C_j & \text{if } k = i \\ -\beta_{jf} R_i - \beta_{if} C_j & \text{if } k = j \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i,j,f,k \in N \qquad (6.12)$$

$$\sum_{j \in N, j \neq f} \beta_{jf} = \alpha_f \quad \forall\, f \in N \qquad (6.13)$$

$$\sum_{i,j \in N} x_e^{ij} + \sum_{i,j \in N} y_{ef}^{ij} \leq u_e \quad \forall\, e \in E, \quad \forall\, f \in N \qquad (6.14)$$

$$\alpha_i, \beta_{if} \geq 0 \quad \forall\, i,f \in N \qquad (6.15)$$

$$x_e^{ij}, y_{ef}^{ij} \geq 0 \quad \forall\, e \in E, \quad \forall\, i,j,f \in N \qquad (6.16)$$

Constraints (6.11) corresponds to the routing of $\alpha_j R_i + \alpha_i C_j$ amount of flow from node i to node j under normal (no-failure) conditions. Constraints (6.12) correspond to the routing of *restoration flow* of value $\beta_{jf} R_i + \beta_{if} C_j$ from node i to node j after failure of node f. Constraints (6.13) state that for a given router node failure, the

sum of the traffic redistribution ratios should sum to the traffic split ratio (under no-failure conditions) of the failed node. Constraints (6.14) are the link capacity constraints – the sum of working flow under normal (no-failure) conditions and maximum restoration flow under any router node failure is at most the capacity of the link.

By using per-source flow variables x_e^i and y_{ef}^i instead of per source-destination variables x_e^{ij} and y_{ef}^{ij} respectively, the number of variables in the above linear program can be reduced to $O(n^2m)$. This is still a factor of n more variables than the linear program for failure independent provisioning. It is well known that running times of general linear programming based algorithms for network problems do not scale well with increasing network size. Fortunately, our problem formulation for failure dependent provisioning accommodates a fast combinatorial algorithm. Moreover, because the combinatorial algorithm we develop works explicitly with paths, delay constraints on the paths can be efficiently handled within the optimization framework, as explained for the unprotected case in Section 4.3.6. In contrast, the link flow based linear program formulation above does not deal explicitly with paths, and hence, cannot accommodate constraints like bounding path delay.

6.4.2 Path Flow Based LP Formulation

In this section, we develop a path flow based linear programming formulation for this problem. This will be subsequently used to develop the fast combinatorial algorithm (FPTAS) in Section 6.4.4.

Let $x(P)$ denote the flow on path P under normal (no-failure) conditions. Let $y_f(P)$ be the *restoration flow* that appears on path P after failure of node f. Let $\mathcal{P}_{ij}$ denote the set of all paths from node i to node j. Then, the problem of two-phase routing with failure dependent provisioning so as to maximize the network throughput can be formulated as the following path-indexed linear program:

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

subject to

$$\sum_{P \in \mathcal{P}_{ij}} x(P) = \alpha_j R_i + \alpha_i C_j \quad \forall\, i,j \in N \tag{6.17}$$

$$\sum_{P \in \mathcal{P}_{ij}} y_f(P) = \beta_{jf} R_i + \beta_{if} C_j \quad \forall\, i,j,f \in N \tag{6.18}$$

$$\sum_{j \in N, j \neq f} \beta_{jf} = \alpha_f \quad \forall\, f \in N \tag{6.19}$$

$$\sum_{i,j \in N} \sum_{P \in \mathcal{P}_{ij}, P \ni e} x(P) + \sum_{i,j \in N} \sum_{P \in \mathcal{P}_{ij}, P \ni e} y_f(P) \leq u_e \quad \forall\, e \in E, f \in N \tag{6.20}$$

$$\alpha_i, \beta_{if} \geq 0 \quad \forall\, i,f \in N \tag{6.21}$$

$$x(P), y_f(P) \geq 0 \quad \forall\, P \in \mathcal{P}_{ij}, \quad \forall\, i,j,f \in N \tag{6.22}$$

In Section 6.4.3, we state the dual of the above path-indexed linear program. In general, a network can have an exponential number of paths (in the size of the network). Hence, this (primal) linear program can have possibly exponential number of variables and its dual can have an exponential number of constraints – they are both not suitable for solving the problem on medium to large sized networks. The usefulness of the primal and dual formulation is in designing a fast combinatorial algorithm for the problem.

6.4.3 Dual of Path Flow Based LP Formulation

We will use a primal-dual approach to develop a fast combinatorial algorithm (FPTAS) for failure dependent provisioning that computes the traffic split ratios and routing of Phase 1 and Phase 2 paths up to $(1+\epsilon)$-factor of the optimal objective function value (maximum throughput) for any $\epsilon > 0$. The primal-dual scheme extends the approach used for maximizing throughput for the unprotected case in Section 4.3.

We begin with the dual formulation of the linear program in Section 6.4.2. The dual formulation associates a variable π_{ij} with each demand constraint in (6.17), a variable ψ_{ijf} with each demand constraint in (6.18), a variable σ_f with each traffic

split ratio redistribution constraint in (6.19), and a non-negative variable $w(e,f)$ with each link capacity constraint in (6.20). The dual program can be written as:

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

subject to

$$\sum_{e\in P}\sum_{f\in N} w(e,f) \geq \pi_{ij} \quad \forall\, P\in\mathcal{P}_{ij},\ \ \forall\, i,j\in N \tag{6.23}$$

$$\sum_{e\in P} w(e,f) \geq \psi_{ijf} \quad \forall\, P\in\mathcal{P}_{ij},\ \ \forall\, i,j,f\in N \tag{6.24}$$

$$\sum_{i\notin\{k,f\}} R_i\psi_{ikf} + \sum_{j\notin\{k,f\}} C_j\psi_{kjf} \geq \sigma_f \quad \forall\, k,f\in N \tag{6.25}$$

$$\sum_{i\neq f} R_i\pi_{if} + \sum_{j\neq f} C_j\pi_{fj} + \sigma_f \geq 1 \quad \forall\, f\in N \tag{6.26}$$

$$w(e,f) \geq 0 \quad \forall\, e\in E,\ \ \forall\, f\in N \tag{6.27}$$

Because of the nature of constraints (6.26), we can assume that the variables π_{ij} attain the maximum possible value given by constraints (6.23) in an optimal solution. Similarly, the variables ψ_{ijf} and σ_f attain the maximum possible values given by constraints (6.24) and (6.25) respectively in an optimal solution. Then, we have

$$\pi_{ij} = \min_{P\in\mathcal{P}_{ij}} \sum_{e\in P}\sum_{f\in N} w(e,f) \quad \forall\, i,j\in N \tag{6.28}$$

$$\psi_{ijf} = \min_{P\in\mathcal{P}_{ij}} \sum_{e\in P} w(e,f) \quad \forall\, i,j,f\in N \tag{6.29}$$

$$\sigma_f = \min_{k\in N, k\neq f} \left(\sum_{i\notin\{k,f\}} R_i\psi_{ikf} + \sum_{j\notin\{k,f\}} C_j\psi_{kjf} \right) \quad \forall\, f\in N \tag{6.30}$$

After removal of the dual variables π_{ij}, ψ_{ijf}, and σ_f using equations (6.28)-(6.30), the dual problem can be written as:

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

subject to

$$\sum_{i \in N, i \neq f} R_i \min_{P \in \mathcal{P}_{if}} \sum_{e \in P} \sum_{f' \in N} w(e,f') \;+\; \sum_{j \in N, j \neq f} C_j \min_{P \in \mathcal{P}_{fj}} \sum_{e \in P} \sum_{f' \in N} w(e,f') +$$
$$\sum_{i \in N, i \notin \{k,f\}} R_i \min_{P \in \mathcal{P}_{ik}} \sum_{e \in P} w(e,f) \;+\; \sum_{j \in N, j \notin \{k,f\}} C_j \min_{P \in \mathcal{P}_{kj}} \sum_{e \in P} w(e,f) \;\geq\; 1$$
$$\forall\, k,f \in N, k \neq f \qquad (6.31)$$

$$w(e,f) \;\geq\; 0 \;\; \forall\, e \in E, \;\; \forall\, f \in N \qquad (6.32)$$

6.4.4 Combinatorial Algorithm

In this section, we use the primal and dual formulations of the path flow based linear program to develop a fast combinatorial algorithm for the problem.

For given $k, f \in N, k \neq f$ and weights $w(e,f)$, let $U(k,f)$ denote the LHS of dual constraint (6.31). Given any set of weights $w(e,f)$, we show how $U(k,f)$ can be computed in polynomial time for all $k, f \in N, k \neq f$ using simple shortest path computations. Note that $\min_{P \in \mathcal{P}_{ij}} \sum_{e \in P} \sum_{f' \in N} w(e,f')$ is the cost of the shortest path from node i to node j under link costs $c(e) = \sum_{f' \in N} w(e,f)$ for all $e \in E$. Hence, the quantities $\min_{P \in \mathcal{P}_{ij}} \sum_{e \in P} \sum_{f' \in N} w(e,f')$ for all $i, j \in N$ can be computed using a single all-pairs shortest path computation. Similarly, $\min_{P \in \mathcal{P}_{ij}} \sum_{e \in P} w(e,f)$ is the cost of the shortest path from node i to node j under link costs $c(e) = w(e,f)$ for all $e \in E$. Hence, the quantity $\min_{P \in \mathcal{P}_{ij}} \sum_{e \in P} w(e,f)$ for all $i, j, f \in N$ can be computed using n all-pairs shortest path computation, one associated with each node $f \in N$. Using these computed values, the LHS of (6.31), $U(k,f)$, for all $k, f \in N, k \neq f$, can be computed in polynomial time.

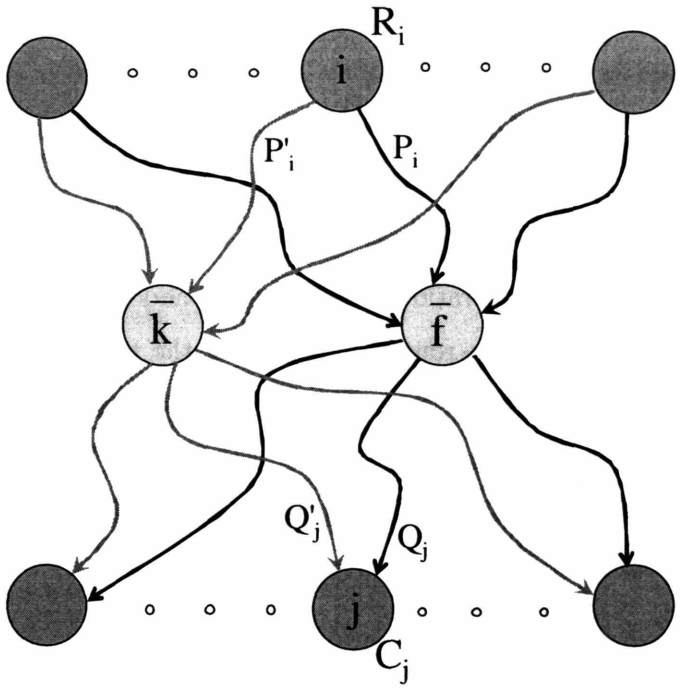


Figure 6-1: One Step in the Primal-Dual Computation for Failure Dependent Provisioning.

The algorithm works as follows. Start with initial weights $w(e,f) = \frac{\delta}{u_e} \quad \forall\, e \in E, \quad \forall\, f \in N$ (the quantity δ depends on ϵ and is derived later). Repeat the following until the dual objective function value becomes greater than 1:

1. Compute nodes $f = \bar{f}$ and $k = \bar{k}$ for which $U(k,f)$ is minimum. This also identifies (i) paths P_i from node i to node $\bar{f}$ for all $i \neq \bar{f}$, (ii) paths Q_j from node $\bar{f}$ to node j for all $j \neq \bar{f}$, (iii) paths P_i' from node i to node $\bar{k}$ for all $i \notin \{\bar{k}, \bar{f}\}$, and (iv) paths Q_j' from node $\bar{k}$ to node j for all $j \notin \{\bar{k}, \bar{f}\}$. (These are the corresponding shortest paths used in evaluating $U(\bar{k}, \bar{f})$ as described above.) This is illustrated in Figure 6-1.

2. For a traffic split ratio of 1 for intermediate node $\bar{f}$, the traffic on path P_i is R_i for all $i \neq \bar{f}$ and the traffic on path Q_j is C_j for all $j \neq \bar{f}$. Using this, compute

the traffic $s(e)$ on link e under normal (no-failure) conditions per unit split ratio $\alpha_{\bar{f}}$ for intermediate node $\bar{f}$ as

$$s(e) = \sum_{i \neq \bar{f}, P_i \ni e} R_i + \sum_{j \neq \bar{f}, Q_j \ni e} C_j \quad \forall\, e \in E \tag{6.33}$$

3. For every unit of traffic split ratio for intermediate node $\bar{f}$ that is redistributed to intermediate node $\bar{k}$ when router node $\bar{f}$ fails, the traffic on path P_i' is R_i for all $i \notin \{\bar{k}, \bar{f}\}$ and the traffic on path Q_j' is C_j for all $j \notin \{\bar{k}, \bar{f}\}$. Using this, compute the traffic $s'(e, f)$ that appears on link e after failure of router node $\bar{f}$ per unit split ratio $\alpha_{\bar{f}}$ of intermediate node $\bar{f}$ that is redistributed to intermediate node $\bar{k}$ as

$$s'(e) = \sum_{i \notin \{\bar{k}, \bar{f}\}, P_i' \ni e} R_i + \sum_{j \notin \{\bar{k}, \bar{f}\}, Q_j' \ni e} C_j \quad \forall\, e \in E \tag{6.34}$$

4. Compute the maximum value α for the traffic split ratio for intermediate node $\bar{f}$ that is redistributed to intermediate node $\bar{k}$ after failure of router node $\bar{f}$ such that for every link e, the sum of working flow and restoration flow due to failure of router node $\bar{f}$ *sent during this iteration* is at most the link capacity u_e. The working flow uses intermediate node $\bar{f}$ and is sent along paths P_i, Q_j and the restoration flow uses intermediate node $\bar{k}$ and is sent along paths P_i', Q_j'. The maximum value of α is given by

$$\alpha = \min_{e \in E} \frac{u_e}{s(e) + s'(e)} \tag{6.35}$$

5. For this value α of the traffic split ratio for intermediate node $\bar{f}$, send αR_i amount of working flow on path P_i for all $i \neq \bar{f}$ and αC_j amount of working flow on path Q_j for all $j \neq \bar{f}$. Compute the total working flow on link e as $\Delta(e) = \alpha s(e)$ for all $e \in E$.

6. The traffic split ratio α for intermediate node $\bar{f}$ is redistributed to intermediate node $\bar{k}$ after failure of router node $\bar{f}$. Send αR_i amount of restoration flow on path P_i' for all $i \notin \{\bar{k}, \bar{f}\}$ and αC_j amount of restoration flow on path Q_j' for all $j \notin \{\bar{k}, \bar{f}\}$. Compute the total restoration flow that appears on link e after failure of router node $\bar{f}$ as $\Delta'(e, \bar{f}) = \alpha s'(e)$ for all $e \in E$.

7. For each $e \in E$, update weights $w(e, \bar{f})$ as

$$w(e, \bar{f}) \leftarrow w(e, \bar{f}) \left(1 + \frac{\epsilon[\Delta(e) + \Delta'(e, \bar{f})]}{u_e}\right)$$

8. For each $e \in E$, $f \in N, f \neq \bar{f}$, update weights $w(e, f)$ as

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

9. Increment by α both the traffic split ratio $\alpha_{\bar{f}}$ associated with intermediate node $\bar{f}$ and the redistribution ratio $\beta_{\bar{k}\bar{f}}$ to intermediate node $\bar{k}$ after failure of router node $\bar{f}$.

When the above procedure terminates, primal link capacity constraints 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 working and restoration flows and traffic split ratios α_i, β_{jf} 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.

The pseudo-code for the above procedure, called Algorithm FDP (for Failure Dependent Provisioning), is provided below. 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 router node f as the algorithm progresses. The variable

D is initialized to 0 and remains less than 1 as long as the dual objective function value is less than 1. After the **while** loop terminates (through a jump out of it), the factor by which the capacity constraint on each link e gets violated is computed into array $scale(e)$. Finally, the α_i and β_{jf} values are divided by the maximum capacity violation factor and the resulting values are output.

Algorithm FDP:

$\alpha_k \leftarrow 0 \quad \forall\, k \in N$;

$\beta_{kf} \leftarrow 0 \quad \forall\, k, f \in N, k \neq f$;

$w(e,f) \leftarrow \frac{\delta}{u_e} \quad \forall\, e \in E, f \in N$;

$work(e) \leftarrow 0 \quad \forall\, e \in E$;

$bkp(e,f) \leftarrow 0 \quad \forall\, e \in E, f \in N$;

$D \leftarrow 0$;

while $D < 1$ **do**

For each $i, j \in N$, compute shortest path from i to j under link costs $c(e) = \sum_{f \in N} w(e,f)$ and denote its cost by $SP(i,j)$;

For each $i, j \in N$, compute shortest path from i to j under link costs $c(e) = w(e,f)$ for each $f \in N$ and denote its cost by $SP_f(i,j)$;

$U(k,f) \leftarrow \sum_{i \neq k} R_i SP(i,k) + \sum_{j \neq k} C_j SP(k,j) +$
$\sum_{i \notin \{k,f\}} R_i SP_f(i,k) + \sum_{j \notin \{k,f\}} C_j SP_f(k,j) \quad \forall\, k, f \in N, k \neq f$;

$(\bar{k}, \bar{f}) \leftarrow \arg\min_{k,f \in N, k \neq f} U(k,f)$;

(This identifies paths P_i, Q_j, P'_i, and Q'_j as defined earlier.)

$s(e) = \sum_{i \neq \bar{f}, P_i \ni e} R_i + \sum_{j \neq \bar{f}, Q_j \ni e} C_j \quad \forall\, e \in E$;

$s'(e) = \sum_{i \notin \{\bar{k}, \bar{f}\}, P'_i \ni e} R_i + \sum_{j \notin \{\bar{k}, \bar{f}\}, Q'_j \ni e} C_j \quad \forall\, e \in E$;

$\alpha \leftarrow \min_{e \in E} \frac{u_e}{s(e) + s'(e)}$;

$\Delta(e) \leftarrow \alpha s(e) \quad \forall\, e \in E$;

$\Delta'(e, \bar{f}) \leftarrow \alpha s'(e) \quad \forall\, e \in E$;

$work(e) \leftarrow work(e) + \Delta(e) \quad \forall\, e \in E$;

$bkp(e, \bar{f}) \leftarrow bkp(e, \bar{f}) + \Delta'(e, \bar{f}) \ \ \forall\, e \in E$;

$w(e, f) \leftarrow w(e, f)(1 + \frac{\epsilon\Delta(e)}{u_e}) \ \ \forall\, e \in E, \ \ \forall\, f \in N, f \neq \bar{f}$;

$w(e, \bar{f}) \leftarrow w(e, \bar{f})(1 + \frac{\epsilon[\Delta(e)+\Delta'(e,\bar{f})]}{u_e}) \ \ \forall\, e \in E$;

$\alpha_{\bar{f}} \leftarrow \alpha_{\bar{f}} + \alpha$;

$\beta_{\bar{k}\bar{f}} \leftarrow \beta_{\bar{k}\bar{f}} + \alpha$;

$D \leftarrow \sum_{e\in E} \sum_{f\in N} u_e w(e, f)$;

end while

$bkp\_max(e) \leftarrow \max_{f\in N} bkp(e, f) \ \ \forall\, e \in E$;

$scale(e) \leftarrow \frac{work(e)+bkp\_max(e)}{u_e} \ \ \forall\, e \in E$;

$scale\_max \leftarrow \max_{e\in E} scale(e)$;

$\alpha_k \leftarrow \frac{\alpha_k}{scale\_max}$ for all $k \in N$;

$\beta_{kf} \leftarrow \frac{\beta_{kf}}{scale\_max}$ for all $k, f \in N, k \neq f$;

Output traffic split ratios α_k and failure redistribution ratios β_{kf} ;

We next analyze the approximation guarantee and running time of Algorithm MAX_LAMBDA.

Analysis of Approximation Guarantee

The analysis follows the same approach as that of the strongly polynomial time algorithm for maximum throughput two-phase routing for the unprotected case in Section 4.3.5.

Given a set of dual weights $w(e, f)$, let $D(w)$ denote the dual objective function value and let $\Gamma(w)$ denote the minimum value of the LHS of dual program constraint (6.31) over all nodes $k, f \in N, k \neq f$. Then, solving the dual program is equivalent to finding a set of weights $w(e, f)$ such that $\frac{D(w)}{\Gamma(w)}$ is minimized. Denote the optimal objective function value of the latter by θ, i.e., $\theta = \min_w \frac{D(w)}{\Gamma(w)}$. Let w_{t-1} denote the weight function at the beginning of iteration t of the **while** loop, and let A_{t-1} be the value of $\sum_{j\in N} \alpha_j$ (primal objective function) up to the end of iteration $t-1$. Suppose

the algorithm terminates after iteration L.

Lemma 6.4.1 *At the end of every iteration t, $1 \leq t \leq L$, of Algorithm FDP, the following holds*

$$D(w_t) \leq mn\delta \prod_{j=1}^{t} [1 + \frac{\epsilon}{\theta}(A_j - A_{j-1})]$$

Proof: Let $\bar{f}, \bar{k} \in N$ be the nodes for which LHS of dual constraint (6.31) is minimum and let P_i, Q_j, P'_i, Q'_j be the corresponding paths (as defined earlier) along which flow is augmented during iteration t. Recall that the weights are updated as:

$$\begin{aligned} w_t(e,f) &\leftarrow w_{t-1}(e,f)\left(1 + \frac{\epsilon\Delta(e)}{u_e}\right) \quad \forall\, e \in E, \quad \forall\, f \in N, f \neq \bar{f} \\ w_t(e,\bar{f}) &\leftarrow w_{t-1}(e,\bar{f})\left(1 + \frac{\epsilon[\Delta(e) + \Delta'(e,\bar{f})]}{u_e}\right) \quad \forall\, e \in E \end{aligned}$$

where $\Delta(e)$ is the total working flow on link e, and $\Delta'(e,\bar{f})$ is the total restoration flow on link e due to failure of router node $\bar{f}$ (both sent during iteration t). Using this, we have

$$\begin{aligned} D(w_t) &= \sum_{e\in E, f\in N} u_e w_t(e,f) \\ &= \sum_{e\in E, f\in N} u_e w_{t-1}(e,f) + \epsilon \sum_{e\in E, f\neq \bar{f}} w_{t-1}(e,f)\Delta(e) + \\ &\quad \epsilon \sum_{e\in E} w_{t-1}(e,\bar{f})[\Delta(e) + \Delta'(e,\bar{f})] \\ &= D(w_{t-1}) + \epsilon \sum_{e\in E, f\in N} w_{t-1}(e,f)\Delta(e) + \epsilon \sum_{e\in E} w_{t-1}(e,\bar{f})\Delta'(e,\bar{f}) \\ &= D(w_{t-1}) + \epsilon \sum_{e\in E, f\in N} w_{t-1}(e,f)[\sum_{i\neq\bar{f}, P_i \ni e} \alpha R_i + \sum_{j\neq\bar{f}, Q_j\ni e} \alpha C_j] + \\ &\quad \epsilon \sum_{e\in E} w_{t-1}(e,\bar{f})[\sum_{i\notin\{\bar{k},\bar{f}\}, P'_i\ni e} \alpha R_i + \sum_{j\notin\{\bar{k},\bar{f}\}, Q'_j\ni e} \alpha C_j] \\ &= D(w_{t-1}) + \epsilon\alpha \sum_{e\in E, f\in N} w_{t-1}(e,f)[\sum_{i\neq\bar{f}, P_i\ni e} R_i + \sum_{j\neq\bar{f}, Q_j\ni e} C_j] + \\ &\quad \epsilon\alpha \sum_{e\in E} w_{t-1}(e,\bar{f})[\sum_{i\notin\{\bar{k},\bar{f}\}, P'_i\ni e} R_i + \sum_{j\notin\{\bar{k},\bar{f}\}, Q'_j\ni e} C_j] \end{aligned}$$

We interchange the order of summation in the second and third terms on RHS of the above equation and first sum along links on paths P_i, Q_j, and then over i, j respectively. Similarly, for the third term, we interchange the order of summation and first sum along links on paths P'_i, Q'_j, and then over i, j respectively. This gives

$$\begin{aligned} D(w_t) &= D(w_{t-1}) + \epsilon\alpha[\sum_{i \neq \bar{f}} R_i \sum_{e \in P_i, f \in N} w_{t-1}(e,f) + \sum_{j \neq \bar{f}} C_j \sum_{e \in Q_j, f \in N} w_{t-1}(e,f) + \\ &\quad \sum_{i \notin \{\bar{k}, \bar{f}\}} R_i \sum_{e \in P'_i} w_{t-1}(e,\bar{f}) + \sum_{j \notin \{\bar{k}, \bar{f}\}} C_j \sum_{e \in Q'_j} w_{t-1}(e,\bar{f})] \\ &= D(w_{t-1}) + \epsilon\alpha U(\bar{k}, \bar{f}) \qquad (6.36) \\ &= D(w_{i-1}) + \epsilon\alpha\Gamma(w_{i-1}) \\ &= D(w_{i-1}) + \epsilon(A_t - A_{t-1})\Gamma(w_{i-1}) \end{aligned}$$

The step leading to (6.36) follows from the choice of nodes $\bar{k}, \bar{f}$ and associated paths P_i, Q_j, P'_i, Q'_j that minimize $U(k,f)$ for the set of weights $w_{t-1}(e,f)$ at the beginning of iteration t.

Using this for each iteration down to the first one, we have

$$D(w_t) = D(w_0) + \epsilon \sum_{j=1}^{t} (A_j - A_{j-1})\Gamma(w_{j-1}) \qquad (6.37)$$

From the definition of θ, we have $\theta \leq \frac{D(w_{j-1})}{\Gamma(w_{j-1})}$, whence $\Gamma(w_{j-1}) \leq \frac{1}{\theta} D(w_{j-1})$. Also, $D(w_0) = mn\delta$. Using these in equation (6.37), we have

$$D(w_t) \leq mn\delta + \frac{\epsilon}{\theta} \sum_{j=1}^{t} (A_j - A_{j-1}) D(w_{j-1}) \qquad (6.38)$$

The property claimed in the lemma can now be proved using inequality (6.38) and mathematical induction on the iteration number t. The method is similar to that used in the proof of Lemma 4.3.1. ■

We now estimate the factor by which the objective function value value $A_{I,}$ in the primal solution needs to be scaled when the algorithm terminates so as to ensure that

link capacity constraints are not violated.

Lemma 6.4.2 *When Algorithm FDP terminates, the primal solution needs to be scaled by a factor of at most* $\log_{1+\epsilon} \frac{1+\epsilon}{\delta}$ *to ensure primal feasibility.*

Proof: Consider any link e and router failure at some node $f \in N$. We will show that the working flow on link e plus the restoration flow on link e due to failure of router node f is at most u_e when the primal solution is scaled by the above factor.

The value of $w(e, f)$ changes (due to an update) when flow is augmented on edge e under either or both of the following circumstances:

- Link e appears on any of the paths P_i, Q_j, in which case the flow is working traffic on this link, or
- Link e appears on any of the paths P_i', Q_j', in which case the flow appears as restoration traffic on link e after failure of router node f.

Let the sequence of flow augmentations (working plus restoration) on link e that *require update of weight* $w(e, f)$ be $\Delta_1, \Delta_2, \ldots, \Delta_r$, where $r \leq L$. Let $\sum_{t=1}^{r} \Delta_t = \kappa u_e$, i.e., the total flow (working traffic plus restoration traffic after failure of router node f) routed on link e exceeds its capacity by a factor of κ.

Because of the way in which α is chosen in accordance with equations (6.33)-(6.35), we have $\Delta_i \leq u_e$ for all i. Hence, dual weights are updated by a factor of at most $1 + \epsilon$ after each iteration. Since the algorithm terminates when $D(w) \geq 1$, and since dual weights are updated by a factor of at most $1 + \epsilon$ after each iteration, we have $D(w_L) < 1 + \epsilon$. Since the weight $w(e, f)$, with coefficient u_e, is one of the summing components of $D(w)$, we have $u_e w_L(e, f) < 1 + \epsilon$. Also, the value of $w_L(e, f)$ is given by

$$w_L(e, f) = \frac{\delta}{u_e} \prod_{t=1}^{r} (1 + \frac{\Delta_t}{u_e} \epsilon)$$

Using the inequality $(1+cx) \geq (1+x)^c \ \forall\ x \geq 0$ and any $0 \leq c \leq 1$ and setting $x = \epsilon$ and $c = \frac{\Delta_t}{u_e} \leq 1$, we have

$$\begin{aligned} \frac{1+\epsilon}{u_e} \ > \ w_L(e,f) \ &\geq \ \frac{\delta}{u_e} \prod_{t=1}^{r} (1+\epsilon)^{\Delta_t/u_e} \\ &= \ \frac{\delta}{u_e} (1+\epsilon)^{\sum_{t=1}^{r} \Delta_t/u_e} \\ &= \ \frac{\delta}{u_e} (1+\epsilon)^{\kappa} \end{aligned}$$

whence,

$$\kappa < \log_{1+\epsilon} \frac{1+\epsilon}{\delta}$$

■

The values of ϵ and δ are related, in the following theorem, to the approximation factor guarantee of Algorithm FDP.

Theorem 6.4.3 *For any given $0 < \epsilon' \leq 0.5$, Algorithm FDP computes a solution with objective function value within $(1+\epsilon')$-factor of the optimum for*

$$\delta = \frac{1+\epsilon}{[(1+\epsilon)m]^{1/\epsilon}} \quad \textit{and} \quad \epsilon = \frac{\epsilon'}{2}$$

Proof: Using Lemma 6.4.1 and the inequality $1 + x \leq e^x$ for all $x \geq 0$, we have

$$\begin{aligned} D(w_t) \ &\leq \ mn\delta \prod_{j=1}^{t} e^{\frac{\epsilon}{\theta}(A_j - A_{j-1})} \\ &= \ mn\delta e^{\epsilon A_t/\theta} \end{aligned}$$

The simplification in the above step uses telescopic cancellation of the sum $(A_j - A_{j-1})$ over j. Since the algorithm terminates after iteration L, we must have $D(w_L) \geq 1$. Thus,

$$1 \leq D(w_L) \leq mn\delta e^{\epsilon A_L/\theta}$$

whence,

$$\frac{\theta}{A_L} \leq \frac{\epsilon}{\ln(\frac{1}{mn\delta}} \tag{6.39}$$

From Lemma 6.4.2, the objective function value of the feasible primal solution after scaling is at least

$$\frac{A_L}{\log_{1+\epsilon}\frac{1+\epsilon}{\delta}}$$

The approximation factor for the primal solution is at most the gap (ratio) between the dual and primal solutions. Using the lower bound on the primal solution and inequality (6.39), this is at most

$$\begin{aligned}\frac{\theta}{\frac{A_L}{\log_{1+\epsilon}\frac{1+\epsilon}{\delta}}} &\leq \frac{\epsilon \log_{1+\epsilon}\frac{1+\epsilon}{\delta}}{\ln\frac{1}{mn\delta}} \\ &= \frac{\epsilon}{\ln(1+\epsilon)}\frac{\ln\frac{1+\epsilon}{\delta}}{\ln\frac{1}{mn\delta}}\end{aligned}$$

The quantity $\ln\frac{1+\epsilon}{\delta}/\ln\frac{1}{mn\delta}$ equals $\frac{1}{1-\epsilon}$ for $\delta = (1+\epsilon)/[(1+\epsilon)nm]^{1/\epsilon}$. Using this value of δ, the approximation factor is upper bounded by $\frac{\epsilon}{(1-\epsilon)\ln(1+\epsilon)}$. This is at most $1+2\epsilon$ for $\epsilon \leq 0.25$. Setting $\epsilon = \frac{\epsilon'}{2}$, we get the desired approximation ratio of $1+\epsilon'$. ■

Analysis of Running Time

We show that the running time of Algorithm FDP is strongly polynomial.

Theorem 6.4.4 *For any given $\epsilon > 0$ chosen to provide the desired approximation factor guarantee in accordance with Theorem 6.4.3, Algorithm FDP runs in time*

$$O\left(\frac{1}{\epsilon^2}n^3m(m+n\log n)\log nm\right)$$

which is strongly polynomial.

Proof: We first consider the running time of each iteration of the algorithm during which nodes $\bar{f}, \bar{k}$ and associated paths P_i, Q_j, P_i', Q_j' are chosen to augment flow. Computation of the minimum value of $U(k, f)$ over all $k, f \in N, k \neq f$ involves n all-pairs shortest path computations which can be implemented in $O(n^2m+n^3 \log n)$ time using Dijkstra's shortest path algorithm with Fibonacci heaps [AMO93]. It can be verified that all other operations within an iteration are absorbed by the time taken for these n all-pairs shortest path computations, leading to a total of $O(n^2(m+n \log n))$ time per iteration.

We next estimate the number of iterations before the algorithm terminates. Recall that in each iteration, flow is augmented along paths P_i, Q_j, P_i', Q_j' corresponding to the maximum value of intermediate node split ratio α such that the working flow $\Delta(e)$ plus the restoration flow $\Delta'(e, \bar{f})$ sent on link e during that iteration is at most u_e. Thus, for at least one link e, the total flow sent equals u_e and the weight $w(e, \bar{f})$ increases by a factor of $1 + \epsilon$. Accordingly, with each iteration, we can associate a weight $w(e, f)$ which increases by a factor of $1 + \epsilon$.

Consider the weight $w(e, f)$ for fixed $e \in E, f \in N$. Since $w_0(e, f) = \frac{\delta}{u_e}$ and $w_L(e, f) < \frac{1+\epsilon}{u_e}$ (as deduced in the proof of Lemma 6.4.2), the maximum number of times that this weight can be associated with any iteration is

$$\log_{1+\epsilon} \frac{1+\epsilon}{\delta} = \frac{1}{\epsilon}(1 + \log_{1+\epsilon} nm) = O(\frac{1}{\epsilon} \log_{1+\epsilon} nm)$$

Since there are a total of nm weights $w(e, f)$, hence the total number of iterations is upper bounded by $O(\frac{nm}{\epsilon} \log_{1+\epsilon} nm)$. Multiplying this by the running time per iteration, we obtain the overall algorithm running time as $O(\frac{1}{\epsilon} n^3 m(m+n \log n) \log_{1+\epsilon} nm)$. Since $\ln(1 + \epsilon) = \Theta(\epsilon)$, this is $O(\frac{1}{\epsilon^2} n^3 m(m + n \log n) \log nm)$. ■

Topology	Routers (original)	Links (inter-router)	PoPs (coalesced)	Links (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 6.1: Rocketfuel topologies with AS number and name. The table lists the original number of routers and inter-router links, and the number of coalesced PoPs and inter-PoP links.

6.5 Evaluation on ISP Topologies

In this section, we evaluate the throughput performance of the two schemes for protecting against router node failures in two-phase routing. In order to make exact comparisons with the throughput for the unprotected case, we compute the throughput for both the schemes using the link flow based linear programming formulations developed in this chapter. For the throughput of the unprotected scheme, we use the linear programming formulation from Section 4.1.1. We use CPLEX [CPLEX] to solve all linear programs.

6.5.1 Topologies and Link/Ingress-Egress Capacities

We use the six ISP maps from the Rocketfuel dataset which had accompanying (deduced) OSPF/IS-IS weights [SMWH, SMW02, MSWA02]. These topologies list multiple intra-PoP (Point of Presence) routers and/or multiple intra-city PoPs as individual nodes. We coalesced such nodes so that nodes correspond to cities and the topology represents geographical PoP-to-PoP ISP topologies. Some data about the original topologies and their coalesced versions is listed in Table 6.1.

The Rocketfuel topologies are router-level (IP layer) topologies. The PoP-to-PoP topologies we obtained as above all have average node degrees less than 4. Physical

WDM topologies of ISPs are characterized by small average node degrees (typically less than 4). Assuming that all physical WDM links appear in the IP topology, it is conceivable that "most" of the links in the PoP-to-PoP Rocketfuel topologies correspond to physical WDM links (instead of multi-hop physical layer paths). We make this assumption for our experiments with IP-over-Optical networks. ISPs regard their topologies as proprietary information – we are not aware of other credible sources for information about actual ISP topologies.

The topologies provided by Rocketfuel did not include the capacities of the links, which were needed for our study. The Rocketfuel maps did include derived OSPF/ISIS weights of links, which were computed to match observed routes. In the absence of any other information on capacities, we need a way to deduce the link capacities from the weights. For this purpose, we assumed that the given link weights are the Cisco default setting for OSPF weights, i.e., inversely-proportional to the link capacities [Cisco97]. The link capacities obtained in this manner turned out to be symmetric, i.e., $u_{ij} = u_{ji}$ for all $(i, j) \in E$.

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 $R_i (= C_i) \propto \sum_{e \in E^+(i)} u_e$.

6.5.2 Experiments and Results

We denote the throughput values for the unprotected and router node failure protected versions of two-phase routing as follows: (i) λ_{UNP} for unprotected, (ii) λ_{FIP} for protecting router node failures with failure independent provisioning, and (iii) λ_{FDP} for protecting router node failures with failure dependent provisioning. Clearly,

Topology	$\frac{\lambda_{FIP}}{\lambda_{UNP}}$	$\frac{\lambda_{FDP}}{\lambda_{UNP}}$	$\frac{n-1}{n}$	Closeness of $\frac{\lambda_{FIP}}{\lambda_{UNP}}$ to $\frac{n-1}{n}$	$\frac{\lambda_{FDP}}{\lambda_{FIP}}$
Telstra (Australia) 1221	0.9744	0.9750	0.9825	0.82%	1.0006
Sprintlink (US) 1239	0.9683	0.9294*	0.9773	0.92%	0.9598*
Ebone (Europe) 1755	0.9500	0.9524	0.9565	0.68%	1.0025
Tiscali (Europe) 3257	0.9677	0.9293*	0.9800	1.26%	0.9603*
Exodus (Europe) 3967	0.9375	0.9412	0.9545	1.79%	1.0039
Abovenet (US) 6461	0.9369	0.9448	0.9545	1.85%	1.0084

Table 6.2: Throughput of Two-Phase Routing for failure independent (λ_{FIP}) and failure dependent (λ_{FDP}) provisioning schemes for protecting against router node failures compared to unprotected case (λ_{UNP}).

$\lambda_{UNP} > \lambda_{FDP} \geq \lambda_{FIP}$ (the last inequality follows from the nature of failure dependent provisioning as explained in Section 6.2.2). We are also interested in the number of intermediate nodes i with non-zero traffic split ratios α_i (under normal no-failure conditions), which we denote for the three cases by N_{UNP}, N_{FIP}, and N_{FDP} respectively.

For the Sprintlink 1239 and Tiscali 3257 topologies, the CPLEX processes for solving the linear program for failure dependent provisioning ran out of memory and were killed on a 2.4GHz Dual Xeon machine with 1GB of RAM and running Linux. This was the fastest machine with the highest RAM that we had access to for running CPLEX. Hence, for these two topologies, we used the combinatorial algorithm to obtain solutions within 5% of optimality. The corresponding entries for the two topologies are marked with an asterisk (*) in Tables 6.2 and 6.3. For these two topologies, the value of λ_{FDP}, computed approximately as described above, turned out to be less than λ_{FIP}.

Throughput

In Table 6.2, we list the relative increase in throughput of two-phase routing, compared to the unprotected case, for the two schemes for protecting against router node failures for the six Rocketfuel topologies. We also list the closeness of the ratio

Topology	O_{FIP}	O_{FDP}
Telstra (Australia) 1221	2.56%	2.50%
Sprintlink (US) 1239	3.17%	7.06%*
Ebone (Europe) 1755	5.00%	4.76%
Tiscali (Europe) 3257	3.23%	7.07%*
Exodus (Europe) 3967	6.25%	5.88%
Abovenet (US) 6461	6.31%	5.52%

Table 6.3: Overhead of failure independent (O_{FIP}) and failure dependent (O_{FDP}) provisioning schemes for protecting against router node failures compared to unprotected case for Two-Phase Routing.

$\frac{\lambda_{FIP}}{\lambda_{UNP}}$ to the theoretical upper bound of $\frac{n-1}{n}$ – in all cases, this is less than 2%. The throughput of failure dependent provisioning (λ_{FDP}) is higher than that of failure independent provisioning (λ_{FIP}) by less than 1% for the four topologies for which we could compute λ_{FDP} exactly by solving the corresponding linear programs in CPLEX.

The overhead of protecting against router node failures can be measured by the percentage decrease in network throughput over that for the unprotected case. For failure independent provisioning, this is $O_{FIP} = \frac{\lambda_{UNP}-\lambda_{FIP}}{\lambda_{UNP}}$. For failure dependent provisioning, this is $O_{FDP} = \frac{\lambda_{UNP}-\lambda_{FDP}}{\lambda_{UNP}}$. These values are listed in Table 6.3. For both failure independent and failure dependent provisioning schemes, the overhead range is about 2-7% for the six topologies. Thus, *it is relatively inexpensive to provide resiliency against router node failures in two-phase routing.*

We also observe that the overhead of failure dependent provisioning is only marginally lower than that for failure independent provisioning (less than a percentage point) for the four topologies for which we could compute λ_{FDP} exactly. Hence, given the static optical layer provisioning property of failure independent provisioning in handling router node failures, it might be the preferred one among the two schemes.

Number of Intermediate Nodes

In Table 6.4, we list the number of intermediate nodes i with non-zero traffic split ratios (under normal no-failure conditions) for the three cases for the six Rocketfuel

Topology	N_{UNP}	N_{FIP}	N_{FDP}
Telstra (Australia) 1221	1	38	38
Sprintlink (US) 1239	5	30	24
Ebone (Europe) 1755	4	19	19
Tiscali (Europe) 3257	7	30	29
Exodus (Europe) 3967	3	16	16
Abovenet (US) 6461	7	17	18

Table 6.4: Number of Intermediate Nodes in Two-Phase Routing for unprotected (N_{UNP}), and failure independent (N_{FIP}), failure dependent (N_{FDP}) provisioning schemes for protecting against router node failures.

topologies. In our experiments with both the schemes for protecting against router node failures, we observed that some intermediate nodes have quite small α_i values – they carry less than a percentage point of total network traffic. In practice, the traffic split ratios associated with these intermediate nodes can be redistributed to other nodes without any significant decrease in throughput. Hence, in Table 6.4, for the N_{FIP} and N_{FDP} values, we plot the number of intermediate nodes with the largest α_i values (normalized) that sum to at least 0.95, i.e., the nodes with largest traffic split ratios that together carry at least 95% of total network traffic.

Interestingly, the number of intermediate nodes *increases* when we provide protection against router node failures (it is almost always the same for the failure independent and failure dependent schemes). This behavior can be attributed to the same reason for which the straightforward approach of redistributing traffic for the failed intermediate router node to other intermediate nodes in proportion to that of the traffic split ratios for the unprotected case does not lead to the best throughput. For the unprotected case, a small number of intermediate nodes is preferred – these nodes are presumably located at geographical center(s) of the network and provide the best opportunities for serving as intermediate nodes without increasing the length of end-to-end paths or decreasing throughput significantly. However, a small number of intermediates nodes is associated with a relatively large traffic split ratio for each such node. Thus, in the event of a route node failure at any of these intermediate

nodes, a relatively large fraction of the traffic needs to be restored, thus increasing the resources reserved for restoration. Hence, in an effort to maximize the throughput, the algorithms developed in this chapter intelligently spread the traffic to many intermediate nodes so as to prevent any single traffic split ratio from becoming too large.

Chapter 7

Protecting Against Link Failures

In this chapter, we consider making two-phase routing resilient to link failures through three different pre-provisioned restoration mechanisms – (i) local (link/span) restoration (LR), (ii) K-route path restoration (KPR), and (iii) shared backup path restoration (SBPR). In two-phase routing, the first and second phase paths can be protected using any of the above three restoration mechanisms so as to provide resiliency against link failures. The first mechanism reroutes traffic locally around a failure and continues to use the portion of the primary path unaffected by failure. The last two mechanisms are end-to-end (path) based and switch traffic to a diverse backup path after a failure on the primary path.

We assume that a single link failure brings down the link in both directions. This is motivated by the fact that ports (line cards) of network routers/switches are bidirectional, hence a port failure can affect traffic in both direction son the link. In all of the three restoration models that we consider, 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 chapter 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 dedicated backup bandwidth versions of

the optimization problems are simpler but lead to large restoration capacity usage.)

In local restoration, upon failure of a link, traffic is rerouted along a backup path (detour) joining the two nodes adjacent to the failed link. Detours for different links can share bandwidth on their common links. In K-route path restoration, upon failure of a link, traffic is rerouted to a backup path that is disjoint with the primary path and serves to protect against failure of any link on the primary path. Multiple disjoint primary paths between the same source-destination pair can share a disjoint backup path. In shared backup path restoration, each connection consists of a link-disjoint primary and backup path pair – two backup paths can share bandwidth on their common links if their primary paths are link disjoint.

We provide linear programming formulations and fast combinatorial algorithms with performance guarantees for maximum throughput two-phase routing with local restoration and K-route path restoration against link failures. We show that the optimization problem for maximum throughput two-phase routing with shared backup path restoration is $\mathcal{NP}$-hard. Assuming an approximation oracle for a certain disjoint paths problem (called SBPR-DISJOINT-PATHS, which is also $\mathcal{NP}$-hard) involving the dual variables of a path indexed linear programming formulation for the problem, we design a combinatorial algorithm with provable guarantees. We also provide heuristics for finding approximating solutions to the SBPR-DISJOINT-PATHS problem. We evaluate the throughput performance and number of intermediate nodes in two-phase routing with the above three restoration mechanisms on actual ISP topologies collected for the Rocketfuel project [SMWH].

7.1 Restoration Mechanisms

We introduce the three restoration models considered in this chapter, namely *local restoration*, *K-route path restoration*, and *shared backup path restoration*. The first two mechanisms have been classified under *fast restoration* in the literature because

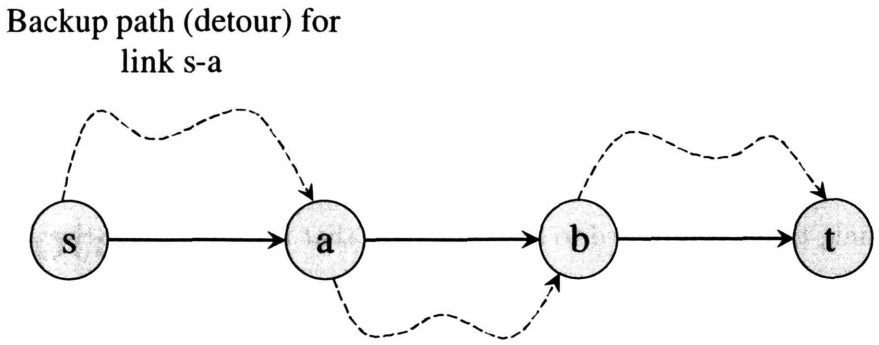


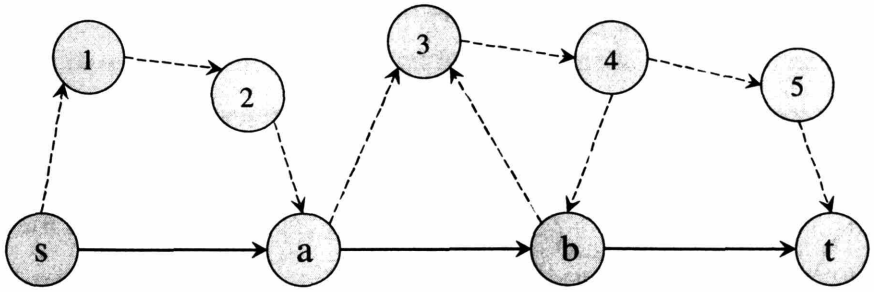
Figure 7-1: Link backup detours protecting links on primary path.

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, the Phase 1 and Phase 2 paths can be protected against link failures by any of the three restoration mechanisms. In this chapter, we develop algorithms for maximum throughput two-phase routing with resiliency against link failures provided by the described restoration mechanisms.

7.1.1 Local Restoration

For protecting against link failures with local 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 $W(P)$. A link backup detour for a link e is a (simple) path joining the two nodes adjacent to link e that does not include this link and is used to reroute the working traffic on link e when it fails. This is illustrated in Figure 7-1. 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 local restoration, it will consist of the primary path and the link backup detours for each link on the primary path. For notational convenience, in the case that link e is not on the primary path $W(P)$, the quantity $B_e(P)$ will denote the empty path (with no links).



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

Figure 7-2: Backup bandwidth sharing across link backup detours.

Under the single link failure model, backup paths for *different* links both within the same as well as across different connection(s) can share bandwidth on their common links. As illustrated in Figure 7-2, 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 detour paths, in case such signaling is required, as in optical mesh networks in order to setup cross-connects on the link detours [RS02].

7.1.2 K-Route Path Restoration

For K-route path restoration, each connection consists of K (≥ 2) link-disjoint paths from source to destination (hence the name K-route). 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.

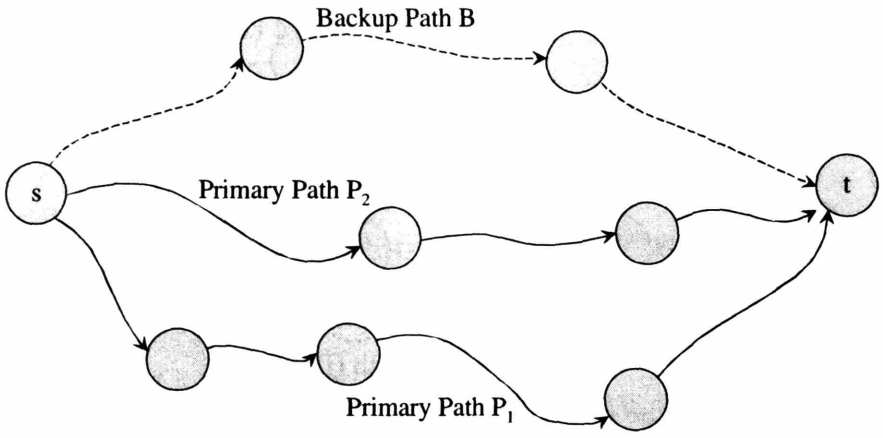


Figure 7-3: Diverse primary paths P_1, P_2 and backup path B for K-Route Path Restoration.

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 7-3. 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 backup path bandwidth for K-route path restoration is shared across the primary paths for the same connection but not with other connections. Thus, for purposes of computing bandwidth usage on a link, the bandwidth allocated for the backup path in a K-route path restored connection can be considered to be *dedicated* to that connection in the same manner as the bandwidth on the primary paths.

The fast nature of the K-route path restoration mechanism 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.

For K-route path restoration, we will use the term "path" to collectively denote the (multiple) primary path(s) and the backup path and refer to it as a "link-disjoint path set". For such a path set P, the summation over all links that belong to any individual path in P will be conveniently written as $\sum_{e \in P}$.

7.1.3 Shared Backup Path Restoration

Under shared backup path restoration, a connection consists of a primary 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 the primary is affected by a link failure. In this respect, it is similar to K-route restoration for $K = 2$. The main difference lies in the sharing of backup bandwidth across different connections in two possible ways as we explain below. Both ways of sharing backup bandwidth guarantee that all connections affected by the failure of a single link have sufficient bandwidth on their backup paths to be completely restored.

First, two backup paths can share bandwidth on their common links provided their corresponding primary paths are link-disjoint (and, hence cannot be affected simultaneously by a single link failure). This sharing of bandwidth occurs across two backup paths for all possible single link failure scenarios. This method of sharing is illustrated in Figure 7-4, where two connections, one from s_1 to t_1 and the other from s_2 to t_2, have primary paths P_1, P_2 and backup paths B_1, B_2 respectively. Since primary path P_1 and P_2 are link-disjoint, hence their backup paths B_1 and B_2 can share bandwidth on link a-b.

Second, if the primary paths of two connections have a link f in common, then

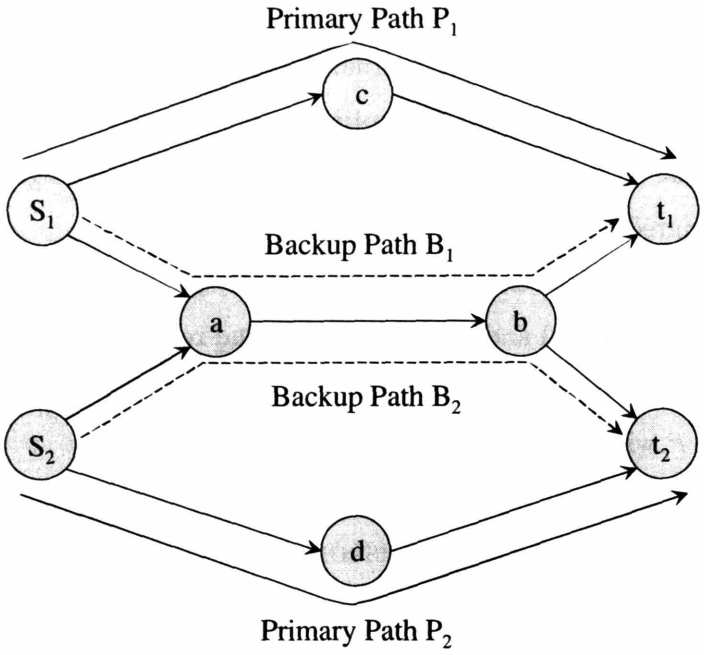


Figure 7-4: Bandwidth sharing between backup paths in Shared Backup Path Restoration.

the primary path of one connection and the backup path of another (and, vice versa) can share bandwidth (on common links) under failure of link f. This is because after failure of link f, there will be no traffic on the primary path of either of the connections. This sharing of bandwidth occurs across a primary path and a backup path (of two different connections) for specific single link failure scenarios (i.e., those links common to both primary paths).

7.2 Additional Notation

We assume that a single link failure brings down the link in both directions. This is motivated by the fact that ports (line cards) of network routers/switches are bidirectional, hence a port failure can affect traffic in both direction son the link. Hence, we will refer to link failures in the undirected sense. To simplify the notation, we

use $\tilde{f}$ to denote the undirected link corresponding to link f. Let $\tilde{E}$ denote the set of undirected links $\tilde{e}$ for all $e \in E$, i.e., $\tilde{E} = \{\tilde{e} \mid e \in E\}$.

Let $\hat{e}$ denote the reverse of link e(if it exists). Thus, the failure of links f and $\hat{f}$ will be referred to by the failure of the undirected link $\tilde{f}$. A working path $W(P)$ is affected by the failure of link $\tilde{f}$ if $f \in W(P)$ or $\hat{f} \in W(P)$. We will denote the latter condition by $\tilde{f} \in W(P)$ and consider the (non-)containment of undirected links in directed paths in the undirected sense. Also, we will use $B_{\tilde{f}}(P)$ to denote the detour corresponding to link f or $\cap f$ on $W(P)$. Since a primary path $W(P)$ will not contain both a link and its reverse, this notation can be interpreted in an unambiguous manner. It can also be written as $B_{\tilde{f}}(P) = B_f(P) \cup B_{\hat{f}}(P)$.

Because we are considering link failures in both directions, we need to define link diversity of two paths to reflect this. Throughput this chapter, two paths are said to be link-disjoint if they do not have any common link *in the undirected sense*, i.e., for any link e on one path, the other path cannot contain link e or its reverse.

We assume without any loss of generality that whenever there is a link (i, j) in G, there is also the link (j, i) (we can add links with zero capacity if required to satisfy this).

7.3 Adding Local Restoration to Two-Phase Routing

In order to make two-phase routing resilient to link failures using local restoration, we add link backup detours protecting the Phase 1 and Phase 2 paths as discussed in Section 7.1.1. Given a network with link capacities u_e and constraints R_i, C_j on the ingress-egress traffic, we consider the problem of two-phase routing with local restoration so as to maximize throughput. The throughput is the maximum multiplier λ such that all matrices in $\lambda \cdot \mathcal{T}(\vec{R}, \vec{C})$ can be feasibly routed with local restoration under given link capacities.

We can express throughput in terms of the traffic split ratios in a manner similar to that for the unprotected case. Suppose we relax the requirement that the traffic split ratios α_j sum to 1 in a feasible solution of the problem. Consider the sum

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

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

7.3.1 Link Flow Based LP Formulation

Let x_e^{ij} denote the flow value on link e for routing the demand of $\alpha_j R_i + \alpha_i C_j$ from node i to node j under normal (no-failure) conditions. The total traffic on link e under normal (no-failure) conditions is $\sum_{i,j \in N} x_e^{ij}$. Let y_e^f be the flow on link e for sending restoration flow of value $\sum_{i,j \in N} x_e^{ij}$ along link detours for link f after it fails. Then, the problem of two-phase routing with local restoration so as to maximize the network throughput can be formulated as the following link flow based linear program:

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

subject to

$$\sum_{e \in E^+(k)} x_e^{ij} - \sum_{e \in E^-(k)} x_e^{ij} = \begin{cases} \alpha_j R_i + \alpha_i C_j & \text{if } k = i \\ -\alpha_j R_i - \alpha_i C_j & \text{if } k = j \\ 0 & \text{otherwise} \end{cases} \qquad \forall\, i, j, k \in N \quad (7.1)$$

$$\sum_{e\in E^+(k),e\neq f} y_e^f - \sum_{e\in E^-(k),e\neq f} y_e^f = \begin{cases} \sum_{i,j\in N} x_e^{ij} & \text{if } k=i \\ -\sum_{i,j\in N} x_e^{ij} & \text{if } k=j \\ 0 & \text{otherwise} \end{cases} \quad \forall\, k\in N, \quad \forall\, f=(i,j)\in E \tag{7.2}$$

$$\sum_{i,j\in N} x_e^{ij} + y_e^f + y_e^{\hat{f}} \leq u_e \quad \forall\, e,f\in E, \tilde{e}\neq\tilde{f} \tag{7.3}$$

$$\alpha_i \geq 0 \quad \forall\, i\in N \tag{7.4}$$

$$x_e^{ij}, y_e^f \geq 0 \quad \forall\, e,f\in E, \quad \forall\, i,j\in N \tag{7.5}$$

Constraints (7.1) correspond to the routing of $\alpha_j R_i + \alpha_i C_j$ amount of flow from node i to node j under normal (no-failure) conditions. Constraints (7.2) correspond to the routing of restoration flow along link detours for link f after it fails. Constraints (7.3) 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 (and its reverse) is at most the capacity of the link. Recall that the reverse of link f is denoted by $\hat{f}$.

By using per-source flow variables x_e^i instead of per source-destination variables x_e^{ij}, the number of x variables in the above linear program can be reduced to nm. However, the same reduction cannot be done for the y_e^f variables because of the link diversity requirement for link detours – link f cannot be used for the restoration flow for failure of link f while it can be used for the restoration flow for failure of other links emanating out of the same node. Hence, the number of variables in the above linear program is $O(m^2)$.

It is well known that running times of general linear programming based algorithms for network problems do not scale well with increasing network size. Fortunately, our problem formulation for two-phase routing with local restoration accommodates a fast combinatorial algorithm. Moreover, because the combinatorial algorithm we develop works explicitly with paths, delay constraints on the paths (both primary and link detours) can be efficiently handled within the optimization framework, as explained for the unprotected case in Section 4.3.6. In contrast, the

link flow based linear program formulation above does not deal explicitly with paths, and hence, cannot accommodate constraints like bounding path delay.

7.3.2 Path Flow Based Linear Programming Formulation

In this section, we develop a path indexed linear programming formulation for this problem. This will be subsequently used to develop the fast combinatorial algorithm in Section 7.3.4.

Let $\mathcal{PL}_{ij}$ denote the set of all paths from node i to node j, consisting of a primary path as well as link backup detours protecting each link on the primary. Let $x(P)$ denote the traffic associated with path P. Then, the problem of two-phase routing with local restoration so as to maximize the network throughput can be formulated as the following path flow based linear program:

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

subject to

$$\sum_{P \in \mathcal{PL}_{ij}} x(P) = \alpha_j R_i + \alpha_i C_j \quad \forall\, i, j \in N \tag{7.6}$$

$$\sum_{i,j \in N} \sum_{P \in \mathcal{PL}_{ij}, W(P) \ni e} x(P) + \sum_{i,j \in N} \sum_{P \in \mathcal{PL}_{ij}, B_{\tilde{f}}(P) \ni e} x(P) \leq u_e \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f} \tag{7.7}$$

$$\alpha_i \geq 0 \quad \forall\, i \in N \tag{7.8}$$

$$x(P) \geq 0 \quad \forall\, P \in \mathcal{PL}_{ij}, \forall\, i, j \in N \tag{7.9}$$

In Section 7.3.3, we consider the dual of the above linear programs. In general, a network can have an exponential number of paths (in the size of the network). Hence, the primal program can have possibly exponential number of variables and its dual can have an exponential number of constraints, and are both not suitable for fast implementation on large sized networks. The usefulness of the primal and dual

formulation is in designing a fast (polynomial time) combinatorial algorithm for the problem.

7.3.3 Dual of Path Flow Based LP Formulation

We will use a primal-dual approach to develop a fast combinatorial algorithm (FPTAS) for two-phase routing with link restoration that computes the traffic split ratios and routing of Phase 1 and Phase 2 paths (with link detours) up to $(1+\epsilon)$-factor of the optimal objective function value (maximum throughput) for any $\epsilon > 0$. The primal-dual scheme extends the approach used for maximizing throughput for the unprotected case in Section 4.3.

We begin with the dual formulation of the linear program in Section 7.3.2. The dual formulation of the linear program associates a variable a variable π_{ij} with each demand constraint in (7.6), and a non-negative variable $w(e,\tilde{f})$ with each link capacity constraint in (7.7). The dual program can be written as:

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

subject to

$$\sum_{e\in W(P)}\sum_{\tilde{f}\in\tilde{E},\tilde{f}\neq\tilde{e}} w(e,\tilde{f}) + \sum_{\tilde{f}\in W(P)}\sum_{e\in B_{\tilde{f}}(P)} w(e,\tilde{f}) \geq \pi_{ij} \quad \forall\, P\in\mathcal{PL}_{ij}, \forall\, i,j\in N \quad (7.10)$$

$$\sum_{i\in N, i\neq k} R_i\pi_{ik} + \sum_{j\in N, j\neq k} C_j\pi_{kj} \geq 1 \quad \forall\, k\in N \quad (7.11)$$

$$w(e,\tilde{f}) \geq 0 \quad \forall\, e,f\in E, \tilde{e}\neq\tilde{f} \quad (7.12)$$

Because of the nature of constraints (7.11), we can assume that the variables π_{ij} attain the maximum possible value given by constraints (7.10) in any optimal

solution. Then, we have

$$\pi_{ij} = \min_{P \in \mathcal{PL}_{ij}} \left(\sum_{e \in W(P)} \sum_{\tilde{f} \in \tilde{E}, \tilde{f} \neq \tilde{e}} w(e, \tilde{f}) + \sum_{\tilde{f} \in W(P)} \sum_{e \in B_{\tilde{f}}(P)} w(e, \tilde{f}) \right) \quad \forall\, i, j \in N \quad (7.13)$$

This allows us to eliminate the dual variables π_{ij}.

For a path P (with link detours) and given weights $w(e, \tilde{f})$, denote the value of the quantity inside the min on the RHS of (7.13) by $\Phi(w, P)$, that is,

$$\Phi(w, P) = \sum_{e \in W(P)} \sum_{\tilde{f} \in \tilde{E}, \tilde{f} \neq \tilde{e}} w(e, \tilde{f}) + \sum_{\tilde{f} \in W(P)} \sum_{e \in B_{\tilde{f}}(P)} w(e, \tilde{f})$$

With the removal of the dual variables π_{ij} and use of the new notation, the dual problem can be written as:

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

subject to

$$\sum_{i \in N, i \neq k} R_i \min_{P \in \mathcal{PL}_{ik}} \Phi(w, P) \;+\; \sum_{j \in N, j \neq k} C_j \min_{P \in \mathcal{PL}_{kj}} \Phi(w, P) \geq 1 \quad \forall\, k \in N \quad (7.14)$$

$$w(e, \tilde{f}) \;\geq\; 0 \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f} \quad (7.15)$$

7.3.4 Combinatorial Algorithm

In this section, we use the primal and dual formulations of the path flow based linear program to develop a combinatorial algorithm for the problem.

For a given node $k \in N$ and weights $w(e, tildef)$, let $V(k)$ denote the LHS of dual constraint (7.14). Given the weights $w(e, \tilde{f})$, note that $V(k)$ can be computed in polynomial time for all $k \in N$ if we can compute $\min_{P \in \mathcal{PL}_{ij}} \Phi(w, P)$ in polynomial

time for all $i, j \in N$. We show how, for given $i, j \in N$, the path $P \in \mathcal{PL}_{ij}$ that minimizes $\Phi(w, P)$ can be obtained using simple shortest path computations.

For each link $f = (i, j) \in E$, let D_f denote the set of all possible *simple* paths from node i to node j that do not contain link f, i.e., D_f is the set of all possible link detours for link f. For each link $f \in E$, denote by $g(f)$ the cost of the shortest path in D_f under link costs $c(e) = w(e, \tilde{f})$ for all $e \in E, \tilde{e} \neq \tilde{f}$. That is,

$$g(f) = \min_{P \in D_f} \sum_{e \in P} w(e, \tilde{f}) \quad \forall\, f \in E$$

Let d_{ij} denote the cost of the shortest *simple* path P_{ij} from node i to node j under links costs

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

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

Now form the path $P \in \mathcal{PL}_{ij}$ by making its primary $W(P)$ equal to P_{ij} and for each $e \in W(P)$, the link detour $B_e(P)$ equal to the simple path in D_e whose cost is equal to $g(e)$. It is easy to see that this is the path that minimizes $\Phi(w, P)$ over all $P \in \mathcal{PL}_{ij}$ and that $d_{ij} = \min_{P' \in \mathcal{PL}_{ij}} \Phi(w, P')$.

We summarize the above efficient method for computing $\min_{P \in \mathcal{PL}_{ij}} \Phi(w, P)$ for all $i, j \in N$. As we shall see, this is required in each iteration of the combinatorial algorithm for maximizing throughput.

A. For each link $f = (i, j) \in E$, compute the cost $g(f)$ of the shortest link detour from node i to node j under link costs $c(e) = w(e, \tilde{f}) \quad \forall\, e \in E, e \neq f$ and $c(f) = \infty$, using Dijkstra's algorithm [AMO93].

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

The complete path (primary plus link detours) $P \in \mathcal{PL}_{ij}$ with cost $d_{ij} = \min_{P' \in \mathcal{PL}_{ij}} \Phi(w, P')$ 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. All other operations are subsumed by the time taken for these shortest path computations. Dijkstra's shortest path algorithm can be implemented in $O(m + n \log n)$ time using Fibonacci heaps [AMO93]. Hence, this procedure can be implemented in $O(m^2 + nm \log n)$ time.

The overall algorithm works as follows. Start with initial weights $w(e, \tilde{f}) = \frac{\delta}{u_e}$ for all $e, f \in E, \tilde{e} \neq \tilde{f}$ (the quantity δ depends on ϵ and is derived later). Repeat the following until the dual objective function value is greater than 1:

1. Compute the node 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. (These are the paths with link detours between respective node pairs obtained during computation of $V(k)$ as described above.) This is illustrated in Figure 7-5.

2. For a traffic split ratio of 1 for intermediate node $\bar{k}$, the traffic on path P_i is R_i for all $i \neq \bar{k}$ and the traffic on path Q_j is C_j for all $j \neq \bar{k}$. Using this, compute the working traffic $s(e)$ on link e under normal (no-failure) conditions per unit split ratio $\alpha_{\bar{k}}$ for intermediate node $\bar{k}$ as

$$s(e) = \sum_{i \in N, i \neq \bar{k}, W(P_i) \ni e} R_i + \sum_{j \in N, j \neq \bar{k}, W(Q_j) \ni e} C_j \quad \forall\, e \in E \tag{7.16}$$

3. For the above working traffic and given link detours in paths P_i, Q_j, compute

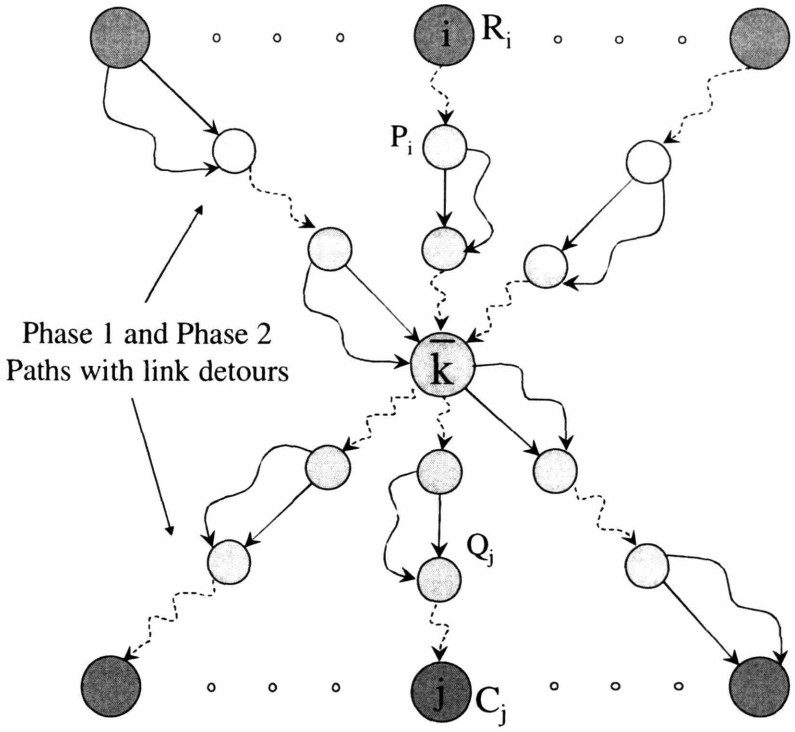


Figure 7-5: One Step in the Primal-Dual Computation for Local Restoration in Two-Phase Routing.

the restoration traffic $s'(e, \tilde{f})$ that appears on link e after failure of link $\tilde{f}$ as

$$s'(e,\tilde{f}) = \sum_{i \neq \bar{k}, W(P_i) \ni \tilde{f}, B_{\tilde{f}}(P_i) \ni e} R_i \quad + \sum_{j \neq \bar{k}, W(Q_j) \ni \tilde{f}, B_{\tilde{f}}(Q_j) \ni e} C_j$$
$$\forall\ e, f \in E, \tilde{e} \neq \tilde{f} \qquad (7.17)$$

The maximum possible traffic on link e is thus

$$s(e) + \max_{\tilde{f} \in \tilde{E}, \tilde{f} \neq \tilde{e}} s'(e, \tilde{f})$$

4. Compute the maximum value α for the traffic split ratio for intermediate $\bar{k}$ that does not lead to violation of (original) link capacity constraints for the above

traffic as

$$\alpha = \min_{e \in E} \frac{u_e}{s(e) + \max_{\tilde{f} \in \bar{E}, \tilde{f} \neq \tilde{e}} s'(e, \tilde{f})} \tag{7.18}$$

5. For this value α of the split ratio for intermediate node $\bar{k}$, send αR_i amount of traffic from node i to node $\bar{k}$ along path P_i for all $i \neq \bar{k}$ and αC_j amount of traffic from node $\bar{k}$ to node j along path Q_j for all $j \neq \bar{k}$. Compute the working traffic $\Delta(e)$ on link e under normal (no-failure) conditions as

$$\Delta(e) = \alpha s(e) \quad \forall\, e \in E$$

and the traffic $\Delta'(e, \tilde{f})$ that appears on link e after failure of any other link $\tilde{f}$ as

$$\Delta'(e, \tilde{f}) = \alpha s'(e, \tilde{f}) \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f}$$

6. Update the weights $w(e, \tilde{f})$ as follows:

$$w(e, \tilde{f}) \leftarrow w(e, \tilde{f}) \left(1 + \frac{\epsilon[\Delta(e) + \Delta'(e, \tilde{f})]}{u_e}\right) \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f}$$

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

When the above procedure terminates, primal capacity constraints 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 traffic and 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.

In the context of optical mesh networks [RS02], 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 [AMO93] in Step A of the procedure discussed at the beginning of this section to compute shortest cost paths bounded by a hop count of h.

The pseudo-code for the above procedure, called Algorithm LR, is provided below. 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 after failure of link f as the algorithm progresses. The variable D is initialized to 0 and remains less than 1 as long as the dual objective function value is less than 1. After the **while** loop terminates (through a jump out of it), 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 are output.

Algorithm LR:

$\alpha_k \leftarrow 0 \quad \forall\, k \in N$;

$w(e, \tilde{f}) \leftarrow \frac{\delta}{u_e} \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f}$;

$work(e) \leftarrow 0 \quad \forall\, e \in E$;

$bkp(e, \tilde{f}) \leftarrow 0 \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f}$;

$D \leftarrow 0$;

while $D < 1$ **do**

For each $f = (i, j) \in E$, compute shortest path from i to j that excludes link f under link costs $c(e) = w(e, \tilde{f})$ and denote its cost by $g(f)$;

For each $i, j \in N$, compute shortest path from i to j under link costs $c(e) = g(e) + \sum_{\tilde{f} \neq \tilde{e}} w(e, \tilde{f})$ and denote its cost by d_{ij} ;

(This identifies primary path with link detours from i to j for all $i, j \in N$.)

$V(k) \leftarrow \sum_{i \in N, i \neq k} R_i d_{ik} + \sum_{j \in N, j \neq k} C_j d_{kj}$;

$\bar{k} \leftarrow \arg\min_{k \in N} V(k)$;

(Denote the primary path with link detours from i to $\bar{k}$ by P_i for all $i \neq \bar{k}$ and primary path with link detours from $\bar{k}$ to j by Q_j for all $j \neq \bar{k}$.)

$s(e) \leftarrow \sum_{i \neq \bar{k}, W(P_i) \ni e} R_i + \sum_{j \neq \bar{k}, W(Q_j) \ni e} C_j \quad \forall\, e \in E$;

$s'(e, \tilde{f}) \leftarrow \sum_{i \neq \bar{k}, W(P_i) \ni \tilde{f}, B_{\tilde{f}}(P_i) \ni e} R_i + \sum_{j \neq \bar{k}, W(Q_j) \ni \tilde{f}, B_{\tilde{f}}(Q_j) \ni e} C_j \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f}$;

$\alpha \leftarrow \min_{e \in E} \frac{u_e}{s(e) + \max_{\tilde{f} \in \tilde{E}, \tilde{f} \neq \tilde{e}} s'(e, \tilde{f})}$;

$\Delta(e) \leftarrow \alpha s(e) \quad \forall\, e \in E$;

$\Delta'(e, \tilde{f}) \leftarrow \alpha s'(e, \tilde{f}) \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f}$;

$work(e) \leftarrow work(e) + \Delta(e) \quad \forall\, e \in E$;

$bkp(e, \tilde{f}) \leftarrow bkp(e, \tilde{f}) + \Delta'(e, \tilde{f}) \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f}$;

$w(e, \tilde{f}) \leftarrow w(e, \tilde{f})(1 + \frac{\epsilon[\Delta(e) + \Delta'(e, \tilde{f})]}{u_e}) \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f}$;

$\alpha_{\bar{k}} \leftarrow \alpha_{\bar{k}} + \alpha$;

$D \leftarrow \sum_{e \in E} \sum_{\tilde{f} \in \tilde{E}, \tilde{f} \neq \tilde{e}} u_e w(e, \tilde{f})$;

end while

$bkp\_max(e) \leftarrow \max_{\tilde{f} \neq \tilde{e}} bkp(e, \tilde{f}) \quad \forall\, e \in E$;

$scale(e) \leftarrow \frac{work(e) + bkp\_max(e)}{u_e} \quad \forall\, e \in E$;

$scale\_max \leftarrow \max_{e \in E} scale(e)$;

$\alpha_k \leftarrow \frac{\alpha_k}{scale\_max}$ for all $k \in N$;

Output traffic split ratios α_k ;

We next analyze the approximation guarantee and running time of Algorithm LR.

Analysis of Approximation Guarantee

The analysis follows the same approach as that of the strongly polynomial time algorithm for maximum throughput two-phase routing for the unprotected case in Section 4.3.5.

Given a set of dual weights $w(e,\tilde{f})$, let $D(w)$ denote the dual objective function value and let $\Gamma(w)$ denote the minimum value of the LHS of dual program constraint (7.14) over all nodes $k \in N$. Then, solving the dual program is equivalent to finding a set of weights $w(e,\tilde{f})$ such that $\frac{D(w)}{\Gamma(w)}$ is minimized. Denote the optimal objective function value of the latter by θ, i.e., $\theta = \min_w \frac{D(w)}{\Gamma(w)}$. Let w_{t-1} denote the respective weight functions at the beginning of iteration t of the **while** loop, and let A_{t-1} be the value of $\sum_{j\in N} \alpha_j$ (primal objective function) up to the end of iteration $t-1$. Suppose the algorithm terminates after iteration L.

Lemma 7.3.1 *At the end of every iteration t, $1 \leq t \leq L$, of Algorithm LR, the following holds*

$$D(w_t) \leq m(\frac{m}{2} - 1)\delta \prod_{j=1}^{t}[1 + \frac{\epsilon}{\theta}(A_j - A_{j-1})]$$

Proof: Let $\bar{k} \in N$ be the node for which LHS of dual constraint (7.14) is minimum and let P_i, Q_j be the corresponding paths (as defined earlier) along which traffic is sent during iteration t. Recall that the weights $w(e,\tilde{f})$ are updated as:

$$w(e,\tilde{f}) \leftarrow w(e,\tilde{f})\left(1 + \frac{\epsilon[\Delta(e) + \Delta'(e,\tilde{f})]}{u_e}\right)$$

where $\Delta(e)$ is the total working flow on link e, and $\Delta'(e,\tilde{f})$ is the total restoration flow on link e after failure of link f (both sent during iteration t). Using this, we have

$$\begin{aligned}
D(w_t) &= \sum_{e\in E}\sum_{\tilde{f}\in\tilde{E},\tilde{e}\neq\tilde{f}} u_e w_t(e,\tilde{f}) \\
&= \sum_{e\in E}\sum_{\tilde{f}\in\tilde{E},\tilde{e}\neq\tilde{f}} u_e w_{t-1}(e,\tilde{f}) + \epsilon\sum_{e\in E}\sum_{\tilde{f}\in\tilde{E},\tilde{e}\neq\tilde{f}} w_{t-1}(e,\tilde{f})[\Delta(e) + \Delta'(e,\tilde{f})] \\
&= D(w_{t-1}) + \epsilon\sum_{e\in E}\sum_{\tilde{f}\in\tilde{E},\tilde{e}\neq\tilde{f}} w_{t-1}(e,\tilde{f})[\sum_{i\neq\bar{k},W(P_i)\ni e} \alpha R_i + \sum_{j\neq\bar{k},W(Q_j)\ni e} \alpha C_j + \\
&\quad \sum_{i\neq\bar{k},B_{\tilde{f}}(P_i)\ni e} \alpha R_i + \sum_{j\neq\bar{k}.B_{\tilde{f}}(Q_j)\ni e} \alpha C_j]
\end{aligned}$$

Interchanging the summations on the RHS of the above equation and first summing

along paths P_i, Q_j, and then over i, j respectively, we can rewrite the RHS of the above equation to obtain

$$\begin{aligned}
D(w_t) &= D(w_{t-1}) + \epsilon\alpha[\sum_{i\neq\bar{k}} R_i\{\sum_{e\in W(P_i)}\sum_{\tilde{f}\neq\tilde{e}} w_{t-1}(e,\tilde{f}) + \\
&\quad \sum_{\tilde{f}\in W(P_i)}\sum_{e\in B_{\tilde{f}}(P_i)} w_{t-1}(e,\tilde{f})\} + \sum_{j\neq\bar{k}} C_j\{\sum_{e\in W(Q_j)}\sum_{\tilde{f}\neq\tilde{e}} w_{t-1}(e,\tilde{f}) + \\
&\quad \sum_{\tilde{f}\in W(Q_j)}\sum_{e\in B_{\tilde{f}}(Q_j)} w_{t-1}(e,\tilde{f})\}] \\
&= D(w_{t-1}) + \epsilon\alpha[\sum_{i\neq\bar{k}} R_i\Phi(w_{t-1},P_i) + \sum_{j\neq\bar{k}} C_j\Phi(w_{t-1},Q_j)] \\
&= D(w_{t-1}) + \epsilon\alpha[\sum_{i\neq\bar{k}} R_i \min_{P\in\mathcal{PL}_{i\bar{k}}}\Phi(w_{t-1},P) + \sum_{j\neq\bar{k}} C_j \min_{P\in\mathcal{PL}_{\bar{k}j}}\Phi(w_{t-1},P)] \\
&= D(w_{t-1}) + \epsilon\alpha\Gamma(w_{t-1}) \qquad (7.19) \\
&= D(w_{t-1}) + \epsilon(A_t - A_{t-1})\Gamma(w_{t-1})
\end{aligned}$$

The step leading to (7.19) follows from the choice of intermediate node $\bar{k}$ and associated paths P_i, Q_j that minimize $V(k)$ for the set of weights $w_{t-1}(e,\tilde{f})$ at the beginning of iteration t.

Using this for each iteration down to the first one, we have

$$D(w_t) = D(w_0) + \epsilon\sum_{j=1}^{t}(A_j - A_{j-1})\Gamma(w_{j-1}) \qquad (7.20)$$

From the definition of θ, we have $\theta \leq \frac{D(w_{j-1})}{\Gamma(w_{j-1})}$, whence $\Gamma(w_{j-1}) \leq \frac{1}{\theta}D(w_{j-1})$. Since a failure brings down both a link and its reverse, the number of possible failures is $\frac{m}{2}$. Thus, the inner summation over $\tilde{f}$ in the dual objective function runs over $\frac{m}{2}-1$ values of possible link failures $\tilde{f}$, and hence, $D(w_0) = m(\frac{m}{2}-1)\delta$. Using these in equation (7.20), we have

$$D(w_t) \leq m(\frac{m}{2}-1)\delta + \frac{\epsilon}{\theta}\sum_{j=1}^{t}(A_j - A_{j-1})D(w_{j-1}) \qquad (7.21)$$

The property claimed in the lemma can now be proved using inequality (7.21) and mathematical induction on the iteration number t. The method is similar to that used in the proof of Lemma 4.3.1. ■

We now estimate the factor by which the objective function value value A_L in the primal solution needs to be scaled when the algorithm terminates so as to ensure that link capacity constraints are not violated.

Lemma 7.3.2 *When Algorithm LR terminates, the primal solution needs to be scaled by a factor of at most* $\log_{1+\epsilon} \frac{1+\epsilon}{\delta}$ *to ensure primal feasibility.*

Proof: We will show that the working traffic on link e plus the restoration traffic on link e due to failure of any other link $\tilde{f}$ is at most u_e when the primal solution is scaled by the above factor.

Consider any link e and the failure of some other link $\tilde{f}$ for $e, f \in E, \tilde{f} \neq \tilde{e}$. Recall that the associated weight $w(e, \tilde{f})$ is updated as:

$$w(e, \tilde{f}) \leftarrow w(e, \tilde{f}) \left(1 + \frac{\epsilon[\Delta(e) + \Delta'(e, \tilde{f})]}{u_e}\right)$$

The term $(\Delta(e) + \Delta'(e, f))$ corresponds to the traffic on link e sent during an iteration under either (or both) of the following circumstances:

- Link e appears on any of the primary paths of P_i, Q_j, in which case the working traffic on this link totals to $\Delta(e)$, or
- Link e appears on any of the link detours of P_i, Q_j protecting (primary) link $\tilde{f}$ (i.e., either f or its reverse $\hat{f}$) in which case the restoration traffic on link e after failure of link $\tilde{f}$ totals to $\Delta'(e, \tilde{f})$.

Let the sequence of $(\Delta(e) + \Delta'(e, \tilde{f}))$ values corresponding to working plus restoration traffic (due to failure of link $\tilde{f}$) on link e over all iterations be $\Delta_1, \Delta_2, \ldots, \Delta_L$. Let $\sum_{t=1}^{L} \Delta_t = \kappa u_e$, i.e., the sum of working traffic plus restoration traffic after failure of link $\tilde{f}$ routed on link e exceeds its capacity by a factor of κ.

Because of the way in which α is chosen in accordance with equations (7.16)-(7.18), we have $\Delta_i \leq u_e$ for all i. Hence, dual weights $w(e, \tilde{f})$ are updated by a factor of at most $1+\epsilon$ after each iteration. Since the algorithm terminates when $D(w) \geq 1$, and since dual weights are updated by a factor of at most $1+\epsilon$ after each iteration, we have $D(w_L) < 1+\epsilon$. Since the weight $w(e, \tilde{f})$, with coefficient u_e, is one of the summing components of $D(w)$, we have $u_e w_L(e, \tilde{f}) < 1+\epsilon$. Also, the value of $w_L(e, \tilde{f})$ is given by

$$w_L(e, \tilde{f}) = \frac{\delta}{u_e} \prod_{t=1}^{L} (1 + \frac{\Delta_t}{u_e}\epsilon)$$

Using the inequality $(1+cx) \geq (1+x)^c \ \ \forall\, x \geq 0$ and any $0 \leq c \leq 1$ and setting $x = \epsilon$ and $c = \frac{\Delta_t}{u_e} \leq 1$, we have

$$\begin{aligned} \frac{1+\epsilon}{u_e} \ > \ w_L(e, \tilde{f}) \ &\geq \ \frac{\delta}{u_e} \prod_{t=1}^{L} (1+\epsilon)^{\Delta_t/u_e} \\ &= \ \frac{\delta}{u_e} (1+\epsilon)^{\sum_{t=1}^{L} \Delta_t/u_e} \\ &= \ \frac{\delta}{u_e} (1+\epsilon)^{\kappa} \end{aligned}$$

whence,

$$\kappa < \log_{1+\epsilon} \frac{1+\epsilon}{\delta}$$

■

The values of ϵ and δ are related, in the following theorem, to the approximation factor guarantee of Algorithm LR.

Theorem 7.3.3 *For any given $0 < \epsilon' \leq 0.5$, Algorithm LR computes a solution with objective function value within $(1+\epsilon')$-factor of the optimum for*

$$\delta = \frac{1+\epsilon}{[(1+\epsilon)\frac{m^2}{2}]^{1/\epsilon}} \quad \textit{and} \quad \epsilon = \frac{\epsilon'}{2}$$

Proof: Using Lemma 7.3.1 and the inequality $1 + x \leq e^x$ for all $x \geq 0$, we have

$$\begin{aligned} D(w_t) &\leq m(\frac{m}{2} - 1)\delta \prod_{j=1}^{t} e^{\frac{\epsilon}{\theta}(A_j - A_{j-1})} \\ &< \frac{m^2\delta}{2} e^{\epsilon A_t/\theta} \end{aligned}$$

The simplification in the above step uses telescopic cancellation of the sum $(A_j - A_{j-1})$ over j. Since the algorithm terminates after iteration L, we must have $D(w_L) \geq 1$. Thus,

$$1 \leq D(w_L) \leq \frac{m^2\delta}{2} e^{\epsilon A_L/\theta}$$

whence,

$$\frac{\theta}{A_L} \leq \frac{\epsilon}{\ln \frac{2}{m^2\delta}} \tag{7.22}$$

From Lemma 7.3.2, the objective function value of the feasible primal solution after scaling is at least

$$\frac{A_L}{\log_{1+\epsilon} \frac{1+\epsilon}{\delta}}$$

The approximation factor for the primal solution is at most the gap (ratio) between the dual and primal solutions. Using the lower bound on the primal solution and inequality (7.22), this is at most

$$\begin{aligned} \frac{\theta}{\frac{A_L}{\log_{1+\epsilon} \frac{1+\epsilon}{\delta}}} &\leq \frac{\epsilon \log_{1+\epsilon} \frac{1+\epsilon}{\delta}}{\ln \frac{2}{m^2\delta}} \\ &= \frac{\epsilon}{\ln(1+\epsilon)} \frac{\ln \frac{1+\epsilon}{\delta}}{\ln \frac{2}{m^2\delta}} \end{aligned}$$

The quantity $\ln \frac{1+\epsilon}{\delta} / \ln \frac{2}{m^2\delta}$ equals $\frac{1}{1-\epsilon}$ for $\delta = (1+\epsilon)/[(1+\epsilon)\frac{m^2}{2}]^{1/\epsilon}$. Using this value of δ, the approximation factor is upper bounded by $\frac{\epsilon}{(1-\epsilon)\ln(1+\epsilon)}$. This quantity is at most $1 + 2\epsilon$ for $\epsilon \leq 0.25$. Setting $\epsilon = \frac{\epsilon'}{2}$, we get the desired approximation ratio of $1 + \epsilon'$. ■

Analysis of Running Time

We show that the running time of Algorithm LR is strongly polynomial.

Theorem 7.3.4 *For any given $\epsilon > 0$ chosen to provide the desired approximation factor guarantee in accordance with Theorem 7.3.3, Algorithm LR runs in time*

$$O\left(\frac{1}{\epsilon^2}m^3(m + n\log n)\log m\right)$$

which is strongly polynomial.

Proof: We first consider the running time of each iteration of the algorithm during which node $\bar{k}$ and associated paths P_i, Q_j are chosen to send traffic. Computation of $V(k)$ for all $kinN$ using the procedure described at the beginning of Section 7.3.4 takes a total of $O(m^2 + nm\log n)$ time. All other operations within an iteration are absorbed by this, leading to a total of $O(m(m + n\log n))$ time per iteration.

We next estimate the number of iterations before the algorithm terminates. Recall that in each iteration, traffic is sent along paths P_i, Q_j corresponding to the maximum value of intermediate node split ratio α such that for any link e, the working traffic $\Delta(e)$ plus the restoration traffic $\Delta'(e, \tilde{f})$ that appears on link e after failure of some other link $\tilde{f}$ corresponding to that iteration is at most u_e. Thus, for at least one link e and some link failure $\tilde{f} \neq \tilde{e}$, the value of $(\Delta(e) + \Delta'(e, \tilde{f}))$ equals u_e and the weight $w(e, \tilde{f})$, increases by a factor of $1 + \epsilon$. Accordingly, with each iteration, we can associate a weight $w(e, \tilde{f})$ which increases by a factor of $1 + \epsilon$.

Consider the weight $w(e, \tilde{f})$ for fixed $e, f \in E, \tilde{e} \neq \tilde{f}$. Since $w_0(e, \tilde{f}) = \frac{\delta}{u_e}$ and $w_L(e, \tilde{f}) < \frac{1+\epsilon}{u_e}$ (as deduced in the proof of Lemma 7.3.2), the maximum number of times that this weight can be associated with any iteration is

$$\log_{1+\epsilon}\frac{1+\epsilon}{\delta} = \frac{1}{\epsilon}(1 + \log_{1+\epsilon}\frac{m^2}{2}) = O(\frac{1}{\epsilon}\log_{1+\epsilon}\frac{m^2}{2})$$

Since there are a total of $m(\frac{m}{2}-1)$ weights $w(e,\tilde{f})$, hence the total number of iterations is upper bounded by $O(\frac{1}{\epsilon}m(\frac{m}{2}-1)\log_{1+\epsilon}\frac{m^2}{2})$. Multiplying this by the running time per iteration, we obtain the overall algorithm running time as $O(\frac{1}{\epsilon}m^3(m+n\log n)\log_{1+\epsilon}\frac{m^2}{2}) = O(\frac{1}{\epsilon}m^3(m+n\log n)\log_{1+\epsilon} m)$. Since $\ln(1+\epsilon) = \Theta(\epsilon)$, this is $O(\frac{1}{\epsilon^2}m^3(m+n\log n)\log m)$. ■

7.4 Adding K-Route Path Restoration to Two-Phase Routing

In order to make two-phase routing resilient to link failures using K-route path restoration, Phase 1 and Phase 2 traffic is routed along link-disjoint K-route paths as discussed in Section 7.1.2. Given a network with link capacities u_e and constraints R_i, C_j on the ingress-egress traffic, we consider the problem of two-phase routing with K-route path restoration so as to maximize throughput. The throughput is the maximum multiplier λ such that all matrices in $\lambda \cdot \mathcal{T}(\vec{R},\vec{C})$ can be feasibly routed with K-route path restoration under given link capacities.

We do not have a polynomial size linear programming formulation for the problem. We develop a combinatorial algorithm by extending the primal-dual approach used for maximizing throughput for the unprotected case in Section 4.3. The algorithm computes the traffic split ratios and routing of Phase 1 and Phase 2 paths (with K-route path restoration) up to $(1+\epsilon)$-factor of the optimal objective function value (maximum throughput) for any $\epsilon > 0$. It uses a path flow based linear programming formulation for the problem which we describe next.

We will see that there is a polynomial time separation oracle for the constraints in the dual of the path flow based linear program. The dual problem (and hence the primal) is thus solvable in polynomial time using the ellipsoid algorithm for linear programming [S86]. The running time of the ellipsoid algorithm, however, is not feasible for practical implementations – hence the usefulness of the combinatorial

algorithm we develop.

7.4.1 Path Flow Based LP Formulation

Let $\mathcal{DP}_{ij}$ denote the set of all path sets consisting of two or more link-disjoint paths 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 maximum throughput two-phase routing with K-route path restoration can be formulated as the following path flow based linear program:

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

subject to

$$\sum_{P \in \mathcal{DP}_{ij}} (\chi(P)-1)x(P) = \alpha_j R_i + \alpha_i C_j \quad \forall\, i,j \in N \tag{7.23}$$

$$\sum_{i,j \in N} \sum_{P \in \mathcal{DP}_{ij}, P \ni e} x(P) \leq u_e \quad \forall\, e \in E \tag{7.24}$$

$$\alpha_i \geq 0 \quad \forall\, i \in N \tag{7.25}$$

$$x(P) \geq 0 \quad \forall\, P \in \mathcal{DP}_{ij}, \quad \forall\, i,j \in N \tag{7.26}$$

Constraints (7.23) correspond to the routing of $\alpha_j R_i + \alpha_i C_j$ amount of demand from node i to node j along K-route paths. Constraints (7.24) are the link capacity constraints. Similarly to the formulation for the unprotected 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 7.4.2, we consider the dual of the above linear program. In general, a network can have an exponential number of paths (in the size of the network). Hence, the primal program can have possibly exponential number of variables and

its dual can have an exponential number of constraints, and are both not suitable for fast implementation on large sized networks. The usefulness of the primal and dual formulation is in designing a fast (polynomial time) combinatorial algorithm for the problem.

7.4.2 Dual of Path Flow Based LP Formulation

The dual formulation of the above linear program associates a variable π_{ij} with each demand constraint in (7.23) and a non-negative variable $w(e)$ with each link capacity constraint in (7.24). The dual program can be written as:

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

subject to

$$\sum_{e \in P} w(e) \geq (\chi(P) - 1)\pi_{ij} \quad \forall\, P \in \mathcal{DP}_{ij}, \quad \forall\, i, j \in N \tag{7.27}$$

$$\sum_{i \in N, i \neq k} R_i \pi_{ik} + \sum_{j \in N, \neq k} C_j \pi_{kj} \geq 1 \quad \forall\, k \in N \tag{7.28}$$

$$w(e) \geq 0 \quad \forall\, e \in E \tag{7.29}$$

Because of the nature of constraints (7.28), we can assume that the variables π_{ij} attain the maximum possible value given by constraints (7.27) in any optimal solution. Then, we have

$$\pi_{ij} = \min_{P \in \mathcal{DP}_{ij}} \frac{1}{\chi(P) - 1} \sum_{e \in P} w(e) \quad \forall\, i, j \in N$$

This allows us to eliminate the dual variables π_{ij}. The dual problem can now be written as:

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

subject to

$$\sum_{i \in N, i \neq k} R_i \min_{P \in \mathcal{DP}_{ik}} \frac{1}{\chi(P) - 1} \sum_{e \in P} w(e) \quad + \quad \sum_{j \in N, j \neq k} C_j \min_{P \in \mathcal{DP}_{kj}} \frac{1}{\chi(P) - 1} \sum_{e \in P} w(e) \geq 1 \quad \forall\, k \in N \qquad (7.30)$$

$$w(e) \geq 0 \quad \forall\, e \in E \qquad (7.31)$$

In the next section, we show how $\min_{P \in \mathcal{DP}_{ij}} \frac{1}{\chi(P)-1} \sum_{e \in P} w(e)$ can be evaluated in polynomial time for any given $i, j \in N$. This gives us a separation oracle for the constraints (7.27) in the original dual program. The constraints (7.28) can clearly be verified in polynomial time. This implies that the dual problem (and, hence the primal problem) can be solved in polynomial time using the ellipsoid algorithm for linear programming [S86]. As mentioned earlier, the running time of the ellipsoid algorithm is not feasible for practical implementations.

7.4.3 Combinatorial Algorithm

For any node $k \in N$ and weights $w(e)$, let $V(k)$ denote the LHS of constraint (7.30). The combinatorial algorithm we develop is iterative and needs to compute $\min_{k \in N} V(k)$ for given weights $w(e)$ in every iteration. To compute $V(k)$ for all $k \in N$, we need to know the values $\min_{P \in \mathcal{DP}_{ij}} \frac{1}{\chi(P)-1} \sum_{e \in P} w(e)$ for all $i, j \in N$.

Given the weights $w(e)$, the quantity $\min_{P \in \mathcal{DP}_{ij}} \frac{1}{\chi(P)-1} \sum_{e \in P} w(e)$, for each $i, j \in N$, can be computed in polynomial time as follows. For a fixed K, the problem of finding K disjoint paths of minimum cost from node i to node j is equivalent to finding a minimum cost network flow of value K from node i to node j on graph G with link costs equal to $w(e)$ and link capacities equal to 1. Because the minimum cost flow problem has an integer optimal solution [AMO93], the unit link capacities ensure that the paths in the obtained flow are link-disjoint. Note that these paths are

also link-disjoint in the *undirected sense*, since if a link and its reverse are contained in two different paths, the unit flow in each direction of that link would cancel each other.

However, the value of K is not fixed and we need to find a $K \geq 2$ and associated set P of K link-disjoint paths that minimizes $\frac{1}{K-1}\sum_{e\in P} w(e)$. For this purpose, we use the successive shortest paths algorithm [AMO93] for the minimum cost flow problem. This algorithm, when applied to the graph G with unit link capacities, sends an unit flow from i to j in each iteration along shortest cost paths in the residual network. Thus, the value of K increases by 1 between successive iterations and the algorithm finds K disjoint paths of minimum cost incrementally. The algorithm stops when no path exists from i to j in the residual network. The final value of K is the maximum number of link-disjoint paths from i to j in the graph G. We can compute the value $\frac{1}{K-1}\sum_{e\in P} w(e)$ after every iteration and select the value of K and associated K disjoint paths that minimize it.

The overall algorithm works as follows. Start with initial weights $w(e) = \frac{\delta}{u_e}$ (the quantity δ depends on ϵ and is derived later). Repeat the following until the dual objective function value is greater than or equal to 1:

1. Compute the node k for which $V(k)$ 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 $i \neq \bar{k}$ and link-disjoint path set Q_j from node $\bar{k}$ to node j for all $j \neq \bar{k}$. (These are the link-disjoint path sets between respective node pairs obtained during computation of $V(k)$ as described above.) This is illustrated in Figure 7-6.

2. For a traffic split ratio of 1 for intermediate node $\bar{k}$, the traffic associated with path set P_i is R_i for all $i \neq \bar{f}$ and the traffic associated with path set Q_j is C_j for all $j \neq \bar{f}$. Using this, compute the bandwidth $s(e)$ required on link e per

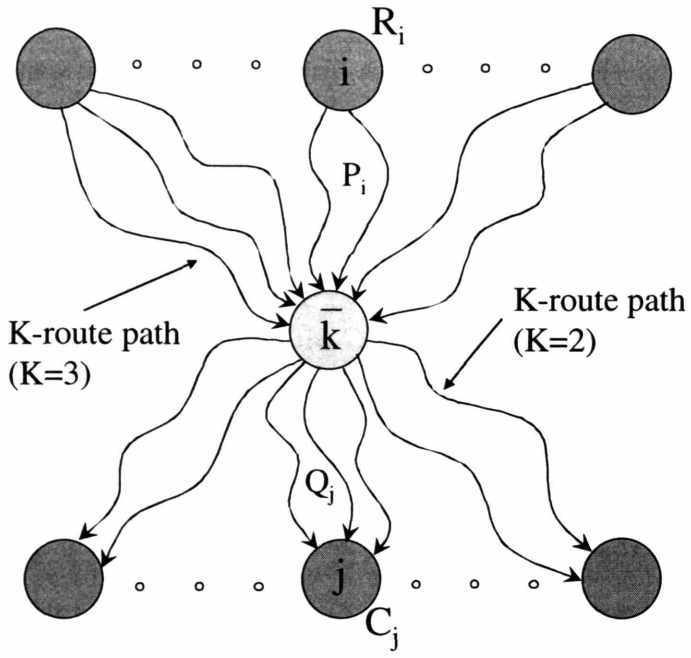


Figure 7-6: One Step in the Primal-Dual Computation for K-Route Path Restoration in Two-Phase Routing.

unit split ratio $\alpha_{\bar{k}}$ for intermediate node $\bar{k}$ as

$$s(e) = \sum_{i \neq \bar{k}, P_i \ni e} \frac{R_i}{\chi(P_i) - 1} + \sum_{j \neq \bar{k}, Q_j \ni e} \frac{C_j}{\chi(Q_j) - 1} \quad \forall\, e \in E \tag{7.32}$$

3. Compute the maximum value α for the traffic split ratio for intermediate $\bar{k}$ that does not lead to violation of (original) link capacity constraints for the above required bandwidth as

$$\alpha = \min_{e \in E} \frac{u_e}{s(e)} \tag{7.33}$$

4. For this value α of the traffic split ratio for intermediate node $\bar{k}$, send αR_i amount of traffic from node i to node $\bar{k}$ along path set P_i for all $i \neq \bar{k}$ and αC_j amount of traffic from node $\bar{k}$ to node j along path set Q_j for all $j \neq \bar{k}$.

(Sending traffic of d units on path set P means sending traffic of $\frac{d}{\chi(P)-1}$ units on each of the $\chi(P)$ paths in the set P.) Compute the bandwidth usage $\Delta(e)$ on link e as

$$\Delta(e) = \alpha s(e) \quad \forall\, e \in E$$

5. Update the weights $w(e)$ as follows:

$$w(e) \leftarrow w(e)\left(1 + \frac{\epsilon\Delta(e)}{u_e}\right) \quad \forall\, e \in E$$

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

When the above procedure terminates, primal capacity constraints 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 traffic and split ratios α_i uniformly so that capacity constraints are obeyed.

The pseudo-code for the above procedure, called Algorithm KPR, is provided in the box below. Array $flow(e)$ keeps track of the bandwidth usage on link e as the algorithm progresses. The variable D is initialized to 0 and remains less than 1 as long as the dual objective function value is less than 1. 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 are output.

Algorithm KPR:

$\alpha_k \leftarrow 0 \quad \forall\, k \in N$;

$w(e) \leftarrow \frac{\delta}{u_e} \quad \forall\, e \in E$;

$flow(e) \leftarrow 0 \quad \forall\, e \in E$;

$D \leftarrow 0$;

while $D < 1$ **do**

For all $i, j \in N$, compute link-disjoint path set P from i to j such that $\frac{1}{\chi(P)-1} \sum_{e \in P} w(e)$ is minimized and denote this value by π_{ij} ;

$V(k) \leftarrow \sum_{i \in N, i \neq k} R_i \pi_{ik} + \sum_{j \in N, j \neq k} C_j \pi_{kj}$;

$\bar{k} \leftarrow \arg\min_{k \in N} V(k)$;

(Denote the link-disjoint path set from i to $\bar{k}$ by P_i for all $i \neq \bar{k}$ and link disjoint path set from $\bar{k}$ to j by Q_j for all $j \neq \bar{k}$.)

$s(e) \leftarrow \sum_{i \neq \bar{k}, P_i \ni e} \frac{R_i}{\chi(P_i)-1} + \sum_{j \neq \bar{k}, Q_j \ni e} \frac{C_j}{\chi(Q_j)-1} \quad \forall\, e \in E$;

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

$\Delta(e) \leftarrow \alpha S(e) \quad \forall\, e \in E$;

$flow(e) \leftarrow flow(e) + \Delta(e)$ for all $e \in E$;

$w(e) \leftarrow w(e)(1 + \frac{\epsilon \Delta(e)}{u_e})$ for all $e \in E$;

$\alpha_{\bar{k}} \leftarrow \alpha_{\bar{k}} + \alpha$;

$D \leftarrow \sum_{e \in E} u_e w(e)$;

end while

$scale(e) \leftarrow \frac{flow(e)}{u_e}$ for all $e \in E$;

$scale\_max \leftarrow \max_{e \in E} scale(e)$;

$\alpha_k \leftarrow \frac{\alpha_k}{scale\_max}$ for all $k \in N$;

Output traffic split ratios α_k ;

We next analyze the approximation guarantee and running time of Algorithm KPR.

Analysis of Approximation Guarantee

The analysis follows the same approach as that of the strongly polynomial time algorithm for maximum throughput two-phase routing for the unprotected case in Section 4.3.5.

Given a set of dual weights $w(e)$, let $D(w)$ denote the dual objective function value and let $\Gamma(w)$ denote the minimum value of the LHS of dual program constraint (7.30)

over all nodes $k \in N$. Then, solving the dual program is equivalent to finding a set of weights $w(e)$ such that $\frac{D(w)}{\Gamma(w)}$ is minimized. Denote the optimal objective function value of the latter by θ, i.e., $\theta = \min_w \frac{D(w)}{\Gamma(w)}$. Let w_{t-1} denote the weight function at the beginning of iteration t of the **while** loop, and let A_{t-1} be the value of $\sum_{j \in N} \alpha_j$ (primal objective function) up to the end of iteration $t-1$. Suppose the algorithm terminates after iteration L. The following lemma upper bounds the value of $D(w)$ at the end of every iteration.

Lemma 7.4.1 *At the end of every iteration t, $1 \leq t \leq L$, of Algorithm KPR, the following holds*

$$D(w_t) \leq m\delta \prod_{j=1}^{t} [1 + \frac{\epsilon}{\theta}(A_j - A_{j-1})]$$

Proof: Let $\bar{k} \in N$ be the node for which LHS of dual constraint (7.30) is minimum and let P_i, Q_j be the corresponding link-disjoint path sets (as defined earlier) along which traffic is sent during iteration t. Recall that the weights $w(e)$ are updated as:

$$w(e) \leftarrow w(e)\left(1 + \frac{\epsilon \Delta(e)}{u_e}\right) \quad \forall\, e \in E$$

where $\Delta(e)$ is the total bandwidth usage on link e as a result of the traffic sent during iteration t. Using this, we have

$$\begin{aligned} D(w_t) &= \sum_{e \in E} u_e w_t(e) \\ &= \sum_{e \in E} u_e w_{t-1}(e) + \epsilon \sum_{e \in E} w_{t-1}(e)\Delta(e) \\ &= D(w_{t-1}) + \epsilon \sum_{e \in E} w_{t-1}(e)[\sum_{i \neq \bar{k}, P_i \ni e} \frac{\alpha R_i}{\chi(P_i) - 1} + \sum_{j \neq \bar{k}, Q_j \ni e} \frac{\alpha C_j}{\chi(Q_j) - 1}] \end{aligned}$$

Interchanging the summations on the RHS of the above equation and first summing along links on paths P_i, Q_j, and then over i, j respectively, we can rewrite the RHS

of the above equation to obtain

$$\begin{aligned} D(w_t) &= D(w_{t-1}) + \epsilon\alpha[\sum_{i\neq\bar{k}} R_i \sum_{e\in P_i} \frac{w_{t-1}(e)}{\chi(P_i)-1} + \sum_{j\neq\bar{k}} C_j \sum_{e\in Q_j} \frac{w_{t-1}(e)}{\chi(Q_j)-1}] \\ &= D(w_{t-1}) + \epsilon\alpha[\sum_{i\neq\bar{k}} R_i \min_{P\in\mathcal{DP}_{i\bar{k}}} \sum_{e\in P} \frac{w_{t-1}(e)}{\chi(P)-1} + \sum_{j\neq\bar{k}} C_j \min_{P\in\mathcal{DP}_{\bar{k}j}} \sum_{e\in P} \frac{w_{t-1}(e)}{\chi(P)-1}] \\ &= D(w_{t-1}) + \epsilon\alpha\Gamma(w_{t-1}) \\ &= D(w_{t-1}) + \epsilon(A_t - A_{t-1})\Gamma(w_{t-1}) \end{aligned}$$

The step leading to (7.34) follows from the choice of intermediate node $\bar{k}$ and associated path sets P_i, Q_j that minimize $V(k)$ for the weights $w_{t-1}(e)$ at the beginning of iteration t.

Using this for each iteration down to the first one, we have

$$D(w_t) = D(w_0) + \epsilon\sum_{j=1}^{t}(A_j - A_{j-1})\Gamma(w_{j-1}) \tag{7.34}$$

From the definition of θ, we have $\theta \leq \frac{D(w_{j-1})}{\Gamma(w_{j-1})}$, whence $\Gamma(w_{j-1}) \leq \frac{1}{\theta}D(w_{j-1})$. Also, $D(w_0) = m\delta$. Using these in equation (7.34), we have

$$D(w_t) \leq m\delta + \frac{\epsilon}{\theta}\sum_{j=1}^{t}(A_j - A_{j-1})D(w_{j-1}) \tag{7.35}$$

The property claimed in the lemma can now be proved using inequality (7.35) and mathematical induction on the iteration number t. The method is similar to that used in the proof of Lemma 4.3.1. ■

We now estimate the factor by which the objective function value value A_L in the primal solution needs to be scaled when the algorithm terminates so as to ensure that link capacity constraints are not violated.

Lemma 7.4.2 *When Algorithm KPR terminates, the primal solution needs to be scaled by a factor of at most* $\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$ *to ensure primal feasibility.*

Proof: We need to show that the bandwidth usage on any link e is at most u_e when the primal solution is scaled by the above factor. Consider any link e and associated weight $w(e)$. The value of $w(e)$ is updated by multiplying with the quantity $(1+\frac{\epsilon\Delta(e)}{u_e})$ where $\Delta(e)$ is the bandwidth usage on this link corresponding to the traffic sent during an iteration. Let the sequence of such values $\Delta(e)$ associated with link e over all iterations be $\Delta_1, \Delta_2, \ldots, \Delta_L$. Let $\sum_{t=1}^{L} \Delta_t = \kappa u_e$, i.e., the total bandwidth usage on link e exceeds its capacity by a factor of κ.

Since α is chosen so that link capacity constraints are obeyed for the traffic sent during an iteration (see equations (7.32)-(7.33)), we have $\Delta_i \leq u_e$ for all i. Hence, dual weights $w(e)$ are updated by a factor of at most $1+\epsilon$ after each iteration. Since the algorithm terminates when $D(w) \geq 1$, and since dual weights are updated by a factor of at most $1+\epsilon$ after each iteration, we have $D(w_L) < 1+\epsilon$. Since the weight $w(e)$, with coefficient u_e, is one of the summing components of $D(w)$, we have $u_e w_L(e) < 1+\epsilon$. Also, the value of $w_L(e)$ is given by

$$w_L(e) = \frac{\delta}{u_e} \prod_{t=1}^{L} (1 + \frac{\Delta_t}{u_e}\epsilon)$$

Using the fact that $(1+cx) \geq (1+x)^c$ for all $x \geq 0$ and any $0 \leq c \leq 1$ and setting $x = \epsilon$ and $c = \frac{\Delta_t}{u_e} \leq 1$, we have

$$\begin{aligned} \frac{1+\epsilon}{u_e} > w_L(e) &\geq \frac{\delta}{u_e} \prod_{t=1}^{L} (1+\epsilon)^{\Delta_t/u_e} \\ &= \frac{\delta}{u_e} (1+\epsilon)^{\sum_{t=1}^{L} \Delta_t/u_e} \\ &= \frac{\delta}{u_e} (1+\epsilon)^{\kappa} \end{aligned}$$

whence,

$$\kappa < \log_{1+\epsilon} \frac{1+\epsilon}{\delta}$$

■

The values of ϵ and δ are related, in the following theorem, to the approximation factor guarantee of Algorithm KPR.

Theorem 7.4.3 *For any given $0 < \epsilon' \leq 0.5$, Algorithm KPR computes a solution with objective function value within $(1+\epsilon')$-factor of the optimum for*

$$\delta = \frac{1+\epsilon}{[(1+\epsilon)m]^{1/\epsilon}} \quad \textit{and} \quad \epsilon = \frac{\epsilon'}{2}$$

Proof: Using Lemma 7.4.1 and the inequality $1 + x \leq e^x$ for all $x \geq 0$, we have

$$\begin{aligned} D(w_t) &\leq m\delta \prod_{j=1}^{t} e^{\frac{\epsilon}{\theta}(A_j - A_{j-1})} \\ &= m\delta e^{\epsilon A_t/\theta} \end{aligned}$$

The simplification in the above step uses telescopic cancellation of the sum $(A_j - A_{j-1})$ over j. Since the algorithm terminates after iteration L, we must have $D(w_L) \geq 1$. Thus,

$$1 \leq D(w_L) \leq m\delta e^{\epsilon A_L/\theta}$$

whence,

$$\frac{\theta}{A_L} \leq \frac{\epsilon}{\ln \frac{1}{m\delta}} \tag{7.36}$$

From Lemma 7.4.2, the objective function value of the feasible primal solution after scaling is at least

$$\frac{A_L}{\log_{1+\epsilon} \frac{1+\epsilon}{\delta}}$$

The approximation factor for the primal solution is at most the gap (ratio) between the dual and primal solutions. Using the lower bound on the primal solution and

inequality (7.36), this is at most

$$\begin{aligned}\frac{\theta}{\frac{A_L}{\log_{1+\epsilon}\frac{1+\epsilon}{\delta}}} &\leq \frac{\epsilon \log_{1+\epsilon}\frac{1+\epsilon}{\delta}}{\ln\frac{1}{m\delta}} \\ &= \frac{\epsilon}{\ln(1+\epsilon)}\frac{\ln\frac{1+\epsilon}{\delta}}{\ln\frac{1}{m\delta}}\end{aligned}$$

The quantity $\ln\frac{1+\epsilon}{\delta}/\ln\frac{1}{m\delta}$ equals $\frac{1}{1-\epsilon}$ for $\delta = (1+\epsilon)/[(1+\epsilon)m]^{1/\epsilon}$. Using this value of δ, the approximation factor is upper bounded by $\frac{\epsilon}{\ln(1+\epsilon)}\frac{1}{(1-\epsilon)}$. This quantity is at most $1+2\epsilon$ for $\epsilon \leq 0.25$. Setting $\epsilon = \frac{\epsilon'}{2}$, we get the desired approximation ratio of $1+\epsilon'$. ■

Analysis of Running Time

We show that the running time of Algorithm LR is strongly polynomial. Let K_{max} be the maximum number of link-disjoint paths between any pair of nodes in G. Since either a link or its reverse (but not both) appears on at most one of the paths in a link-disjoint path set, we have $K_{max} \leq \frac{m}{2}$.

Theorem 7.4.4 *For any given $\epsilon > 0$ chosen to provide the desired approximation factor guarantee in accordance with Theorem 7.4.3, Algorithm KPR runs in time*

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

which is strongly polynomial.

Proof: We first consider the running time of each iteration of the algorithm during which node $\bar{k}$ and associated path sets P_i, Q_j are chosen to send traffic. As described at the beginning of this section, computation of $\min_{P\in\mathcal{DP}_{ij}}\frac{1}{\chi(P)-1}\sum_{e\in P}w(e)$ for each $i,j \in N$ uses the successive shortest paths algorithm for minimum cost flow [AMO93]. Since the shortest path computation in this algorithm operates on non-negative costs, we can use Dijkstra's shortest path algorithm with Fibonacci heaps

[AMO93] that takes $O(m + n \log n)$ time. Since the number of link-disjoint paths between any two nodes is at most K_{max}, the maximum number of shortest path computations to compute the above for given $i, j \in N$ is K_{max}. Thus, the time for computing $\min_{P \in \mathcal{DP}_{ij}} \frac{1}{\chi(P)-1} \sum_{e \in P} w(e)$ for all $i, j \in N$ is $O(n^2 K_{max}(m + n \log n))$. All other operations within an iteration are absorbed by this, leading to a total of $O(K_{max} n^2 (m + n \log n))$ time per iteration.

We next estimate the number of iterations before the algorithm terminates. Recall that in each iteration, traffic is sent along paths P_i, Q_j corresponding to the maximum value of intermediate node split ratio α such that for any link e, the bandwidth usage $\Delta(e)$ on link e corresponding to traffic sent during that iteration is at most u_e. Thus, for at least one link e, the value of $\Delta(e)$ equals u_e and the weight $w(e)$ increases by a factor of $1 + \epsilon$. Accordingly, with each iteration, we can associate a weight $w(e)$ which increases by a factor of $1 + \epsilon$.

Consider the weight $w(e)$ for fixed $e \in E$. Since $w_0(e) = \frac{\delta}{u_e}$ and $w_L(e) < \frac{1+\epsilon}{u_e}$ (as deduced in the proof of Lemma 7.4.2), the maximum number of times that this weight can be associated with any iteration is

$$\log_{1+\epsilon} \frac{1+\epsilon}{\delta} = \frac{1}{\epsilon}(1 + \log_{1+\epsilon} m) = O(\frac{1}{\epsilon} \log_{1+\epsilon} m)$$

Since there are a total of m weights $w(e)$, hence the total number of iterations is upper bounded by $O(\frac{1}{\epsilon} m \log_{1+\epsilon} m)$. Multiplying this by the running time per iteration, we obtain the overall algorithm running time as $O(\frac{1}{\epsilon} K_{max} n^2 m (m + n \log n) \log_{1+\epsilon} m)$. Since $\ln(1 + \epsilon) = \Theta(\epsilon)$, this is $O(\frac{1}{\epsilon^2} K_{max} n^2 m (m + n \log n) \log m)$. ■

7.5 Adding Shared Backup Path Restoration to Two-Phase Routing

In order to make two-phase routing resilient to link failures using shared backup path restoration, Phase 1 and Phase 2 traffic is routed along link-disjoint primary-backup path pairs as discussed in Section 7.1.3. Given a network with link capacities u_e and constraints R_i, C_j on the ingress-egress traffic, we consider the problem of two-phase routing with shared backup path restoration so as to maximize throughput. The throughput is the maximum multiplier λ such that all matrices in $\lambda \cdot \mathcal{T}(\vec{R}, \vec{C})$ can be feasibly routed with shared backup path restoration under given link capacities.

We will show that this problem in $\mathcal{NP}$-hard. Hence, there is no polynomial size linear programming formulation for the problem. We develop a primal-dual scheme by considering a path flow based linear programming formulation and extending the primal-dual approach used for maximizing throughput for the unprotected case in Section 4.3. Checking feasibility for the dual linear program reduces to finding a link-disjoint path pair between every pair of nodes that minimizes a certain cost metric involving the dual program variables. We show that this problem is strongly $\mathcal{NP}$-hard and not approximable within any constant factor even in pseudo-polynomial time unless $\mathcal{P} = \mathcal{NP}$. The approximation guarantee of the overall primal-dual method for our problem depends on how closely this disjoint path pair problem can be solved to optimality.

7.5.1 Path Flow Based LP Formulation

We begin with the path flow based linear programming formulation for this problem. Let $\mathcal{PB}_{ij}$ denote the set of all link-disjoint primary-backup path pairs from node i to node j. Let $x(P)$ denote the traffic associated with path pair P. The primary (working) path carrying traffic under normal (no-failure) conditions in P will be denoted by $W(P)$ and the backup path to which traffic is rerouted after failure by

$B(P)$. The problem of two-phase routing with shared backup path restoration so as to maximize the network throughput can be formulated as the following path flow based linear program:

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

subject to

$$\sum_{P \in \mathcal{PB}_{ij}} x(P) = \alpha_j R_i + \alpha_i C_j \quad \forall\, i, j \in N \tag{7.37}$$

$$\sum_{i,j \in N} \sum_{P \in \mathcal{PB}_{ij}, W(P) \ni e} x(P) \leq u_e \quad \forall\, e \in E \tag{7.38}$$

$$\sum_{i,j \in N} \sum_{P \in \mathcal{PB}_{ij}, W(P) \ni e, W(P) \not\ni \tilde{f}} x(P) + \sum_{i,j \in N} \sum_{P \in \mathcal{PB}_{ij}, B(P) \ni e, W(P) \ni \tilde{f}} x(P) \leq u_e \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f} \tag{7.39}$$

$$\alpha_i \geq 0 \quad \forall\, i \in N \tag{7.40}$$

$$x(P) \geq 0 \quad \forall\, P \in \mathcal{PB}_{ij}, \quad \forall\, i, j \in N \tag{7.41}$$

Constraints (7.37) correspond to the routing of $\alpha_j R_i + \alpha_i C_j$ amount of demand from node i to node j along primary paths that are protected by link-disjoint backup paths. Constraints (7.38) are the link capacity constraints under normal (no-failure) conditions. Constraints (7.38) state that after failure of any link $\tilde{f}$ (i.e., link f and its reverse $\hat{f}$), the sum of working traffic (not affected by the failure of link $\tilde{f}$) on a link and the restoration traffic that appears on the link after failure of link $\tilde{f}$ is at most the capacity of the link. These constraints model the two ways of sharing backup capacity described in Section 7.1.3. Note that the (non-)containment $\tilde{f} \in W(P)$ of failure links in primary paths for the summation in these constraints are in the *undirected sense.* Similarly to the formulation for the unprotected case, the throughput is the sum of the traffic split ratios α_i *when these ratios are not constrained to sum to 1.*

7.5.2 Dual of Path Flow Based LP Formulation

The dual formulation of the above linear program associates a variable π_{ij} with each demand constraint in (7.37), a non-negative variable $z(e)$ with each link capacity constraint in (7.38), and a non-negative variable $w(e, \tilde{f})$ with each link capacity constraint in (7.39). The dual program can be written as:

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

subject to

$$\sum_{e \in W(P)} z(e) + \sum_{e \in W(P), \tilde{f} \notin W(P)} w(e, \tilde{f}) + \sum_{e \in B(P), \tilde{f} \in W(P)} w(e, \tilde{f}) \geq \pi_{ij} \quad \forall\, P \in \mathcal{PB}_{ij}, \quad \forall\, i, j \in N \tag{7.42}$$

$$\sum_{i \in N, i \neq k} R_i \pi_{ik} + \sum_{j \in N, \neq k} C_j \pi_{kj} \geq 1 \quad \forall\, k \in N \tag{7.43}$$

$$z(e), w(e, \tilde{f}) \geq 0 \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f} \tag{7.44}$$

Because of the nature of constraints (7.43), we can assume that the variables π_{ij} attain the maximum possible value given by constraints (7.42) in any optimal solution. Then, we have

$$\pi_{ij} = \min_{P \in \mathcal{P}_{ij}} \left(\sum_{e \in W(P)} z(e) + \sum_{e \in W(P), \tilde{f} \notin W(P)} w(e, \tilde{f}) + \sum_{e \in B(P), \tilde{f} \in W(P)} w(e, \tilde{f}) \right) \quad \forall\, i, j \in N$$

This allows us to eliminate the dual variables π_{ij}. Let $\Psi(z, w, P)$ denote the quantity on the LHS of (7.42) (or, inside the min on RHS of above equation) for a primary-backup path pair P and given weights $z(e), w(e, \tilde{f})$. That is,

$$\Psi(z, w, P) = \sum_{e \in W(P)} z(e) + \sum_{e \in W(P), \tilde{f} \notin W(P)} w(e, \tilde{f}) + \sum_{e \in B(P), \tilde{f} \in W(P)} w(e, \tilde{f}) \tag{7.45}$$

After removal of constraints (7.42), the dual problem can be written as:

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

subject to

$$\sum_{i\in N, i\neq k} R_i \min_{P\in\mathcal{PB}_{ik}} \Psi(z,w,P) \;+\; \sum_{j\in N, j\neq k} C_j \min_{P\in\mathcal{PB}_{kj}} \Psi(z,w,P) \geq 1 \quad \forall\, k\in N \tag{7.46}$$

$$z(e), w(e,\tilde{f}) \;\geq\; 0 \quad \forall\, e,f\in E, \tilde{e}\neq\tilde{f} \tag{7.47}$$

In order to check feasibility for the dual problem for given set of weights $z(e), w(e,\tilde{f})$, we need to compute $\min_{P\in\mathcal{PB}_{ij}} \Psi(z,w,P)$ for all node pairs $i,j\in N$ and verify inequality (7.46) for all $k\in N$. For given $i,j\in N$ and weights $z(e), w(e,\tilde{f})$, let SBPR-DISJOINT-PATHS denote the problem of computing a link-disjoint primary backup path pair from node i to node j that minimizes $\Psi(z,w,P)$. We show next that this problem is strongly $\mathcal{NP}$-hard and not approximable within any constant factor even in pseudo-polynomial time unless $\mathcal{P}=\mathcal{NP}$.

7.5.3 Hardness of SBPR-DISJOINT-PATHS Problem for Checking Dual Feasibility

To show that the problem SBPR-DISJOINT-PATHS is $\mathcal{NP}$-hard, we exhibit a reduction from the another disjoint paths problem which we call SIMPLE-DISJOINT-PATHS:

> Problem SIMPLE-DISJOINT-PATHS: *Given a graph $G=(V,E)$ with two sets of non-negative link costs a_e and b_e for all $e\in E$ and source, destination nodes $i,j\in V$, find a minimum cost link-disjoint path pair from node i to node j where the link costs on one of the paths is a_e and on the other path is b_e.*

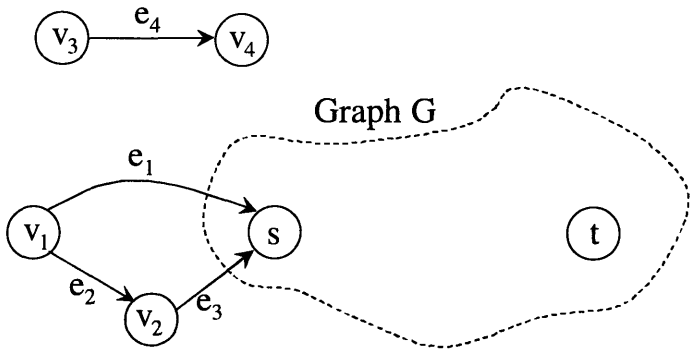


Figure 7-7: Reduction to show $\mathcal{NP}$-hardness of SBPR-DISJOINT-PATHS.

This problem is known to be strongly $\mathcal{NP}$-hard and not approximable within any constant factor even in pseudo-polynomial time unless $\mathcal{P} = \mathcal{NP}$ [LMS92]. Without any loss of generality, we can assume that one of the path pairs in this problem is designated as primary and has link costs a_e, while the other path is designated as backup and has link costs b_e.

Theorem 7.5.1 *The problem SBPR-DISJOINT-PATHS is strongly $\mathcal{NP}$-hard and not approximable within any constant factor even in pseudo-polynomial time unless $\mathcal{P} = \mathcal{NP}$.*

Proof: We construct an approximation preserving reduction from the $\mathcal{NP}$-hard problem SIMPLE-DISJOINT-PATHS. Consider an instance of this problem on a graph $G = (V, E)$ with link costs a_e and b_e and source, destination nodes $s, t \in V$. We construct a graph $G' = (V', E')$ by adding a few nodes and links to G as follows: $V' = V \cup \{v_1, v_2, v_3, v_4\}$ and $E' = E \cup \{e_1 = (v_1, s), e_2 = (v_1, v_2), e_3 = (v_2, s), e_4 = (v_3, v_4)\}$. This is illustrated in Figure 7-7.

Now consider an instance of the SBPR-DISJOINT-PATHS problem on graph G' with source, destination nodes v_1, t and weights $z(e)$ and $w(e, f)$ as follows:

- $z(e) = 0$ for all $e \in E'$.

- $w(e_2, \tilde{e_4})$ is sufficiently large.
- $w(e, \tilde{e_4}) = a_e$ for all $e \in E$.
- $w(e, \tilde{e_1}) = b_e$ for all $e \in E$.
- All other $w(e, \tilde{f})$ values for $e, f \in E', \tilde{e} \neq \tilde{f}$ are zero.

The problem SBPR-DISJOINT-PATHS consists of computing a primary-backup path pair P from node v_1 to node t so as to minimize $\Psi(z, w, P)$ as given by equation (7.45). Since e_4 is an isolated link, it does not appear on any of the paths from v_1 to t. Since there are exactly two links emanating out of node v_1 in G', one of these must be on the primary path and the other on the backup path. If link e_2 is on the primary path in an optimal primary-backup path pair P for SBPR-DISJOINT-PATHS, then the weight $w(e_2, \tilde{e_4})$ will appear in $\Psi(w, z, P)$ and lead to a large value. Thus, we can assume that link e_1 appears on the primary path.

Given a link-disjoint primary-backup path pair (P, B) from s to t in G, these naturally map to link-disjoint primary-backup path pairs (P', B') from v_1 to t in G' as follows: $P' = \{e_1\} \cup P$ and $B' = \{e_2, e_3\} \cup B$. The inverse mapping is also obvious.

Because of the choice of the weights $z(e)$ and $w(e, f)$ in G', we can simplify the expression for $\Psi(z, w, (P', B'))$ on RHS of (7.45) as follows:

$$\begin{aligned} \sum_{e \in P'} z(e) &= 0 \\ \sum_{e \in P', \tilde{f} \notin P'} w(e, \tilde{f}) &= \sum_{e \in P'} w(e, \tilde{e_4}) = \sum_{e \in P} a_e \\ \sum_{e \in B', \tilde{f} \in P'} w(e, \tilde{f}) &= \sum_{e \in B'} w(e, \tilde{e_1}) = \sum_{e \in B} b_e \end{aligned}$$

Adding the above three equations, we have

$$\sum_{e \in P'} z(e) + \sum_{e \in P', \tilde{f} \notin P'} w(e, \tilde{f}) + \sum_{e \in B', \tilde{f} \in P'} w(e, \tilde{f}) = \sum_{e \in P} a_e + \sum_{e \in B} b_e \qquad (7.48)$$

Thus, minimizing the RHS of (7.48) in an instance of SIMPLE-DISJOINT-PATHS is equivalent to minimizing the LHS of (7.48) in the above defined instance of SBPR-DISJOINT-PATHS. This completes the approximation preserving reduction and proves the theorem. ■

7.5.4 Hardness of Two-Phase Routing with Shared Backup Path Restoration

In this section, we use the $\mathcal{NP}$-hardness of the SBPR-DISJOINT-PATHS problem to show that the problem of maximum throughput two-phase routing with shared backup path restoration is $\mathcal{NP}$-hard. We use the equivalence of polynomial time optimization and polynomial time separation for linear programming established by Grötschel, Lovász, and Schrijver [GLS88]. The problem of separation consists of determining whether a given set of values for the variables of the linear program is feasible and if not, identifying at least one constraint that is violated. Such a procedure is also called a *separation oracle* for the linear program.

Theorem 7.5.2 *The problem of computing the maximum throughput for two-phase routing with shared backup path restoration under given link capacities and ingress-egress traffic capacities is $\mathcal{NP}$-hard.*

Proof: The problem SBPR-DISJOINT-PATHS consists of computing a link-disjoint primary-backup path pair P from node i to node j so as to minimize $\Psi(z, w, P)$ as given by equation (7.45). Since $\Psi(z, w, P)$ is linear in the variables $z(e), w(e, \tilde{f})$, this problem can be expressed as the following linear program:

$$\text{maximize } x$$

subject to

$$\Psi(z, w, P) \geq x \ \forall \ P \in \mathcal{PB}_{ij} \tag{7.49}$$

$$z(e), w(e, \tilde{f}) \geq 0 \ \forall \ e, f \in E, \tilde{e} \neq \tilde{f} \tag{7.50}$$

Because this problem was shown to be $\mathcal{NP}$-hard (Theorem 7.5.1) and because of the equivalence of polynomial time optimization and polynomial time separation for linear programming, it follows that the problem of separation for the constraints (7.49) is $\mathcal{NP}$-hard.

Now consider the first dual linear program in Section 7.5.2 for the problem of maximum throughput two-phase routing with shared backup path restoration. In this linear program, the constraints (7.42) for each $i, j \in N$ are identical to constraints (7.49) in the above linear program for SBPR-DISJOINT-PATHS (since the LHS of constraints (7.42) is simply $\Psi(z, w, P)$). Hence, the separation problem for constraints (7.42) is $\mathcal{NP}$-hard. (The constraints (7.43) can easily be verified in polynomial time.) Again using the equivalence of polynomial time optimization and polynomial time separation for linear programming, it follows that the dual and hence the primal optimization problem is $\mathcal{NP}$-hard. This completes the proof. ■

We will show that having access to an approximation oracle for the SBPR-DISJOINT-PATHS problem that outputs a solution within $(1+\xi)$-factor of the optimum value allows us to compute a solution for two-phase routing with shared backup path restoration whose throughput is guaranteed to be arbitrarily close to $(1 + \xi)$-factor of the optimum. The combinatorial algorithm we develop makes a polynomial number of calls to the oracle for SBPR-DISJOINT-PATHS (the rest of the computation is also polynomial time).

Before we proceed with the development of this algorithm, we give a heuristic for the SBPR-DISJOINT-PATHS problem. By using the combinatorial algorithm together with this heuristic as the oracle for SBPR-DISJOINT-PATHS, we obtained solutions within 5% of optimality for maximum throughput two-phase routing with shared backup path restoration, as reported in Section 7.6.

7.5.5 Heuristics for SBPR-DISJOINT-PATHS Problem

Since the problem SBPR-DISJOINT-PATHS is strongly $\mathcal{NP}$-hard and not approximable within any constant factor even in pseudo-polynomial time, we must depend on heuristics to obtain near-optimal solutions. The approach we consider is to generate candidate primary paths in a greedy manner, then compute the best backup path for each such primary path, and choose the least cost solution. By generating more candidate primary paths, the obtained solution can be made closer to the optimal one.

In an optimal solution, it is reasonable to expect that the primary path will have a small number of hops. Thus, we can approximate the sum $\sum_{e \in W(P), \tilde{f} \notin W(P)} w(e, \tilde{f})$ by adding terms for links $\tilde{f}$ that are in $W(P)$. This sum then becomes $\sum_{e \in W(P)} \sum_{\tilde{f} \neq \tilde{e}} w(e, \tilde{f})$. Now define link costs

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

Then, we have

$$\sum_{e \in W(P)} z(e) + \sum_{e \in W(P)} \sum_{\tilde{f} \neq \tilde{e}} w(e, \tilde{f}) = \sum_{e \in W(P)} c(e)$$

Using the link costs $c(e)$, we generate a suitably large number of candidate primary paths in increasing order of cost, using a K-shortest paths algorithm [Y71, La72].

The SBPR-DISJOINT-PATHS problem is easy if the primary path is known. Given a candidate primary path $P_1 = W(P)$, we need to generate a link-disjoint backup path $B_1 = B(P)$ that minimizes $\sum_{e \in B(P), \tilde{f} \in W(P)} w(e, \tilde{f})$. This also involves a shortest path computation as follows. We set the costs of each link (and its reverse) in P_1 to infinity to model the link diversity requirement of the backup path. The costs of the other links are $d(e) = \sum_{\tilde{f} \in W(P)} w(e, \tilde{f})$. This gives

$$\sum_{e \in B(P), \tilde{f} \in W(P)} w(e, \tilde{f}) = \sum_{e \in B(P)} d(e)$$

Thus, a shortest path computation with link costs $d(e)$ will give the best link-disjoint backup path B_1 for the given primary path P_1.

We choose the primary-backup path pair that gives the lowest cost solution to SBPR-DISJOINT-PATHS. By generating more candidate primary paths, we can attempt to improve the cost of the solution. By choosing a K-shortest paths algorithm that generates the paths in increasing order of cost through *incremental* shortest path computations [Y71, La72], more candidate primary paths can be generated in an efficient manner without starting the procedure from the beginning.

7.5.6 Combinatorial Algorithm using Oracle for SBPR-DISJOINT-PATHS

In this section, we use the primal and dual formulations of the path flow based linear program to develop a combinatorial algorithm for the problem. We assume that there is an oracle that outputs a solution to the SBPR-DISJOINT-PATHS problem that is guaranteed to be within $(1+\xi)$-factor of the optimum value and runs in time $T_{DP}(n,m)$ on a graph with n nodes and m links.

For a given node k and weights $z(e), w(e,\tilde{f})$, let $V(k)$ denote the LHS of constraint (7.46). The combinatorial algorithm we develop is iterative and needs to compute $\min_{k\in N} V(k)$ *approximately* in every iteration. This can be done as follows. Given the weights $z(e)$ and $w(e,f)$, we compute $\min_{P\in\mathcal{PB}_{ij}} \Psi(z,w,P)$ for each $i,j \in N$ within $(1+\xi)$-factor of its actual value using the approximation oracle for SBPR-DISJOINT-PATHS. We then use these values to compute $V(k)$ and then take the minimum over all $k \in N$. This gives us $\min_{k\in N} V(k)$ within $(1+\xi)$-factor of its actual value.

The overall algorithm works as follows. Start with initial weights $z(e) = \frac{\delta}{u_e} \;\; \forall\, e \in E$ and $w(e,\tilde{f}) = \frac{\delta}{u_e} \;\; \forall\, e,f \in E, \tilde{e} \neq \tilde{f}$ (the quantity δ depends on ϵ and is derived later). Repeat the following until the dual objective function value is greater than or equal to 1:

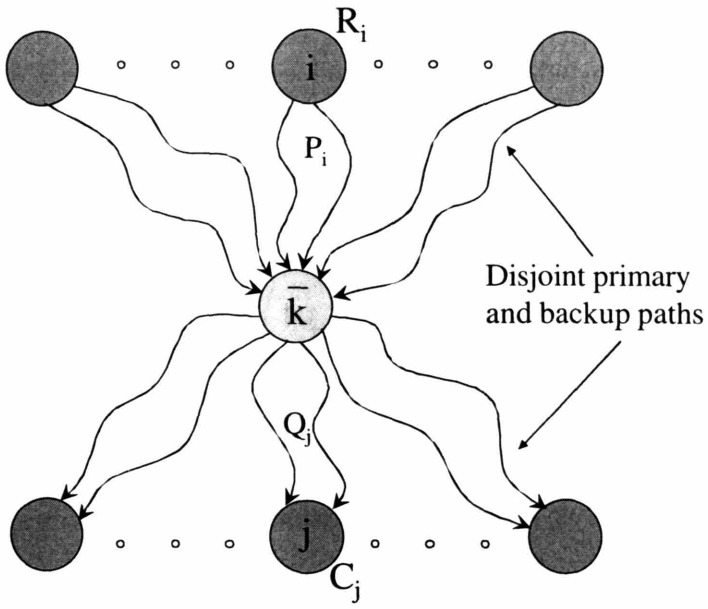


Figure 7-8: One Step in the Primal-Dual Computation for Shared Backup Path Restoration in Two-Phase Routing.

1. Compute node $k = \bar{k}$ for which $V(k)$ is minimum as described above. This identifies a node $\bar{k}$ as well as primary-backup path pairs P_i from node i to node $\bar{k}$ for all $i \neq \bar{k}$ and primary-backup path pairs Q_j from node $\bar{k}$ to node j for all $j \neq \bar{k}$. (These are the link-disjoint primary-backup path pairs between respective nodes obtained from the oracle for SBPR-DISJOINT-PATHS.) This is illustrated in Figure 7-8.

2. For a traffic split ratio of 1 for intermediate node $\bar{k}$, the traffic on path P_i is R_i for all $i \neq \bar{k}$ and the traffic on path Q_j is C_j for all $j \neq \bar{k}$. Using this, compute the working traffic $s(e)$ under normal (no-failure) conditions on link e per unit split ratio $\alpha_{\bar{k}}$ for intermediate node $\bar{k}$ as

$$s(e) = \sum_{i \neq \bar{k}, W(P_i) \ni e} R_i + \sum_{j \neq \bar{k}, W(Q_j) \ni e} C_j \quad \forall\, e \in E \tag{7.51}$$

3. For the above working traffic and primary-backup path pairs P_i, Q_j, compute the traffic $s'(e, \tilde{f})$ on link e after failure of link $\tilde{f}$. The quantity $s'(e, \tilde{f})$ is the sum of working traffic on link e that is not affected by the failure of link $\tilde{f}$ and the restoration traffic that appears on the link after failure of link $\tilde{f}$ and is computed as

$$s'(e,\tilde{f}) = \sum_{i\neq\bar{k},W(P_i)\ni e,W(P_i)\not\ni\tilde{f}} R_i + \sum_{j\neq\bar{k},W(Q_j)\ni e,W(Q_j)\not\ni\tilde{f}} C_j + \sum_{i\neq\bar{k},B(P_i)\ni e,W(P_i)\ni\tilde{f}} R_i + \sum_{j\neq\bar{k},B(Q_j)\ni e,W(Q_j)\ni\tilde{f}} C_j \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f} \quad (7.52)$$

   The maximum possible traffic on link e is thus

$$\max(s(e), \max_{\tilde{f}\neq\tilde{e}} s'(e,\tilde{f})) \quad \forall\, e \in E$$

4. Compute the maximum value α for the traffic split ratio for intermediate $\bar{k}$ that does not lead to violation of (original) link capacity constraints for the above traffic as

$$\alpha = \min_{e\in E} \frac{u_e}{\max(s(e), \max_{\tilde{f}\neq\tilde{e}} s'(e,\tilde{f}))} \quad (7.53)$$

5. For this value α of the traffic split ratio for intermediate node $\bar{k}$, send αR_i amount of traffic from node i to node $\bar{k}$ along primary-backup path pair P_i for all $i \neq \bar{k}$ and αC_j amount of traffic from node $\bar{k}$ to node j along primary-backup path pair Q_j for all $j \neq \bar{k}$. Compute the working traffic $\Delta(e)$ on link e under normal (no-failure) conditions as

$$\Delta(e) = \alpha s(e) \quad \forall\, e \in E$$

and the traffic $\Delta'(e, \tilde{f})$ on link e after failure of any other link f as

$$\Delta'(e, \tilde{f}) = \alpha s'(e, \tilde{f}) \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f}$$

6. Update the weights $z(e)$ and $w(e, \tilde{f})$ as follows:

$$\begin{aligned} z(e) &\leftarrow z(e)\left(1 + \frac{\epsilon\Delta(e)}{u_e}\right) \quad \forall\, e \in E \\ w(e, \tilde{f}) &\leftarrow w(e, \tilde{f})\left(1 + \frac{\epsilon\Delta'(e, \tilde{f})}{u_e}\right) \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f} \end{aligned}$$

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

When the above procedure terminates, primal capacity constraints 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 traffic and 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.

The pseudo-code for the above procedure, called Algorithm SBPR, is provided below. Arrays $work(e)$ and $fail(e, \tilde{f})$ keep track respectively of the working traffic on link e under normal (no-failure) conditions and the traffic on link e after failure of any other link $\tilde{f}$. The variable D is initialized to 0 and remains less than 1 as long as the dual objective function value is less than 1. After the **while** loop terminates (through a jump out of it), 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 are output.

Algorithm SBPR:

$\alpha_k \leftarrow 0 \quad \forall\, k \in N$;

$z(e) \leftarrow \frac{\delta}{u_e} \quad \forall\, e \in E$;

$w(e, \tilde{f}) \leftarrow \frac{\delta}{u_e} \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f}$;

$work(e) \leftarrow 0 \quad \forall\, e \in E$;

$fail(e, \tilde{f}) \leftarrow 0 \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f}$;

$D \leftarrow 0$;

while $D < 1$ **do**

For each $i, j \in N$, compute primary-backup path pair P from i to j that minimizes $\Psi(z, w, P)$ using oracle for SBPR-DISJOINT-PATHS ;

(Denote value of $\Psi(z, w, P)$ for computed path pair P from i to j by $DP(i, j)$ for all $i, j \in N$.)

$V(k) \leftarrow \sum_{i \in N, i \neq k} R_i DP(i, k) + \sum_{j \in N, j \neq k} C_j DP(k, j) \quad \forall\, k \in N$;

$\bar{k} \leftarrow \arg\min_{k \in N} V(k)$;

(Denote primary-backup path pair from i to $\bar{k}$ by P_i for all $i \neq \bar{k}$ and primary-backup path pair from $\bar{k}$ to j by Q_j for all $j \neq \bar{k}$.)

$s(e) \leftarrow \sum_{i \neq \bar{k}, W(P_i) \ni e} R_i + \sum_{j \neq \bar{k}, W(Q_j) \ni e} C_j \quad \forall\, e \in E$;

$s'(e, \tilde{f}) \leftarrow \sum_{i \neq \bar{k}, W(P_i) \ni e, W(P_i) \not\ni \tilde{f}} R_i + \sum_{j \neq \bar{k}, W(Q_j) \ni e, W(Q_j) \not\ni \tilde{f}} C_j +$
$\sum_{i \neq \bar{k}, B(P_i) \ni e, W(P_i) \ni \tilde{f}} R_i + \sum_{j \neq \bar{k}, B(Q_j) \ni e, W(Q_j) \ni \tilde{f}} C_j \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f}$;

$\alpha \leftarrow \min_{e \in E} \frac{u_e}{\max(s(e), \max_{\tilde{f} \neq \tilde{e}} s'(e, \tilde{f}))}$;

$\Delta(e) \leftarrow \alpha s(e) \quad \forall\, e \in E$;

$\Delta'(e, \tilde{f}) \leftarrow \alpha s'(e, \tilde{f}) \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f}$;

$work(e) \leftarrow work(e) + \Delta(e) \quad \forall\, e \in E$;

$fail(e, \tilde{f}) \leftarrow fail(e, \tilde{f}) + \Delta'(e, \tilde{f}) \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f}$;

$z(e) \leftarrow z(e) \left(1 + \frac{\epsilon \Delta(e)}{u_e}\right) \quad \forall\, e \in E$;

$w(e, \tilde{f}) \leftarrow w(e, \tilde{f}) \left(1 + \frac{\epsilon \Delta'(e, \tilde{f})}{u_e}\right) \quad \forall\, e, f \in E, \tilde{e} \neq \tilde{f}$;

$\alpha_{\bar{k}} \leftarrow \alpha_{\bar{k}} + \alpha$;

$D \leftarrow \sum_{e \in E} u_e z(e) + \sum_{e \in E} \sum_{\tilde{f} \in \tilde{E}, \tilde{f} \neq \tilde{e}} u_e w(e, \tilde{f})$;

end while

$fail\_max(e) \leftarrow \max_{\tilde{f} \neq \tilde{e}} fail(e, \tilde{f}) \ \ \forall\, e \in E$;

$scale(e) \leftarrow \frac{\max(work(e), fail\_max(e))}{u_e} \ \ \forall\, e \in E$;

$scale\_max \leftarrow \max_{e \in E} scale(e)$;

$\alpha_k \leftarrow \frac{\alpha_k}{scale\_max}$ for all $k \in N$;

Output traffic split ratios α_k ;

We next show that this combinatorial algorithm provides an approximation guarantee within $(1+\xi+\epsilon)$-factor of the optimum for any given $\epsilon > 0$. (Recall that $(1+\xi)$ is the approximation factor guaranteed by the oracle for the SBPR-DISJOINT-PATHS problem.)

Analysis of Approximation Guarantee

The analysis follows the same approach as that of the strongly polynomial time algorithm for maximum throughput two-phase routing for the unprotected case in Section 4.3.5.

Given a set of dual weights $z(e)$ and $w(e, \tilde{f})$, let $D(z, w)$ denote the dual objective function value and let $\Gamma(z, w)$ denote the minimum value of the LHS of dual program constraint (7.46) over all nodes $k \in N$. Then, solving the dual program is equivalent to finding a set of weights $z(e)$ and $w(e, \tilde{f})$ such that $\frac{D(z,w)}{\Gamma(z,w)}$ is minimized. Denote the optimal objective function value of the latter by θ, i.e., $\theta = \min_{z,w} \frac{D(z,w)}{\Gamma(z,w)}$. Let z_{t-1} and w_{t-1} denote the respective weight functions at the beginning of iteration t of the **while** loop, and let A_{t-1} be the value of $\sum_{j \in N} \alpha_j$ (primal objective function) up to the end of iteration $t-1$. Suppose the algorithm terminates after iteration L. The following lemma upper bounds the value of $D(z, w)$ at the end of every iteration.

Lemma 7.5.3 *At the end of every iteration t, $1 \leq t \leq L$, of Algorithm SBPR, the following holds*

$$D(z_t, w_t) \leq \frac{m^2 \delta}{2} \prod_{j=1}^{t} [1 + \frac{\epsilon(1+\xi)}{\theta}(A_j - A_{j-1})]$$

Proof: Let $k = \bar{k} \in N$ be the node for which $V(k)$ is minimum and let P_i, Q_j be the corresponding primary-backup path pairs (as defined earlier) along which traffic is sent during iteration t. (Note that $V(k)$ for all $k \in N$ is not computed exactly but within $(1+\xi)$-factor of its actual value using the approximation oracle for SBPR-DISJOINT-PATHS.) Recall that the weights $z(e), w(e,\tilde{f})$ are updated as:

$$\begin{aligned} z_t(e) &\leftarrow z_{t-1}(e)\left(1+\frac{\epsilon\Delta(e)}{u_e}\right) \quad \forall\, e \in E \\ w_t(e,\tilde{f}) &\leftarrow w_{t-1}(e,\tilde{f})\left(1+\frac{\epsilon[\Delta(e)+\Delta'(e,\tilde{f})]}{u_e}\right) \quad \forall\, e,f \in E, \tilde{e} \neq \tilde{f} \end{aligned}$$

where $\Delta(e)$ is the total working traffic on link e under normal (no-failure) conditions, and $\Delta'(e,\tilde{f})$ is the total traffic on link e after failure of some other link f (both sent during iteration t). Using this, we have

$$\begin{aligned} D(z_t,w_t) &= \sum_{e\in E} u_e z_t(e) + \sum_{e\in E}\sum_{\tilde{f}\in\tilde{E},\tilde{e}\neq\tilde{f}} u_e w_t(e,\tilde{f}) \\ &= \sum_{e\in E} u_e z_{t-1}(e) + \epsilon\sum_{e\in E} z_{t-1}(e)\Delta(e) + \\ &\quad \sum_{e\in E}\sum_{\tilde{f}\in\tilde{E},\tilde{e}\neq\tilde{f}} u_e w_{t-1}(e,\tilde{f}) + \epsilon\sum_{e\in E}\sum_{\tilde{f}\in\tilde{E},\tilde{e}\neq\tilde{f}} w_{t-1}(e,\tilde{f})\Delta'(e,\tilde{f}) \\ &= D(z_{t-1},w_{t-1}) + \epsilon\sum_{e\in E} z_{t-1}(e)\Big(\sum_{i\neq\bar{k},W(P_i)\ni e}\alpha R_i + \sum_{j\neq\bar{k},W(Q_j)\ni e}\alpha C_j\Big) + \\ &\quad \epsilon\sum_{e\in E}\sum_{\tilde{f}\in\tilde{E},\tilde{e}\neq\tilde{f}} w_{t-1}(e,\tilde{f})\Big(\sum_{i\neq\bar{k},W(P_i)\ni e,W(P_i)\not\ni\tilde{f}}\alpha R_i + \sum_{j\neq\bar{k},W(Q_j)\ni e,W(Q_j)\not\ni\tilde{f}}\alpha C_j + \\ &\quad \sum_{i\neq\bar{k},B(P_i)\ni e,W(P_i)\ni\tilde{f}}\alpha R_i + \sum_{j\neq\bar{k},B(Q_j)\ni e,W(Q_j)\ni\tilde{f}}\alpha C_j\Big) \end{aligned}$$

Interchanging the summations on the RHS of the above equation and first summing along paths P_i, Q_j, and then over i, j respectively, we can rewrite the RHS of the above equation to obtain

$$D(z_t,w_t) = D(z_{t-1},w_{t-1}) + \epsilon\alpha[\sum_{i\neq\bar{k}} R_i \sum_{e\in W(P_i)} z_{t-1}(e) + \sum_{j\neq\bar{k}} C_j \sum_{e\in W(Q_j)} z_{t-1}(e)] +$$

$$\begin{aligned}
& \epsilon\alpha[\sum_{i\neq\bar{k}} R_i \sum_{e\in W(P_i),\tilde{f}\notin W(P_i)} w_{t-1}(e,\tilde{f}) + \sum_{j\neq\bar{k}} C_j \sum_{e\in W(Q_j),\tilde{f}\notin W(Q_j)} w_{t-1}(e,\tilde{f}) + \\
& \sum_{i\neq\bar{k}} R_i \sum_{e\in B(P_i),\tilde{f}\in W(P_i)} w_{t-1}(e,\tilde{f}) + \sum_{j\neq\bar{k}} C_j \sum_{e\in B(Q_j),\tilde{f}\in W(Q_j)} w_{t-1}(e,\tilde{f})] \\
= \; & D(z_{t-1},w_{t-1}) + \epsilon\alpha[\sum_{i\neq\bar{k}} R_i\Psi(z_{t-1},w_{t-1},P_i) + \sum_{j\neq\bar{k}} C_j\Psi(z_{t-1},w_{t-1},Q_j)]
\end{aligned} \tag{7.54}$$

Recall that we obtained the primary-backup path pairs P_i, Q_j by using the approximation oracle for SBPR-DISJOINT-PATHS. This oracle computes $\min_P \Psi(z,w,P)$ over all primary-backup path pairs P between a given pair of nodes within $(1+\xi)$-factor of its actual value. We then used these values to compute $V(k)$ and then took the minimum over all $k \in N$ to identify node $\bar{k}$. The value of $V(\bar{k})$ thus obtained must be within $(1+\xi)$-factor of $\Gamma(z_{t-1},w_{t-1})$. Hence, we have

$$\sum_{i\neq\bar{k}} R_i\Psi(z_{t-1},w_{t-1},P_i) + \sum_{j\neq\bar{k}} C_j\Psi(z_{t-1},w_{t-1},Q_j) \leq (1+\xi)\Gamma(z_{t-1},w_{t-1})$$

Using this in (7.54), we have

$$\begin{aligned}
D(z_t,w_t) &\leq D(z_{t-1},w_{t-1}) + \epsilon\alpha(1+\xi)\Gamma(z_{t-1},w_{t-1}) \\
&= D(z_{t-1},w_{t-1}) + \epsilon(1+\xi)(A_t - A_{t-1})\Gamma(z_{t-1},w_{t-1})
\end{aligned}$$

Using this for each iteration down to the first one, we have

$$D(z_t,w_t) \leq D(w_0) + \epsilon(1+\xi)\sum_{j=1}^{t}(A_j - A_{j-1})\Gamma(z_{j-1},w_{j-1}) \tag{7.55}$$

From the definition of θ, we have $\theta \leq \frac{D(z_{j-1},w_{j-1})}{\Gamma(z_{j-1},w_{j-1})}$, whence $\Gamma(z_{j-1},w_{j-1}) \leq \frac{1}{\theta}D(z_{j-1},w_{j-1})$. The number of weights $z(e)$ is m. Since a failure brings down both a link and its reverse, the number of possible failures is $\frac{m}{2}$. The number of weights $w(e,\tilde{f})$ is thus

$m(\frac{m}{2}-1)$. Hence, $D(w_0) = (m + m(\frac{m}{2}-1))\delta = \frac{m^2\delta}{2}$. Using these in (7.55), we have

$$D(z_t, w_t) \leq \frac{m^2\delta}{2} + \frac{\epsilon(1+\xi)}{\theta}\sum_{j=1}^{t}(A_j - A_{j-1})D(z_{j-1}, w_{j-1}) \tag{7.56}$$

The property claimed in the lemma can now be proved using inequality (7.56) and mathematical induction on the iteration number t. The method is similar to that used in the proof of Lemma 4.3.1. ■

We now estimate the factor by which the objective function value value A_L in the primal solution needs to be scaled when the algorithm terminates so as to ensure that link capacity constraints are not violated.

Lemma 7.5.4 *When Algorithm SBPR terminates, the primal solution needs to be scaled by a factor of at most* $\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$ *to ensure primal feasibility.*

Proof: We need to show two things here after the primal solution is scaled by the above factor. First, the working traffic on every link e under normal (no-failure) conditions is at most the capacity of the link. Second, after failure of any link $\tilde{f}$, the sum of working traffic (not affected by the failure of link $\tilde{f}$) on each link e and the restoration traffic that appears on the link after failure of link $\tilde{f}$ is at most the capacity of the link.

First, consider any link e and associated weight $z(e)$. The value of $z(e)$ is updated by multiplying with the quantity $(1 + \frac{\epsilon\Delta(e)}{u_e})$ where $\Delta(e)$ is working traffic on this link under normal (no-failure) conditions corresponding to an iteration. Let the sequence of such traffic values $\Delta(e)$ associated with link e over all iterations be $\Delta_1, \Delta_2, \ldots, \Delta_L$. Let $\sum_{t=1}^{L}\Delta_t = \kappa u_e$, i.e., the total working traffic routed on link e exceeds its capacity by a factor of κ.

Because of the way in which α is chosen in accordance with equations (7.51)-(7.53), it follows that all dual weights are updated by a factor of at most $1+\epsilon$ after each iteration. Since the algorithm terminates when $D(z, w) \geq 1$, and since dual weights are updated by a factor of at most $1+\epsilon$ after each iteration, we have

$D(z_L, w_L) < 1 + \epsilon$. Since the weight $z(e)$, with coefficient u_e, is one of the summing components of $D(z, w)$, we have $u_e z_L(e) < 1 + \epsilon$. Also, the value of $z_L(e)$ is given by

$$z_L(e, f) = \frac{\delta}{u_e} \prod_{t=1}^{L} (1 + \frac{\Delta_t}{u_e} \epsilon)$$

Using the inequality $(1 + cx) \geq (1 + x)^c \ \forall\, x \geq 0$ and any $0 \leq c \leq 1$ and setting $x = \epsilon$ and $c = \frac{\Delta_t}{u_e} \leq 1$, we have

$$\begin{aligned} \frac{1+\epsilon}{u_e} \ > \ z_L(e) \ &\geq \ \frac{\delta}{u_e} \prod_{t=1}^{L} (1+\epsilon)^{\Delta_t/u_e} \\ &= \ \frac{\delta}{u_e} (1+\epsilon)^{\sum_{t=1}^{L} \Delta_t/u_e} \\ &= \ \frac{\delta}{u_e} (1+\epsilon)^{\kappa} \end{aligned}$$

whence,

$$\kappa < \log_{1+\epsilon} \frac{1+\epsilon}{\delta}$$

Second, consider any link e after failure of some other link $\tilde{f}$ and associated weight $w(e, \tilde{f})$. The value of $w(e, \tilde{f})$ is updated by multiplying with the quantity $(1 + \frac{\epsilon \Delta'(e, \tilde{f})}{u_e})$ where $\Delta'(e, \tilde{f})$ is the traffic on this link after failure of link $\tilde{f}$ corresponding to an iteration. Let the sequence of such traffic values $\Delta'(e, \tilde{f})$ for a given link e and failure link $\tilde{f}$ over all iterations be $\Delta'_1, \Delta'_2, \ldots, \Delta'_L$. Let $\sum_{t=1}^{L} \Delta'_t = \kappa' u_e$, i.e., the total traffic routed on link e after failure of link $\tilde{f}$ exceeds its capacity by a factor of κ'.

As established in the first part of the proof, we have $D(z_L, w_L) < 1 + \epsilon$. Since the weight $w(e, \tilde{f})$, with coefficient u_e, is one of the summing components of $D(z, w)$, we have $u_e w_L(e, \tilde{f}) < 1 + \epsilon$. Also, the value of $w_L(e, \tilde{f})$ is given by

$$w_L(e, \tilde{f}) = \frac{\delta}{u_e} \prod_{t=1}^{L} (1 + \frac{\Delta'_t}{u_e} \epsilon)$$

Using the same inequality as for the first argument, we have

$$\begin{aligned}\frac{1+\epsilon}{u_e} > w_L(e,\tilde{f}) &\geq \frac{\delta}{u_e}\prod_{t=1}^{L}(1+\epsilon)^{\Delta'_t/u_e} \\ &= \frac{\delta}{u_e}(1+\epsilon)^{\sum_{t=1}^{L}\Delta'_t/u_e} \\ &= \frac{\delta}{u_e}(1+\epsilon)^{\kappa'}\end{aligned}$$

whence,

$$\kappa' < \log_{1+\epsilon}\frac{1+\epsilon}{\delta}$$

■

The values of ϵ and δ are related, in the following theorem, to the approximation factor guarantee of Algorithm SBPR.

Theorem 7.5.5 *For any given $0 < \epsilon' \leq 0.5(1+\xi)$, Algorithm SBPR computes a solution with objective function value within $(1+\xi+\epsilon')$-factor of the optimum for*

$$\delta = \frac{1+\epsilon}{[(1+\epsilon)\frac{m^2}{2}]^{1/\epsilon}} \quad \textit{and} \quad \epsilon = \frac{\epsilon'}{2(1+\xi)}$$

Proof: Using Lemma 7.5.3 and the inequality $1+x \leq e^x$ for all $x \geq 0$, we have

$$\begin{aligned}D(z_t,w_t) &\leq \frac{m^2\delta}{2}\prod_{j=1}^{t} e^{\frac{\epsilon(1+\xi)}{\theta}(A_j - A_{j-1})} \\ &= \frac{m^2\delta}{2}e^{\epsilon(1+\xi)A_t/\theta}\end{aligned}$$

The simplification in the above step uses telescopic cancellation of the sum $(A_j - A_{j-1})$ over j. Since the algorithm terminates after iteration L, we must have $D(z_L,w_L) \geq 1$. Thus,

$$1 \leq D(z_L,w_L) \leq \frac{m^2\delta}{2}e^{\epsilon(1+\xi)A_L/\theta}$$

whence,

$$\frac{\theta}{A_L} \leq \frac{\epsilon(1+\xi)}{\ln \frac{2}{m^2\delta}} \tag{7.57}$$

From Lemma 7.5.4, the objective function value of the feasible primal solution after scaling is at least

$$\frac{A_L}{\log_{1+\epsilon} \frac{1+\epsilon}{\delta}}$$

The approximation factor for the primal solution is at most the gap (ratio) between the dual and primal solutions. Using the lower bound on the primal solution and inequality (7.57), this is at most

$$\begin{aligned} \frac{\theta}{\frac{A_L}{\log_{1+\epsilon} \frac{1+\epsilon}{\delta}}} &\leq \frac{\epsilon(1+\xi)\log_{1+\epsilon} \frac{1+\epsilon}{\delta}}{\ln \frac{2}{m^2\delta}} \\ &= \frac{\epsilon(1+\xi)}{\ln(1+\epsilon)} \frac{\ln \frac{1+\epsilon}{\delta}}{\ln \frac{2}{m^2\delta}} \end{aligned}$$

The quantity $\ln \frac{1+\epsilon}{\delta} / \ln \frac{2}{m^2\delta}$ equals $\frac{1}{1-\epsilon}$ for $\delta = (1+\epsilon)/[(1+\epsilon)\frac{m^2}{2}]^{1/\epsilon}$. Using this value of δ, the approximation factor is upper bounded by $\frac{\epsilon(1+\xi)}{(1-\epsilon)\ln(1+\epsilon)}$. The quantity $\frac{\epsilon}{(1-\epsilon)\ln(1+\epsilon)}$ is at most $1+2\epsilon$ for $\epsilon \leq 0.25$. Setting $\epsilon = \frac{\epsilon'}{2(1+\xi)}$, we get the desired approximation ratio of $1+\epsilon'$. ■

Analysis of Running Time

We show that the running time of Algorithm SBPR is a polynomial (in the network size and $\frac{1}{\epsilon}$) times the time taken per call for the oracle for SBPR-DISJOINT-PATHS.

Theorem 7.5.6 *For any given $\epsilon > 0$ chosen to provide the desired approximation factor guarantee in accordance with Theorem 7.5.5, Algorithm SBPR runs in time*

$$O\left(\frac{1}{\epsilon^2} n^2 m^2 T_{DP}(n,m) \log m\right)$$

Proof: We first consider the running time of each iteration of the algorithm during which node $\bar{k}$ and associated primary-backup paths pairs P_i, Q_j are chosen to send traffic. Computation of $\min_{P \in \mathcal{PB}_{ij}} \Psi(z, w, P)$ for all $i, j \in N$ requires $n(n-1)$ calls to the oracle for the SBPR-DISJOINT-PATHS problem. This takes a total of $n(n-1)T_{DP}(n,m) = O(n^2 T_{DP}(n,m))$ time. It can be verified that the time taken for all other computations is $O(n^2 m)$. This is subsumed by the time taken for the computation by the oracle, since we can assume that $T_{DP}(n,m) = \Omega(m)$. Thus, the running time per iteration is $O(n^2 T_{DP}(n,m))$.

We next estimate the number of iterations before the algorithm terminates. Recall that in each iteration, traffic is sent along primary-backup path pairs P_i, Q_j corresponding to the maximum value of intermediate node split ratio α such that both the working traffic $\Delta(e)$ for link e under normal (no-failure) conditions and the traffic $\Delta'(e, \tilde{f})$ on link e after failure of some other link $\tilde{f}$ corresponding to that iteration are at most u_e.

Thus, for at least one link e, either the value $\Delta(e)$ or the value $\Delta'(e, \tilde{f})$ equals u_e and the weight $z(e)$ or $w(e, \tilde{f})$ respectively increases by a factor of $1+\epsilon$. Accordingly, with each iteration, we can associate a weight $z(e)$ or $w(e, \tilde{f})$ which increases by a factor of $1+\epsilon$.

Consider the weight $z(e)$ for fixed $e \in E$. Since $z_0(e) = \frac{\delta}{u_e}$ and $z_L(e) < \frac{1+\epsilon}{u_e}$ (as deduced in the proof of Lemma 7.5.4), the maximum number of times that this weight can be associated with any iteration is

$$\log_{1+\epsilon} \frac{1+\epsilon}{\delta} = \frac{1}{\epsilon}(1 + \log_{1+\epsilon} \frac{m^2}{2}) = O(\frac{1}{\epsilon} \log_{1+\epsilon} \frac{m^2}{2})$$

Using the same reasoning, it follows that the maximum number of times that the weight $w(e, \tilde{f})$ (for fixed $e, f \in E, \tilde{e} \neq \tilde{f}$) can be associated with any iteration is the same quantity.

Since there are a total of $m + m(\frac{m}{2} - 1)$ weights $z(e)$ and $w(e, \tilde{f})$, hence the total number of iterations is upper bounded by

$$O\left(\frac{1}{\epsilon}(m + m(\frac{m}{2} - 1)) \log_{1+\epsilon} \frac{m^2}{2}\right) = O\left(\frac{1}{\epsilon} m^2 \log_{1+\epsilon} \frac{m^2}{2}\right)$$

Multiplying this by the running time per iteration, we obtain the overall algorithm running time as $O(\frac{1}{\epsilon} n^2 m^2 T_{DP}(n, m) \log_{1+\epsilon} \frac{m^2}{2}) = O(\frac{1}{\epsilon} n^2 m^2 T_{DP}(n, m) \log_{1+\epsilon} m)$. Since $\ln(1 + \epsilon) = \Theta(\epsilon)$, this is $O(\frac{1}{\epsilon^2} n^2 m^2 T_{DP}(n, m) \log m)$. ■

7.6 Evaluation on ISP Topologies

In this section, we compare the throughput performance of the three mechanisms for protecting against link failures – local restoration, K-route path restoration, and shared backup path restoration – with that of the unprotected case for two-phase routing. Because two of the three restoration mechanisms do not have polynomial size linear programming formulations for maximizing throughput for two-phase routing, we use the combinatorial algorithms in all cases in order to make fair comparisons. We run the algorithms so as to provide solutions up to 5% of optimality. For maximum throughput two-phase routing with shared backup path restoration, we obtained solutions within 5% of optimality by using the combinatorial algorithm together with the heuristic in Section 7.5.5 as the oracle for SBPR-DISJOINT-PATHS. For the throughput of the unprotected scheme, we use the numbers reported in Chapter 4.

7.6.1 Topologies and Link/Ingress-Egress Capacities

We use the six ISP maps from the Rocketfuel dataset which had accompanying (deduced) OSPF/IS-IS weights [SMWH, SMW02, MSWA02]. These topologies list multiple intra-PoP (Point of Presence) routers and/or multiple intra-city PoPs as individual nodes. We coalesced such nodes so that nodes correspond to cities and the

Topology	Routers (original)	Links (inter-router)	PoPs (coalesced)	Links (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 7.1: Rocketfuel topologies with AS number and name. The table lists the original number of routers and inter-router links, and the number of coalesced PoPs and inter-PoP links.

topology represents geographical PoP-to-PoP ISP topologies. Some data about the original topologies and their coalesced versions is listed in Table 7.1.

The topologies provided by Rocketfuel did not include the capacities of the links, which were needed for our study. The Rocketfuel maps did include derived OSPF/ISIS weights of links, which were computed to match observed routes. In the absence of any other information on capacities, we need a way to deduce the link capacities from the weights. For this purpose, we assumed that the given link weights are the Cisco default setting for OSPF weights, i.e., inversely-proportional to the link capacities [Cisco97]. The link capacities obtained in this manner turned out to be symmetric, i.e., $u_{ij} = u_{ji}$ for all $(i, j) \in E$.

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.

7.6.2 Experiments and Results

We denote the throughput values for the unprotected and link failure protected versions of two-phase routing as follows: (i) λ_{UNP} for unprotected, (ii) λ_{LR} for local restoration, (iii) λ_{KPR} for K-route path restoration, and (iv) λ_{SBPR} for shared backup path restoration. We are also interested in the number of intermediate nodes i with $\alpha_i > 0$, which we denote for the four cases by N_{UNP}, N_{LR}, N_{KPR} and N_{SBPR} respectively.

Throughput

In Table 7.2, we list the throughput values for the three protection schemes with respect to that for unprotected two-phase routing for the six Rocketfuel topologies.

The overhead of protecting against link failures can be measured by the percentage decrease in network throughput over that for the unprotected case. For local restoration, this is $O_{LR} = \frac{\lambda_{UNP}-\lambda_{LR}}{\lambda_{UNP}}$. For K-route path restoration, this is $O_{KPR} = \frac{\lambda_{UNP}-\lambda_{KPR}}{\lambda_{UNP}}$. For shared backup path restoration, this is $O_{SBPR} = \frac{\lambda_{UNP}-\lambda_{SBPR}}{\lambda_{UNP}}$. These values are listed in Table 7.3.

For local restoration and shared backup path restoration, the overhead ranges from 35-60% for the six topologies. For K-route path restoration, the overhead ranges from 45-60% for the six topologies. All three overheads are relatively high because

Topology	$\frac{\lambda_{LR}}{\lambda_{UNP}}$	$\frac{\lambda_{KPR}}{\lambda_{UNP}}$	$\frac{\lambda_{SBPR}}{\lambda_{UNP}}$
Telstra (Australia) 1221	0.4422	0.4415	0.4554
Sprintlink (US) 1239	0.6214	0.5600	0.6182
Ebone (Europe) 1755	0.3855	0.3763	0.3894
Tiscali (Europe) 3257	0.6294	0.5328	0.6459
Exodus (Europe) 3967	0.5461	0.5390	0.5592
Abovenet (US) 6461	0.4319	0.4184	0.4492

Table 7.2: Throughput of Two-Phase Routing with Local Restoration (λ_{LR}), K-Route Path Restoration (λ_{KPR}), and Shared Backup Path Restoration (λ_{SBPR}) compared to unprotected case (λ_{UNP}).

Topology	O_{LR}	O_{KPR}	O_{SBPR}
Telstra (Australia) 1221	55.78%	55.85%	54.46%
Sprintlink (US) 1239	37.86%	44.00%	38.18%
Ebone (Europe) 1755	61.45%	62.37%	61.06%
Tiscali (Europe) 3257	37.06%	46.72%	35.41%
Exodus (Europe) 3967	45.39%	46.10%	44.08%
Abovenet (US) 6461	56.81%	58.16%	55.08%

Table 7.3: Overhead of Local Restoration (O_{LR}), K-Route Path Restoration (O_{KPR}), and Shared Backup Path Restoration (O_{SBPR}) compared to unprotected case for Two-Phase Routing.

of the limited diversity available in these six topologies. The general trend for the six topologies is that O_{LR} is comparable to O_{SBPR}. For some of the topologies, both of these are appreciably lower than O_{KPR} (while comparable in other cases). K-route path restoration is more constrained by the physical diversity of the network than the other two restoration mechanisms because of the following two reasons: (a) sharing of backup capacity increases with more link-disjoint paths between a given pair of nodes, and (b) backup capacity is shared only among paths within the same connection and not with paths belonging to other connections. Hence, the increased overhead of K-route path restoration.

Topology	N_{UNP}	N_{LR}	N_{KPR}	N_{SBPR}
Telstra (Australia) 1221	1	1	1	1
Sprintlink (US) 1239	5	6	4	7
Ebone (Europe) 1755	4	5	3	5
Tiscali (Europe) 3257	7	6	2	7
Exodus (Europe) 3967	3	7	3	9
Abovenet (US) 6461	7	6	1	6

Table 7.4: Number of Intermediate Nodes in Two-Phase Routing for unprotected case (N_{UNP}), and with Local Restoration (N_{LR}), K-Route Path Restoration (N_{KPR}), and Shared Backup Path Restoration (N_{SBPR}).

Number of Intermediate Nodes

In Table 7.4, we list the number of intermediate nodes with non-zero traffic split ratios for the four cases for the six Rocketfuel topologies. In our experiments for the three restoration schemes, we observed that some intermediate nodes have quite small α_i values – they carry less than a percentage point of total network traffic. In practice, the traffic split ratios associated with these intermediate nodes can be redistributed to other nodes without any significant decrease in throughput. Hence, in Table 7.4, for the N_{LR}, N_{KPR}, and N_{SBPR} values, we list the number of intermediate nodes with the largest α_i values (normalized) that sum to at least 0.95, i.e., the nodes with largest traffic split ratios that together carry at least 95% of total network traffic. Similar to that for the unprotected case, the number of intermediate nodes with each restoration mechanism is a small fraction of the total number of nodes.

The number of intermediate nodes for local restoration and shared backup path restoration is about the same as that for the unprotected case for all the topologies. However, we observe a marked decrease in the number of intermediate node for K-route path restoration. This is again because of the limited diversity available in the six topologies. For K-route path restoration, the algorithm intelligently selects only those intermediate nodes that have sufficient path diversity to many other nodes in the network, since sharing of backup bandwidth is only across paths belonging to the

same connection and can be increased only by having more mutually diverse paths between a given pair of nodes. In contrast, local 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. Similarly, for shared backup path restoration, higher levels of diversity are not required because every connection consists of exactly two mutually diverse paths. For both local restoration and shared backup path restoration, sharing of backup bandwidth occurs across different connections, hence more opportunities exist for such sharing.

Chapter 8

Two-Phase Routing in Wireless Mesh Networks

In this chapter, we consider the potential applicability of two-phase routing in handling variable traffic in wireless mesh networks (WMNs) and avoiding the difficulty of distributed dynamic routing and resource allocation in such networks. This application imposes significant additional constraints on two-phase routing so as to merit separate investigation. We extend our optimization framework for maximum throughput two-phase routing in wired networks to handle routing and scheduling constraints that are peculiar to WMNs and arise from the requirement to handle *radio transmit/receive diversity* and the phenomenon of *wireless link interference.*

It is difficult to provide (theoretical) bandwidth guarantees for variable traffic in WMNs. The main reason for this is that the dynamic MAC and routing protocols in such networks are limited by local knowledge to do link transmission scheduling and packet forwarding – this leads to inefficient use of physical layer resources. We propose application of the two-phase routing scheme to WMNs so as to avoid this difficulty of distributed routing and scheduling while providing throughput guarantees for variable traffic subject to ingress-egress capacity constraints at each WMN node. The routing scheme is also well-suited for providing throughput guaranteed rendezvous

based services in specialized service overlay models where the final destination of the traffic is not known at the source.

The optimization framework for two-phase routing needs to handle two aspects in WMNs related to link transmission scheduling, namely (i) mutually exclusive transmit/receive nature of communication (for a single radio), also called transmit/receive diversity, and (ii) link interference for omnidirectional antenna. Link transmission scheduling is equivalent to a graph edge coloring problem which is $\mathcal{NP}$-hard. We use the linear relaxation of the scheduling constraints associated with (i) and (ii) above and incorporate them as link utilization constraints into our earlier linear programming formulations for two-phase routing in wired networks. Our overall approach is to first solve the routing problem after incorporating scheduling constraints into it and then schedule the link transmissions for the obtained link data rates.

For the case of narrow beam forming (directional) antennae in which link interference can be ignored, we design a combinatorial algorithm with performance guarantees. For the case of omnidirectional antennae, we model link interference using link utilization constraints corresponding to cliques and independent sets in the conflict graph. Clique constraints provide an upper bound on the maximum throughput (which may not always be achievable), while independent set constraints provide an achievable lower bound. Our combinatorial algorithm for the narrow beam antenna case can accommodate clique constraints and hence be used to provide upper bounds on throughput for the omnidirectional antenna case. We show how our optimization framework can be generalized to handle multiple wireless channels and multiple radios.

We investigate the performance of two-phase routing in WMNs under variability of many of the parameters involved, e.g., transmission power, introduction of relay nodes, and link interference. We also compare the throughput performance of two-phase routing in WMNS with that of the optimal scheme that can change the routing with changes in the traffic.

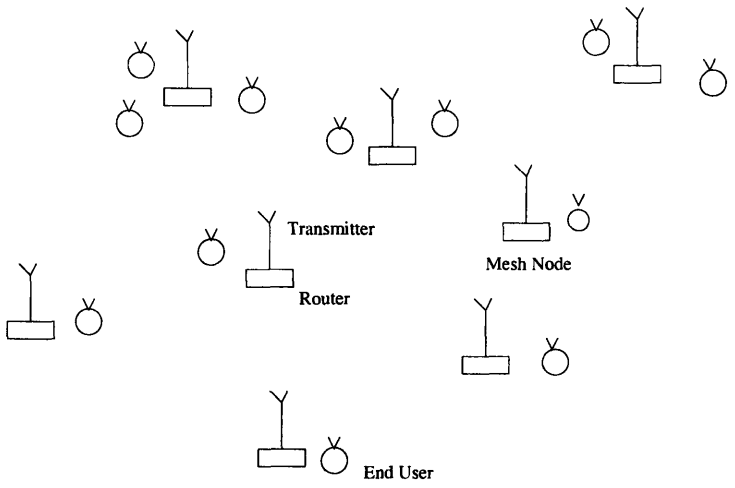


Figure 8-1: Wireless Mesh Network.

8.1 Wireless Mesh Networks

The widespread deployment of WiFi networks and the falling price of wireless components has lead to increased interest in multihop wireless networks [MNS04, BCG05]. These wireless mesh networks comprise of a set of nodes that transmit and receive packets, and also perform the tasks of routing, scheduling, and channel assignment (See Figure 8-1). These nodes are static (or minimally mobile) and have a power source unlike ad-hoc networks. End users (who may be mobile) are homed to one of these mesh network nodes via a wireless link through end-user devices like laptops and PDAs. If two end users want to communicate, then packets are sent from the originating end user to a node in the WMN where it is currently homed. This packet is subsequently routed through the network using multiple hops until it reaches the homing node of the destination. This node then transmits the packet to the destination end user.

Multi-hop wireless mesh networks have recently been of much research interest due to their lowered need for wired infrastructure support and due to envisaged new applications like community wireless networks and in networks that require rapid deployability. There are several proposals to use WMN to increase the capacity and

coverage of traditional cellular networks. Though operationally a WMN is similar to a wired backbone network, the presence of wireless links present several unique challenges.

Most important among these is the phenomenon of *link interference.* In a wired network, every link may be utilized simultaneously up to its maximum capacity. In wireless networks, geographically proximal links interfere with each other, thus allowing only one of every mutually interfering link pair to be active at any one time. Moreover, the interference range is typically much larger than the transmission range. This reduces the achievable throughput of the network.

The research challenges currently being addressed for WMNs include generating sufficient throughput from the system, guaranteeing some quality of service for the end users, and scaling the network to a large number of nodes. There has been activity across the different layers in the communication stack to modify existing protocols to make them more suited for WMNs. Some examples of the work at the different layers are as follows:

- *Improved Physical Layer Design:* One way to improve the throughput of WMNs is to have a better physical layer, including using multiple channels, multiple radios, multiple antennae [FG98] with space-time coding [A98], as well as using directional antennae [SR03] to minimize interference.

- *Redesigned MAC:* Standard single hop MAC like CSMA-CA (IEEE 802.11 MAC) does not work well in multi-hop networks. There have been efforts to redesign the MAC layer for multi-hop networks. An alternate approach that has been attempted is to design a MAC that is appropriate and tuned to the underlying physical layer. For example, the design of a MAC that accounts for directional antennae in [KSV00].

- *New Routing Protocols:* Routing protocols for adhoc networks have been an active area of research for more than a decade. A lot of the mechanisms developed

for these routing algorithms have been to cope with mobility. Since the nodes in WMN are stationary, there has been more recent work on paring down these protocols to make them suitable for WMNs. In addition, there has also been increased interest in developing link quality based routing [DPZ04] as well new routing metrics [DABM03] that are appropriate for WMNs.

- *New Transport Layer Protocols:* There is a large body of literature on the performance problems faced by TCP on wireless links [BPSK97, XPMS01] since TCP does not differentiate between congestion and non-congestion losses. This has lead to the design of new TCP variants like ATP [SAHS03] for adhoc networks.

In spite of the tremendous amount of progress at getting increased data rates at the physical layer, this has not translated into increased throughput at the application layer in multihop wireless networks. A significant reason is that the MAC and routing protocols have to be dynamic to adapt to different traffic patterns that arise in the network. These dynamic MAC and routing protocols use limited local knowledge to do conflict resolution and packet forwarding which leads to inefficient use of the physical layer resources.

If the traffic between the nodes in the network is known and fixed, then there is no need for dynamic routing and link transmission scheduling. The routing can be pre-computed and a link transmission schedule can be determined to meet the bandwidth requirements of the routing protocol. This link transmission schedule can be fixed and downloaded to the nodes. Since the computation is done offline, we can use a centralized, optimal algorithm to maximize the throughput of the network.

However, the traffic between the nodes in the network is variable and mostly unknown. The MAC and routing protocols have to adapt to changing traffic patterns. The inefficiencies in the MAC and routing protocols hve meant that WMNs still cannot meet even basic end-to-end Quality-of-Service (QoS) guarantee for unicast connections. This is especially true in moderate sized networks where the connection has to traverse a few hops on the way to the destination.

Two-phase routing can be adapted to handle variable traffic in WMNs and avoid the difficulty of distributed dynamic resource allocation. Because the Phase 1 and Phase 2 paths and their bandwidths are fixed in two-phase routing, their routing and the resulting link transmission schedule can be computed a priori and uploaded to the WMN nodes. Two-phase routing, when applied to WMNs, provides a routing architecture for WMNs that can meet the following requirements:

- Eliminate the need for a dynamic MAC and routing protocol but instead rely on a static, optimized routing and link scheduling to handle variable traffic in the network efficiently.
- Provide source-destination oblivious throughput guarantees to ingress-egress traffic at each WMN node for both unicast and multicast traffic permissible within the network's end-user spectrum bandwidth constraints. This can then be used by higher layers to provide QoS guarantees to applications.
- Support indirection to provide rendezvous based services in specialized overlay models like i3.

To our knowledge, our approach is the first to provide throughput guarantees in a WMN that handles variable traffic and meets the above requirements. In the next section, we describe basic services that are desirable on a WMN along with associated QoS guarantees.

8.2 Basic Services on WMNs

We describe some basic services on which more advanced WMN applications can be built. All the basic services that we outline below have (end-to-end) bandwidth guarantees associated with them. Applications built on top of these basic services can use these bandwidth guarantees to provide end-to-end QoS guarantees to the applications. There are two classes of basic services that we provide on the WMNs.

The first class is end-to-end bandwidth guaranteed services where the destination is known at the source, and the second class is bandwidth guaranteed rendezvous based communication services where the destination is not known to the source. Two-phase routing permits the implementation of both these classes either independently or concurrently in a single WMN.

8.2.1 End-to-end Bandwidth Guaranteed Services

The end-to-end bandwidth guaranteed services provide basic primitives over which more complex applications can implemented. By making assumptions about the size of the buffers at the nodes in the network, it is possible to translate this bandwidth requirement into delay or loss requirements. We would like to provide the following two bandwidth guaranteed services.

- *Bandwidth Guaranteed Unicast:* We would like the WMN to support bandwidth guaranteed point to point traffic. In the rest of the chapter, we assume that all connections are unidirectional. The algorithms developed can be extended to handle bi-directional connections.

- *Bandwidth Guaranteed Multicast:* In addition to bandwidth guarantees on unicast traffic, we would like the WMN to support bandwidth guaranteed multicast services.

The total amount of bandwidth guaranteed connections that can originate or terminate at any WMN node in the network is naturally constrained by the end-user spectrum capacity available at that node.

8.2.2 Bandwidth Guaranteed Rendezvous Based Services

In rendezvous based communication, the source and the destination of the traffic are decoupled. This eases the deployment of services like mobility, multicast, and anycast

on WMNs. This rendezvous-based communication abstraction through indirection is inspired by the Internet indirection infrastructure (i3) [SAZSS02] which we described in Section 1.5.2. Sources send packets to rendezvous nodes using a logical identifier. Receivers express interest in packets sent to a specific identifier. The rendezvous nodes forward packets to all receivers that express interest in a particular identifier. This is illustrated in Figure 1-4.

The communication between senders and receivers is through these rendezvous points over the WMN. Since the scope of implementation of this service is only on a single WMN and not Internet wide, we do not need a peer-to-peer routing protocol to route packets to and from the rendezvous points. Two-phase routing locates these rendezvous points optimally at the intermediate nodes and also gives the paths to and from all nodes in the WMN to these rendezvous points to maximize network throughput. Unlike i3 which is a best effort service, two-phase routing can support *rendezvous based services with bandwidth guarantees* in a manner analogous to that for wired networks as described in Section 2.2.2.

8.3 Modeling Assumptions

In this section, we outline the modeling assumptions which are made in this chapter and their implications on the problem formulation and solution. We first introduce some notation and interpret the traffic variation model from Section 1.4 in the context of WMNs. We then give details of the communication model and describe the resource sharing model.

8.3.1 Notation

The WMN topology, given by the nodes and the links corresponding to pairs of nodes within direct communication range, is modeled as the *topology graph* $G = (N, E)$ with node set N and (directed) link set E. Each node in the network can be a source

or destination of traffic. Let $|N| = n$ and $|E| = m$. The nodes in N are labeled $\{1, 2, \ldots, n\}$. The sets of incoming and outgoing edges at node i are denoted by $E^-(i)$ and $E^+(i)$ respectively. Given any node i in the network, let $N(i)$ denote the set of links whose transmitter or receiver is node i, i.e., $N(i) = E^+(i) \cup E^-(i)$. 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). Let u_e be the maximum data rate achievable on link e. The *utilization* of a link is defined as the traffic on the link divided by its capacity.

Because of the bi-directional nature of wireless links, the graph G is actually bi-directed, i.e., whenever there is a link $(i, j) \in E$, there is also the link $(j, i) \in E$. Each forward and reverse link also has the same capacity. However, we do not need to make use of the bi-directionality of links in our optimization framework.

8.3.2 Traffic Variation Model

The total amount of traffic that enters (leaves) an ingress (egress) node in the WMN is bounded by the total end-user spectrum capacity available at that node. These form the natural ingress-egress constraints for traffic between end-user devices that need to be routed by the WMN. Using the same notation as before, we denote the upper bounds on the total amount of traffic entering and leaving at WMN node i by R_i and C_i respectively. In the case that one or more WMN nodes connect to Internet access points, the ingress-egress capacity refers to the aggregate wired connectivity to external networks.

The point-to-point matrix for the traffic carried by the WMN 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 WMN, 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

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

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

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

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

8.3.3 Communication Model

We make a distinction between the links that form the WMN and provide connectivity between adjacent WMN nodes and the links that connect the end-user to a node in the WMN. This distinction is similar to the distinction made between the core network links and ingress-egress links in a wired network. We call the links between WMN nodes the *network links* and the links to and from the end-user to a WMN node as the *user links*. The user links comprise the *uplink* from the end-user to the WMN node and the *downlink* from the WMN node to the end-user.

We assume that the WMN is a time-slotted system where all nodes share the same frequency spectrum. Therefore, one can view the WMN as a hybrid CDMA-TDMA system. In addition, we assume that the system is synchronous. In each time slot, a subset of the links are active. The rate that can be achieved on a given link in any given time slot is a function of the signal to interference and noise ratio (SINR) at the receiver. The objective of the MAC layer, or the link transmission scheduling algorithm, is to get a high rate on the active links by ensuring that active links do not have too much interference. The set of transmissions that interfere with a given link

depends on the physical layer characteristics and transmission power. In particular, it depends on the type of transmission antennae used.

- *Narrow Beam Forming (Directional) Antennae:* If the links are realized by a narrow beam antenna, then the interference caused by any active link on any other link can be assumed to be negligible. In this case, the only requirement that is imposed on the system is that a node cannot simultaneously send and/or receive data from multiple nodes in the same time slot. (This is in an ideal beam forming system. In practical systems, it is typically not easy to form beams less than 5-10 degrees, so there will be some interference from neighboring links).

- *Omnidirectional Antennae:* If the transmission is omnidirectional, then all the transmissions in the vicinity of the receiver of link e interfere with transmissions on link e. If there is no effort to avoid this interference, then the data rate that can be achieved will be quite poor. Therefore, we need a MAC to quieten the transmitters which can interfere strongly with link e. This MAC ensures that all nodes in the vicinity of the receiver of the link cannot be transmitting when the link is active. If this condition is met, then the rate that can be achieved on the link is primarily a function of the power and the channel gain of the link.

The applicability of two-phase routing to WMNs does not depend on the particular type of antennae used for wireless communication. The phenomenon of interference for omnidirectional antennae needs to be effectively handled in the optimization framework so that a priori throughput guarantees can be computed and guaranteed. The routing optimization methodology needs to be aware of and model the constraints imposed on link transmission scheduling by the link interference phenomenon.

For developing our methods for maximum throughput two-phase routing in WMNs, we first assume, for simplicity, that narrow beam forming antennae are used for communication so that we need to handle only the mutually exclusive transmit/receive nature of communication at a node. Subsequently, we consider the more compli-

cated omnidirectional antenna case and discuss how to handle link interference in our optimization framework. We assume a non-fading channel with a distance related channel attenuation, and that all nodes transmit at some fixed power (that can vary from node to node). This implies that the maximum achievable data rate u_e on each link e is fixed a priori.

8.3.4 Resource Sharing Model

A node in the WMN shares spectrum and radio resources between the end-users and the network links. To deliver packets in WMNs, each node has to commit some of its radio and spectrum resources to creating network links. The remainder of the resources are made available for use by end-users who are attached to the WMN node. Of the resources that are made available to the end users, some fraction of it is used for uplink transmission from the end-users to the network node and the rest of the resource is used for downlink transmission from the node to end users. The ingress-egress bounds on traffic between end-users at a WMN node depends only the aggregate capacity of user links at that node. Thus, it is essential to decouple the user links and the network links. This decoupling is done via the utilization commitment vector.

Node Commitment to the Network

We assume that a WMN node i commits a fraction of at most η_i of its utilization to support network links. In other words, node i spends a fraction at most η_i of its time to transmitting on the network links. We refer to the vector $\vec{\eta}$ as the *utilization commitment vector*.

- We assume that the network links get precedence in the sense that the network link scheduling is done prior to the end user scheduling. However, the network links will not use more than a fraction η_i of the radio spectrum at node i.

- In case the spectrum and radio resources for the network links at node i are not shared with the spectrum and radio resources for the end-users, then $\eta_i = 1$ for that node.

Network Guarantee to the Nodes

In exchange for routing traffic on the WMN, the network guarantees node i the transport of an ingress data rate of R_i from this node and an egress data rate of C_i to this node. *This data rate guarantee is independent of destination of the outgoing traffic and the source of the incoming traffic.* We refer to the vector $(\vec{R}, \vec{C})$ as the *data rate guarantee vector.* Given a particular network configuration and utilization commitment vector, a given data rate guarantee vector may or may not be achievable. We formulate the *throughput maximization* problem as one of determining the largest scalar λ such that the data rates $\lambda(R, C)$ are achievable (equivalently, all traffic matrices in $\mathcal{T}(\vec{R}, \vec{C})$ can be feasibly routed by the network). The given data rate vector is achievable if and only if the maximum achievable throughput λ is at least 1.

8.4 Two-Phase Routing in Wireless Mesh Networks

In this section, we consider the adaptation of two-phase routing to WMNs so as to meet the requirements of both bandwidth guaranteed and rendezvous-based services while also enabling the network to accommodate arbitrary (and possibly rapidly changing) traffic demands without dynamic adaptation of network routing and link transmission scheduling. We are given the ingress-egress traffic bounds R_i, C_i and utilization commitment η_i for all WMN nodes. Whether these rates are achievable or not will be determined by the optimization methods developed in the rest of the chapter. The description of the scheme is similar to that in Section 2.1 for the wired network case. We summarize the main aspects and then discuss some implementation issues for WMNs.

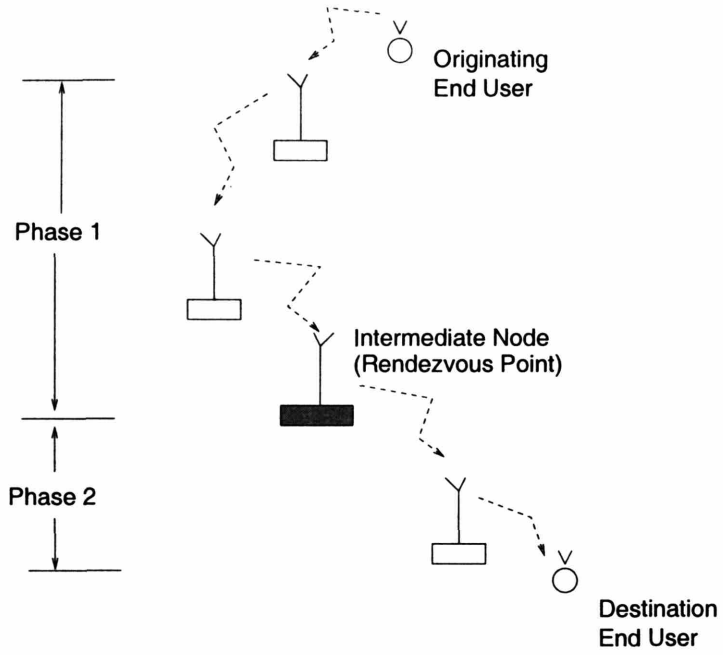


Figure 8-2: Two-Phase Routing in Wireless Mesh Networks.

The two-phase routing scheme works in a WMN as follows:

- **Phase 1:** A predetermined fraction α_j of the traffic entering the WMN 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 WMN node receives traffic destined for different destinations that it routes to their respective destinations in this phase.

This is illustrated in Figure 8-2. The maximum demand from node i to node j as a result of routing in Phases 1 and 2 is $\alpha_j R_i + \alpha_i C_j$ and does not depend on the matrix $T \in \mathcal{T}(\vec{R}, \vec{C})$.

An instance of the scheme requires specification of the traffic split ratios $\alpha_1, \alpha_2, \ldots, \alpha_n$ and routing of the Phase 1 and Phase 2 paths subject to additional scheduling constraints imposed by wireless communication, namely, (i) mutually exclusive trans-

mit/receive nature of communication at a node, and (ii) link interference for the omnidirectional antenna case. In Sections 8.5 and 8.6, we consider computing these so as to maximize network throughput. Similar to the wired network case, the Phase 1 and Phase 2 paths may need to be implemented through multi-path routing in order to maximize throughput.

Implementing the Routing

We consider how to implement the routing in a WMN. Using the throughput maximization methods of Sections 8.5 and 8.6, assume that we have computed the locations of the intermediate nodes, the fraction of traffic that is routed from a source node to each of these intermediate nodes, and the routing of Phase 1 and Phase 2 paths to and from each ingress-egress node to the intermediate nodes along with the associated bandwidths. In order to ensure that maximum throughput is attained by the WMN, traffic has to be routed along these paths respecting the bandwidths that have been computed.

The routing is implemented as follows: When a packet arrives at a source node, it determines which intermediate node has to receive this packet. Intermediate node i receives a fraction α_i of the packets. The intermediate node i for the current packet is chosen with probability α_i either randomly or in a weighted round robin fashion. The paths and the bandwidths to intermediate node i are known. The source node now picks the path of this packet respecting the bandwidth for the path. It then appends the path to the packet and the packet is source-routed to the intermediate node. This process is repeated by the intermediate node in order to route the packet to the ultimate destination. The intermediate node is also responsible for replicating the packet if the packet has to be multicast to several destinations.

A suitable protocol for source-routing in wireless networks can be adapted for this purpose. Once such protocol that is being standardized in the IETF is Dynamic Source Routing Protocol (DSR) [JMB01, JMH04]. *The DSR protocol allows multiple*

routes to any destination and allows each sender to select and control the routes used in routing its packets.

8.5 Maximizing Throughput

Given the maximum achievable link rates u_e, the desired ingress-egress data rate guarantee vector $(\vec{R}, \vec{C})$, and the node commitment vector $\vec{\eta}$, the objective now is to determine the largest λ (throughput) such that the data rate vector $\lambda(\vec{R}, \vec{C})$ is achievable. For this value of λ, node i is guaranteed an ingress data rate of λR_i and an egress data rate of λC_i to and from end-user devices. We first consider the directional antenna case where their is no link interference – we need to handle only the mutually exclusive transmit/receive nature of communication at a WMN node. Subsequently, we extend the approach so as to handle link interference for omnidirectional antennae.

As for the wired network case, we relax the requirement that the traffic split ratios α_i 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 divided by λ (normalized) so that they sum to 1, in which case all matrices in $\lambda \cdot \mathcal{T}(\vec{R}, \vec{C})$ 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.

8.5.1 Link Flow Based LP Formulation

Our overall approach is to first solve the routing problem after incorporating scheduling constraints into it and then schedule the link transmissions for the obtained link

data rates. Link transmission scheduling onto discrete time slots is equivalent to a graph edge coloring problem which is $\mathcal{NP}$-hard (this is discussed in Section 8.7). We use the linear relaxation of the scheduling constraints and incorporate them as link utilization constraints into our earlier linear programming formulations for two-phase routing in wired networks.

Let x_e^{ij} denote the data rate on link e for routing $\alpha_j R_i + \alpha_i C_j$ amount of flow from node i to node j. Then, the total data rate on link e is

$$x_e = \sum_{i,j \in N} x_e^{ij} \quad \forall\, e \in E$$

Then, the fraction of time link e has to be active is $\frac{x_e}{u_e}$. For any node i, since any two links in $N(i)$ cannot be simultaneously active and since the total utilization of the network links for node i must be at most η_i, it is easy to see that

$$\sum_{e \in N(i)} \frac{x_e}{u_e} \leq \eta_i \quad \forall\, i \in N$$

is a necessary condition for the link rates to be schedulable. For the directional antenna case, it can be shown using Shannon's coloring theorem that

$$\sum_{e \in N(i)} \frac{x_e}{u_e} \leq \frac{2}{3}\eta_i \quad \forall\, i \in N$$

is a sufficient condition for the data rates x_e to be achievable after scheduling [HS88, KN03].

Our basic approach is to use the necessary condition and solve the routing problem. We then use edge coloring as described in Section 8.7 to schedule the links in the network. If the edge coloring indicates that the data rates obtained after solving the routing problem are not achievable, then we scale down the throughput accordingly so that the scaled down data rates are achievable.

In order to derive more complex (but linear) necessary conditions for handling link

interference in the case of omnidirectional antennae, we will write the above necessary conditions in the following general form:

$$\sum_{e \in S} \frac{x_e}{u_e} \leq \nu(S)$$

where S is some arbitrary subset of links in E and $\nu(S)$ is a "total utilization" value associated with link set S. Let $\mathcal{S}$ denote the collection of such subsets $S \subseteq E$ for which the above utilization constraints exist. If every link e belongs to some set S in the collection $\mathcal{S}$ with $\nu(S) \leq 1$, then the link capacity constraint $x_e \leq u_e$ is subsumed by link utilization constraints for S. Otherwise, we can add the set $S = \{e\}$ with $\nu(S) = 1$ to $\mathcal{S}$. Thus, we need not consider explicit link capacity constraints.

Then, the problem of two-phase routing so as to maximize throughput for the directional antenna case can be written as the following link flow based linear program:

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

subject to

$$\sum_{e \in E^+(k)} x_e^{ij} - \sum_{e \in E^-(k)} x_e^{ij} = \begin{cases} \alpha_j R_i + \alpha_i C_j & \text{if } k = i \\ -\alpha_j R_i - \alpha_i C_j & \text{if } k = j \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i, j, k \in N \tag{8.1}$$

$$\sum_{e \in S} \frac{1}{u_e} \sum_{i,j \in N} x_e^{ij} \leq \nu(S) \;\; \forall\, S \in \mathcal{S} \tag{8.2}$$

$$\alpha_i \geq 0 \;\; \forall\, i \in N \tag{8.3}$$

$$x_e^{ij} \geq 0 \;\; \forall\, e \in E, \;\; \forall\, i, j \in N \tag{8.4}$$

Constraints (8.1) correspond to the routing of $\alpha_j R_i + \alpha_i C_j$ amount of flow from node i to node j. For constraints (8.2), the collection $\mathcal{S}$ of link subsets for the directional antenna case is $\mathcal{S} = \{N(i) \,|i \in N\}$ so that $|\mathcal{S}| = n$. Also, $\nu(N(i)) = \eta_i \;\; \forall\, i \in N$.

These model the mutually exclusive transmit/receive nature of communication at a node and the commitment at each WMN node to transmit at most η_i of the time on network links.

By using per-source flow variables x_e^i instead of per source-destination variables x_e^{ij} , the number of x variables in the above linear program can be reduced to nm. The number of link utilization constraints is equal to $|\mathcal{S}| = n$ for the directional antenna case and hence the linear program can be solved in polynomial time using a general linear programming algorithm [S86].

However, as we will see for the omnidirectional antenna case in Section 8.6, $|\mathcal{S}|$ is large and could be exponential in the network size in the worst case. Moreover, in practice, the running time of general linear programming based algorithms do not scale well with increase in number of constraints (and, number of variables). The primal-dual combinatorial algorithm that we now develop for solving the above linear program has running times that scale well to handle the large number of link utilization constraints for the omnidirectional antenna case.

8.5.2 Path Flow Based LP Formulation

In this section, we develop a path indexed linear programming formulation for the above problem. This will be subsequently used to develop the fast combinatorial algorithm in Section 8.5.4.

Let $\mathcal{P}_{ij}$ denote the set of all paths from node i to node j. Let $x(P)$ denote the traffic on path P. Then, the problem of two-phase routing in WMNs so as to maximize throughput for the directional antenna case can be expressed as the following path indexed linear program:

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

subject to

$$\sum_{P \in \mathcal{P}_{ij}} x(P) = \alpha_j R_i + \alpha_i C_j \quad \forall\, i, j \in N \tag{8.5}$$

$$\sum_{e \in S} \frac{1}{u_e} \sum_{P \ni e} x(P) \leq \nu(S) \quad \forall\, S \in \mathcal{S} \tag{8.6}$$

$$\alpha_i \geq 0 \quad \forall\, i \in N \tag{8.7}$$

$$x(P) \geq 0 \quad \forall\, P \in \mathcal{P}_{ij}, \quad \forall\, i, j \in N \tag{8.8}$$

In Section 8.5.3, we state the dual of the linear program. In general, a network can have an exponential number of paths (in the size of the network). Hence, the primal linear program can have possibly exponential number of variables and its dual can have an exponential number of constraints – they are both not suitable for solving the problem on medium to large sized networks. The usefulness of the primal and dual formulation is in designing a fast (polynomial time) combinatorial algorithm for the problem.

8.5.3 Dual of Path Flow Based LP Formulation

We will use a primal-dual approach to develop a fast combinatorial algorithm (FPTAS) for two-phase routing in WMN for the directional antena case that computes the traffic split ratios and routing Phase 1 and Phase 2 paths up to $(1+\epsilon)$-factor of the optimal objective function value (maximum throughput) for any $\epsilon > 0$. The primal-dual scheme extends the approach used for maximizing throughput for two-phase routing in wired networks in Section 4.3.

The dual formulation of the linear program in Section 8.5.2 associates a variable a variable π_{ij} with each demand constraint in (8.5), and a non-negative variable $w(S)$ with each link utilization constraint for link set S in (8.6). The dual program can be written as:

$$\text{minimize} \quad \sum_{S \in \mathcal{S}} \nu(S) w(S)$$

subject to

$$\sum_{e \in P} \frac{1}{u_e} \sum_{S \ni e} w(S) \geq \pi_{ij} \quad \forall\, P \in \mathcal{P}_{ij}, \quad \forall\, i, j \in N \tag{8.9}$$

$$\sum_{i \in N, i \neq k} R_i \pi_{ik} + \sum_{j \in N, j \neq k} C_j \pi_{kj} \geq 1 \quad \forall\, k \in N \tag{8.10}$$

$$w(S) \geq 0 \quad \forall\, S \in \mathcal{S} \tag{8.11}$$

Because of the nature of constraints (8.10), we can assume that the variables π_{ij} attain the maximum possible value given by constraints (8.9) in any optimal solution. Then, we have

$$\pi_{ij} = \min_{P \in \mathcal{P}_{ij}} \sum_{e \in P} \frac{1}{u_e} \sum_{S \ni e} w(S) \quad \forall\, i, j \in N$$

This allows us to eliminate the dual variables π_{ij}. The simplified dual problem can be written as:

$$\text{minimize} \quad \sum_{S \in \mathcal{S}} \nu(S) w(S)$$

subject to

$$\sum_{i \in N, i \neq k} R_i \min_{P \in \mathcal{P}_{ik}} \sum_{e \in P} \frac{1}{u_e} \sum_{S \ni e} w(S) + \sum_{j \in N, j \neq k} C_j \min_{P \in \mathcal{P}_{kj}} \sum_{e \in P} \frac{1}{u_e} \sum_{S \ni e} w(S) \geq 1 \quad \forall\, k \in N \tag{8.12}$$

$$w(S) \geq 0 \quad \forall\, S \in \mathcal{S} \tag{8.13}$$

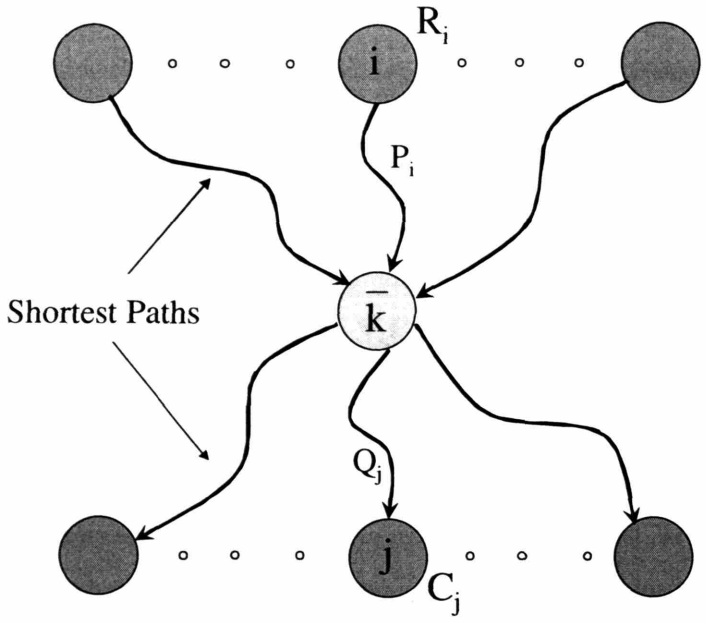


Figure 8-3: One Step in the Primal-Dual Computation for Two-Phase Routing in WMNs.

8.5.4 Combinatorial Algorithm

In this section, we use the primal and dual formulations of the path flow based linear program to develop a fast combinatorial algorithm for the problem.

For a given node k and weights $w(S)$, let $V(k)$ denote the LHS of constraint (8.12). Given the weights $w(S)$, the path $P \in \mathcal{P}_{ij}$ that minimizes $\sum_{e \in P} \frac{1}{u_e} \sum_{S \ni e} w(S)$ is actually the shortest path from node i to node j under link costs $c(e) = \frac{1}{u_e} \sum_{S \ni e} w(S)$ for all $e \in E$. Thus, the quantities $V(k)$, for all $k \in N$, can be computed in polynomial time using a single all-pairs shortest path computation.

The overall algorithm works as follows. Start with initial weights $w(S) = \frac{\delta}{\nu(S)}$ (the quantity δ depends on ϵ and is derived later). Repeat the following until the dual objective function value is greater than or equal to 1:

1. Compute the node $k = \bar{k}$ for which $V(k)$ is minimum. This identifies a node $\bar{k}$ as well as paths P_i from node i to node $\bar{k}$ for all $i \neq \bar{k}$ and paths Q_j from

node $\bar{k}$ to node j for all $j \neq \bar{k}$. (These are the shortest paths between respective nodes obtained during the computation of $V(k)$ as described above.) This is illustrated in Figure 8-3.

2. For a split ratio of 1 for intermediate node $\bar{k}$, the traffic on path P_i is R_i for all $i \neq \bar{k}$ and the traffic on path Q_j is C_j for all $j \neq \bar{k}$. Using this, compute the traffic $\Delta(e)$ on link e per unit split ratio $\alpha_{\bar{k}}$ for intermediate node $\bar{k}$ as

$$\Delta(e) = \sum_{i \neq \bar{k}, P_i \ni e} R_i + \sum_{j \neq \bar{k}, Q_j \ni e} C_j \quad \forall\, e \in E \tag{8.14}$$

3. For this traffic, compute the total utilization for link set S as

$$U(S) = \sum_{e \in S} \frac{\Delta(e)}{u_e} \quad \forall\, S \in \mathcal{S} \tag{8.15}$$

4. Compute the maximum value α for the traffic split ratio for intermediate node $\bar{k}$ that is consistent with the total link utilization constraints for each link set S as

$$\alpha = \min_{S \in \mathcal{S}} \frac{\nu(S)}{U(S)} \tag{8.16}$$

5. For this value α of the traffic split ratio for intermediate node $\bar{k}$, send αR_i amount of traffic from node i to node $\bar{k}$ along path P_i for all $i \neq \bar{k}$ and αC_j amount of traffic from node $\bar{k}$ to node j along path Q_j for all $j \neq \bar{k}$. For this traffic, the total link utilization for link set S is $\alpha U(S)$.

6. Update the weights $w(S)$ as

$$w(S) \leftarrow w(S)\left(1 + \frac{\epsilon \alpha U(S)}{\nu(S)}\right) \quad \forall\, S \in \mathcal{S}$$

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

When the above procedure terminates, total link utilization constraints (8.6) in the primal program will be violated, since we were working with the original $\nu(S)$ values at each stage. To remedy this, we scale down the flow and traffic split ratios α_i uniformly so that these constraints are obeyed.

The pseudo-code for the above procedure, called Algorithm WMN, is provided in the box below. Variables $util(S)$ keep track of the total utilization of link set S as the algorithm progresses. The variable D is initialized to 0 and remains less than 1 as long as the dual objective function value is less than 1. After the **while** loop terminates, the maximum factor by which the utilization constraint for each link set S gets violated is computed into $scale\_fact$. Finally, the α_i values are divided by the maximum utilization violation factor and the resulting values are output.

Algorithm WMN:

$\alpha_k \leftarrow 0 \quad \forall\, k \in N$;

$w(S) \leftarrow \frac{\delta}{\nu(S)} \quad \forall\, S \in \mathcal{S}$;

$util(S) \leftarrow 0 \quad \forall\, S \in \mathcal{S}$;

$D \leftarrow 0$;

while $D < 1$ **do**

For each $i, j \in N$, compute $SP(i,j) = \min_{P \in \mathcal{P}_{ij}} \sum_{e \in P} \frac{1}{u_e} \sum_{S \ni e} w(S)$ using shortest path computation with link costs $c(e) = \frac{1}{u_e} \sum_{S \ni e} w(S)$ for all $e \in E$;

$V(k) \leftarrow \sum_{i \in N, i \neq k} R_i SP(i,k) + \sum_{j \in N, j \neq k} C_j SP(k,j)$;

$\bar{k} \leftarrow \arg\min_{k \in N} V(k)$;

(Denote the shortest path from i to $\bar{k}$ by P_i for all $i \neq \bar{k}$ and the shortest path from $\bar{k}$ to j by Q_j for all $j \neq \bar{k}$.)

$\Delta(e) = \sum_{i \neq \bar{k}, P_i \ni e} R_i + \sum_{j \neq \bar{k}, Q_j \ni e} C_j \quad \forall\, e \in E$;

$U(S) = \sum_{e \in S} \frac{\Delta(e)}{u_e} \quad \forall\, S \in \mathcal{S}$;

$\alpha = \min_{S \in \mathcal{S}} \frac{\nu(S)}{U(S)}$;

$util(S) \leftarrow util(S) + \alpha U(S)$ for all $S \in \mathcal{S}$:

$w(S) \leftarrow w(S)(1 + \frac{\epsilon \alpha U(S)}{\nu(S)})$ for all $S \in \mathcal{S}$;

$\alpha_{\bar{k}} \leftarrow \alpha_{\bar{k}} + \alpha$;

$D \leftarrow \sum_{S \in \mathcal{S}} \nu(S) w(S)$;

end while

$scale\_fact \leftarrow \max_{S \in \mathcal{S}} \frac{util(S)}{\nu(S)}$;

$\alpha_k \leftarrow \frac{\alpha_k}{scale\_fact}$ for all $k \in N$;

Output traffic split ratios α_k ;

We next analyze the approximation guarantee and running time of Algorithm WMN.

Analysis of Approximation Guarantee

The analysis follows the same approach as that of the strongly polynomial time algorithm for maximum throughput two-phase routing in wired networks in Section 4.3.5.

Given a set of dual weights $w(S)$, let $D(w)$ denote the dual objective function value and let $\Gamma(w)$ denote the minimum value of the LHS of dual program constraint (8.12) over all nodes $k \in N$. Then, solving the dual program is equivalent to finding a set of weights $w(S)$ such that $\frac{D(w)}{\Gamma(w)}$ is minimized. Denote the optimal objective function value of the latter by θ, i.e., $\theta = \min_w \frac{D(w)}{\Gamma(w)}$. Let w_{t-1} denote the weight function at the beginning of iteration t of the **while** loop, and let A_{t-1} be the value of $\sum_{j \in N} \alpha_j$ (primal objective function) up to the end of iteration $t-1$. Suppose the algorithm terminates after iteration L. The following lemma upper bounds the value of $D(w)$ at the end of every iteration.

Lemma 8.5.1 *At the end of every iteration t, $1 \leq t \leq L$, of Algorithm WMN, the following holds*

$$D(w_t) \leq |\mathcal{S}| \delta \prod_{j=1}^{t} [1 + \frac{\epsilon}{\theta}(A_j - A_{j-1})]$$

Proof: Let $k = \bar{k}$ be the node for which $V(k)$ is minimum and let P_i, Q_j be the corresponding paths (as defined earlier) along which traffic is sent during iteration t. Recall that the weights are updated as

$$w_t(S) = w_{t-1}(S)\left(1 + \frac{\epsilon\alpha U(S)}{\nu(S)}\right) \quad \forall\, S \in \mathcal{S}$$

where α is the amount by which the traffic split ratio for intermediate node $\bar{k}$ is incremented and $U(S)$ is the total link utilization of link set S per unit split ratio of intermediate node $\bar{k}$ during iteration t. Using this, we have

$$\begin{aligned} D(w_t) &= \sum_{S\in\mathcal{S}} \nu(S) w_t(S) \\ &= \sum_{S\in\mathcal{S}} \nu(S) w_{t-1}(S) + \epsilon\alpha \sum_{S\in\mathcal{S}} w_{t-1}(S) U(S) \\ &= D(w_{t-1}) + \epsilon\alpha \sum_{S\in\mathcal{S}} w_{t-1}(S) \sum_{e\in S} \frac{1}{u_e} [\sum_{i\neq\bar{k}, P_i \ni e} R_i + \sum_{j\neq\bar{k}, Q_j \ni e} C_j] \end{aligned}$$

Interchanging the summations on the RHS of the above equation and first summing over links along paths P_i, Q_j, and then over i, j respectively, we can rewrite the RHS of the above equation to obtain

$$\begin{aligned} D(w_t) &= D(w_{t-1}) + \epsilon\alpha [\sum_{i\neq\bar{k}} R_i \sum_{e\in P_i} \frac{1}{u_e} \sum_{S\ni e} w_{t-1}(S) + \\ &\quad \sum_{j\neq\bar{k}} C_j \sum_{e\in Q_j} \frac{1}{u_e} \sum_{S\ni e} w_{t-1}(S)] \\ &= D(w_{t-1}) + \epsilon\alpha [\sum_{i\neq\bar{k}} R_i \min_{P\in\mathcal{P}_{i\bar{k}}} \sum_{e\in P} \frac{1}{u_e} \sum_{S\ni e} w_{t-1}(S) + \\ &\quad \sum_{j\neq\bar{k}} C_j \min_{P\in\mathcal{P}_{\bar{k}j}} \sum_{e\in P} \frac{1}{u_e} \sum_{S\ni e} w_{t-1}(S)] \qquad (8.17) \\ &= D(w_{t-1}) + \epsilon\alpha\Gamma(w_{t-1}) \\ &= D(w_{t-1}) + \epsilon(A_t - A_{t-1})\Gamma(w_{t-1}) \end{aligned}$$

The step leading to (8.17) is due to the fact that the paths P_i, Q_j are shortest paths between respective nodes under link costs $c(e) = \frac{1}{u_e} \sum_{S \ni e} w(S)$. The next step follows from the choice of intermediate node $k = \bar{k}$ to minimize $V(k)$ for the weights $w_{t-1}(e)$ at the beginning of iteration t.

Using this for each iteration down to the first one, we have

$$D(w_t) = D(w_0) + \epsilon \sum_{j=1}^{t} (A_j - A_{j-1}) \Gamma(w_{j-1}) \tag{8.18}$$

From the definition of θ, we have $\theta \leq \frac{D(w_{j-1})}{\Gamma(w_{j-1})}$, whence $\Gamma(w_{j-1}) \leq \frac{1}{\theta} D(w_{j-1})$. Also, $D(w_0) = |\mathcal{S}|\delta$. Using these in equation (8.18), we have

$$D(w_t) \leq |\mathcal{S}|\delta + \frac{\epsilon}{\theta} \sum_{j=1}^{t} (A_j - A_{j-1}) D(w_{j-1}) \tag{8.19}$$

The property claimed in the lemma can now be proved using inequality (8.19) and mathematical induction on the iteration number t. The method is similar to that used in the proof of Lemma 4.3.1. ■

We now estimate the factor by which the objective function value value A_L in the primal solution needs to be scaled when the algorithm terminates so as to ensure that link capacity constraints are not violated.

Lemma 8.5.2 *When Algorithm WMN terminates, the primal solution needs to be scaled by a factor of at most* $\log_{1+\epsilon} \frac{1+\epsilon}{\delta}$ *to ensure primal feasibility.*

Proof: We need to show that for every link set S, the total utilization of all its links is at most $\nu(S)$ when the primal solution is scaled by the above amount.

Consider any link set S and associated weight $w(S)$. Recall that in every iteration, the weight $w(S)$ is updated as

$$w(S) \leftarrow w(S) \left(1 + \frac{\epsilon \alpha U(S)}{\nu(S)}\right)$$

The quantity $\alpha U(S)$ is equal to the total utilization of all links in S corresponding to the flow sent during iteration t. Let the sequence of such total utilization values values for set S over all iterations be $U_1, U_2, \ldots, U_L$. Let $\sum_{t=1}^{L} U_t = \kappa\nu(S)$, i.e., the total utilization over all iterations for set S exceeds its upper bound $\nu(S)$ by a factor of κ.

Because of the way in which α is chosen in accordance with equations (8.14)-(8.16), we have $U_i \leq \nu(S)$ for all i. Hence, dual weights $w(S)$ are updated by a factor of at most $1+\epsilon$ after each iteration. Since the algorithm terminates when $D(w) \geq 1$, and since dual weights are updated by a factor of at most $1+\epsilon$ after each iteration, we have $D(w_L) < 1+\epsilon$. Since the weight $w(S)$, with coefficient $\nu(S)$, is one of the summing components of $D(w)$, we have $\nu(S)w_L(S) < 1+\epsilon$. Also, the value of $w_L(S)$ is given by

$$w_L(S) = \frac{\delta}{\nu(S)} \prod_{t=1}^{L} (1 + \frac{U_t}{\nu(S)}\epsilon)$$

Using the inequality $(1+cx) \geq (1+x)^c$ for all $x \geq 0$ and any $0 \leq c \leq 1$ and setting $x = \epsilon$ and $c = \frac{U_t}{\nu(S)} \leq 1$, we have

$$\begin{aligned} \frac{1+\epsilon}{\nu(S)} > w_L(S) &\geq \frac{\delta}{\nu(S)} \prod_{t=1}^{L} (1+\epsilon)^{U_t/\nu(S)} \\ &= \frac{\delta}{\nu(S)} (1+\epsilon)^{\sum_{t=1}^{L} U_t/\nu(S)} \\ &= \frac{\delta}{\nu(S)} (1+\epsilon)^{\kappa} \end{aligned}$$

whence,

$$\kappa < \log_{1+\epsilon} \frac{1+\epsilon}{\delta}$$

■

The values of ϵ and δ are related, in the following theorem, to the approximation factor guarantee of Algorithm WMN.

Theorem 8.5.3 *For any given $0 < \epsilon' \leq 0.5$, Algorithm WMN computes a solution*

with objective function value within $(1+\epsilon')$*-factor of the optimum for*

$$\delta = \frac{1+\epsilon}{[(1+\epsilon)|\mathcal{S}|]^{1/\epsilon}} \quad \textit{and} \quad \epsilon = \frac{1}{2}\epsilon'$$

Proof: Using Lemma 8.5.1 and the inequality $1 + x \leq e^x$ for all $x \geq 0$, we have

$$\begin{aligned} D(w_t) &\leq |\mathcal{S}|\delta \prod_{j=1}^{t} e^{\frac{\epsilon}{\theta}(A_j - A_{j-1})} \\ &< |\mathcal{S}|\delta e^{\epsilon A_t/\theta} \end{aligned}$$

The simplification in the above step uses telescopic cancellation of the sum $(A_j - A_{j-1})$ over j. Since the algorithm terminates after iteration L, we must have $D(w_L) \geq 1$. Thus,

$$1 \leq D(w_L) \leq |\mathcal{S}|\delta e^{\epsilon A_L/\theta}$$

whence,

$$\frac{\theta}{A_L} \leq \frac{\epsilon}{\ln \frac{1}{|\mathcal{S}|\delta}} \tag{8.20}$$

From Lemma 8.5.2, the objective function value of the feasible primal solution after scaling is at least

$$\frac{A_L}{\log_{1+\epsilon} \frac{1+\epsilon}{\delta}}$$

The approximation factor for the primal solution is at most the gap (ratio) between the dual and primal solutions. Using the lower bound on the primal solution and inequality (8.20), this is at most

$$\begin{aligned} \frac{\theta}{\frac{A_L}{\log_{1+\epsilon} \frac{1+\epsilon}{\delta}}} &\leq \frac{\epsilon \log_{1+\epsilon} \frac{1+\epsilon}{\delta}}{\ln \frac{1}{|\mathcal{S}|\delta}} \\ &= \frac{\epsilon}{\ln(1+\epsilon)} \frac{\ln \frac{1+\epsilon}{\delta}}{\ln \frac{1}{|\mathcal{S}|\delta}} \end{aligned}$$

The quantity $\ln \frac{1+\epsilon}{\delta} / \ln \frac{1}{|\mathcal{S}|\delta}$ equals $\frac{1}{1-\epsilon}$ for $\delta = (1+\epsilon)/[(1+\epsilon)|\mathcal{S}|]^{1/\epsilon}$. Using this value of δ, the approximation factor is upper bounded by $\frac{\epsilon}{(1-\epsilon)\ln(1+\epsilon)}$. This quantity is at most $1+2\epsilon$ for $\epsilon \leq 0.25$. Setting $\epsilon = \frac{\epsilon'}{2}$, we get the desired approximation ratio of $1+\epsilon'$. ■

Analysis of Running Time

We now analyze the running time of Algorithm WMN.

Theorem 8.5.4 *For any given $\epsilon > 0$ chosen to provide the desired approximation factor guarantee in accordance with Theorem 8.5.3, Algorithm WMN runs in time*

$$O\left(\frac{1}{\epsilon^2}(n(m+n\log n)+\sum_{S\in\mathcal{S}}|S|)|\mathcal{S}|\log|\mathcal{S}|\right)$$

Proof: We first consider the running time of each iteration of the algorithm during which a node $\bar{k}$ and associated paths P_i, Q_j are chosen to augment flow. Selection of this node and the paths involves an all-pairs shortest path computation. The computation of the links costs $c(e) = \frac{1}{u_e}\sum_{S\ni e} w(S)$ for all $e \in E$ can be implemented in $O(\sum_{S\in\mathcal{S}}|S|)$ time by going through the sets once and updating the link costs $c(e)$. The all-pairs shortest path computation itself can be implemented in $O(nm+n^2\log n)$ time using Dijkstra's shortest path algorithm with Fibonacci heaps [AMO93]. All other operations within an iteration, including computation of the value of α and updating weights $w(S)$, are absorbed by the time taken for the above two computations, leading to a total of $O(n(m+n\log n)+\sum_{S\in\mathcal{S}}|S|)$ time per iteration.

We next estimate the number of iterations before the algorithm terminates. Recall that in each iteration, traffic is sent along paths P_i, Q_j corresponding to the maximum value of intermediate node split ratio α such that the total link utilization $\alpha U(S)$ for set S during that iteration is at most $\nu(S)$. Thus, for at least one link set S, $\alpha U(S) = \nu(S)$ and the weight $w(S)$ increases by a factor of $1+\epsilon$. Accordingly, with each iteration, we can associate a weight $w(S)$ which increases by a factor of $1+\epsilon$.

Consider the weight $w(S)$ for fixed $S \in \mathcal{S}$. Since $w_0(S) = \frac{\delta}{\nu(S)}$ and $w_L(S) < \frac{1+\epsilon}{\nu(S)}$ (as deduced in the proof of Lemma 8.5.2), the maximum number of times that this weight can be associated with any iteration is

$$\log_{1+\epsilon} \frac{1+\epsilon}{\delta} = \frac{1}{\epsilon}(1 + \log_{1+\epsilon} |\mathcal{S}|) = O(\frac{1}{\epsilon} \log_{1+\epsilon} |\mathcal{S}|)$$

Since there are a total of $|\mathcal{S}|$ weights $w(S)$, hence the total number of iterations is upper bounded by $O(\frac{1}{\epsilon}|\mathcal{S}| \log_{1+\epsilon} |\mathcal{S}|)$. Multiplying this by the running time per iteration, we obtain the overall algorithm running time as $O(\frac{1}{\epsilon}(n(m + n \log n) + \sum_{S \in \mathcal{S}} |S|)|\mathcal{S}| \log_{1+\epsilon} |\mathcal{S}|)$. Since $\ln(1+\epsilon) = \Theta(\epsilon)$, this is $O(\frac{1}{\epsilon^2}(n(m+n \log n) + \sum_{S \in \mathcal{S}} |S|)|\mathcal{S}| \log |\mathcal{S}|)$. ■

For the directional antenna case, we have $\mathcal{S} = \{N(i) \mid i \in N\}$ so that $|\mathcal{S}| = n$. Also, $\sum_{S \in \mathcal{S}} |S| = \sum_{i \in N} |N(i)| = 2m$. Hence, the running time of Algorithm WMN for the directional antenna case is $O(\frac{1}{\epsilon^2} n^2 (m + n \log n) \log n)$.

8.6 Handling Link Interference

In this section, we discuss how to accommodate link interference for omnidirectional antenna transmission in our optimization framework. We use the *protocol model of interference* introduced in Gupta and Kumar [GK00] which we describe shortly.

The problem of computing the optimal throughput under the protocol model of interference, even for fixed traffic and a single source-destination pair, is shown to be $\mathcal{NP}$-hard in Jain et al. [JPPQ03]. The reduction in the hardness proof is from the problem of finding the independence number of a graph and is an approximation preserving one. Hence, this rules out the existence of Polynomial Time Approximation Schemes (PTAS) for this problem. A PTAS is a family of algorithms that, for any given $\epsilon > 0$, computes a solution that is within $(1 + \epsilon)$-factor of the optimum and has running time that is polynomial in the problem size (ϵ is treated as a constant in the expression for running time and hence the dependence of running time on it could be

arbitrary).

Given this result, it follows that it is $\mathcal{NP}$-hard to even approximate the optimal throughput for two-phase routing in WMNs under the protocol model of interference. We consider how to obtain near-optimal solutions by adding either of two types of constraints – clique constraints and independent set constraints – to our linear program for the directional antenna case. These constraints have been used in [JPPQ03] to model link interference within a linear programming based optimization framework when the traffic matrix is fixed. Either of these approaches can be used to obtain near-optimal solutions with sufficient computational effort. In practice, one has to settle for a trade-off between the closeness to optimality of the solution and the computational resources required to achieve it.

8.6.1 Model of Interference

Let d_{ij} denote the distance between nodes i and j. Let the radio at node i have a communication range of ℓ_i and potentially larger interference range ℓ'_i. Under the *protocol model of interference* [GK00], if there is a single wireless channel, a transmission from node i to node j is successful if

(i) $d_{ij} \leq \ell_i$ (receiver is within communication range of sender), and

(ii) any node k, such that $d_{kj} \leq \ell'_k$, is not transmitting (receiver is free of interference from any other possible sender).

We have already modeled requirement (i) through links in the topology graph. We will model requirement (ii) using the conflict graph as described in the next section. Note that (ii) includes as a special case the constraint that a node may not send and receive at the same time nor transmit to more than one other node at the same time – this aspect was modeled in constraint (8.2) of the linear programming formulation in Section 8.5.1 for the directional antenna case.

The definition of a successful transmission under the protocol model of interference can be made more restrictive in order to accommodate the MAC protocol in the IEEE 802.11 wireless networking standard [G02] that performs virtual sensing through exchange of RTS/CTS (Request to Send/Clear to Send) messages and requires the receiver to send an acknowledgement after successful transmission. In this case, the sending node i should also be free from interference from nodes other than the receiving node j, since the receiver will be sending acknowledgements or a CTS message that should be received successfully by the sender. This is modeled by adding a third constraint for a successful transmission from node i to node j:

(iii) any node k, such that $d_{ki} \leq \ell'_k$, is not transmitting (sender is also free of interference from any other possible sender).

For the rest of this chapter, we assume the first (less restrictive) model of interference where there are no RTS/CTS or acknowledgement messages. Adding the additional constraint to model the latter merely introduces additional edges in the conflict graph – the overall method works for any given conflict graph and hence does not change.

8.6.2 Conflict Graph

Following the approach in [JPPQ03], we construct the *conflict graph* to model the phenomenon of link interference under the protocol model of interference. Denote this graph by $F = (N_c, A_c)$. The vertices of F correspond to links in the topology graph G, i.e., $N_c = A$. There is an (undirected) edge between two nodes in the conflict graph corresponding to topology links (i, j) and (i', j') if $d_{ij'} \leq \ell'_i$, or $d_{i'j} \leq \ell'_{i'}$. This corresponds to the fact that the two links cannot be active simultaneously, i.e., they interfere and violate requirement (ii) in the protocol model of interference described above. In Figure 8-4, the conflict graph corresponding to a topology graph is shown. (The dotted arcs in the topology graph join interfering link pairs.)

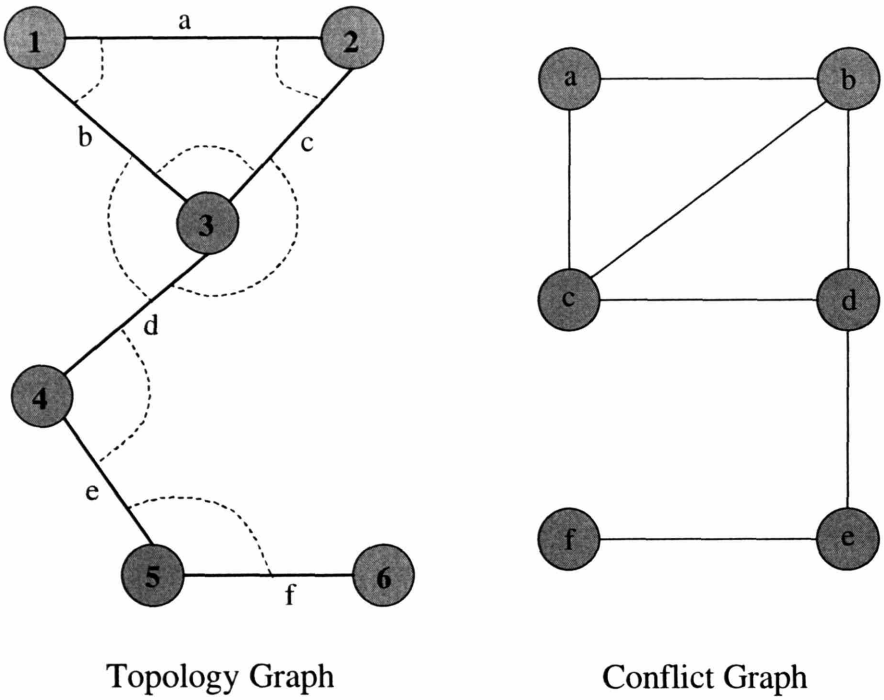


Figure 8-4: A WMN topology graph and its conflict graph. Dotted arcs in the topology graph join interfering link pairs.

8.6.3 Clique Constraints

Consider a *clique* in the conflict graph. Let S be the set of topology links that correspond to nodes of this clique in the conflict graph. Since the links in S mutually conflict with each other, their total utilization cannot exceed 1. This can be modeled by the constraint

$$\sum_{e \in S} \frac{1}{u_e} \sum_{i,j \in N} x_e^{ij} \leq 1 \tag{8.21}$$

We can add constraints of this type for each clique in the conflict graph. Moreover, it is sufficient to add just constraints corresponding to *maximal cliques*, since the constraints corresponding to cliques contained inside maximal cliques are redundant. Since these clique constraints have the same structure as constraint (8.2), they can

be easily incorporated into the combinatorial algorithm of Section 8.5.4.

The number of such maximal clique constraints is exponential in the worst case. The running time of the linear program will not be feasible for practical implementations because of the large number of clique constraints. Herein lies the usefulness of the combinatorial algorithm – it can be expected to run faster in practice because of two aspects, namely (i) it uses simple iterative shortest path computations, and (ii) for a WMN deployment whose geographical span area is large compared to the interference range, the number of maximal cliques in the conflict graph is not likely to be very large, since link interference is a localized phenomenon.

Unfortunately, the clique constraints provide only necessary conditions for the link rates to have a realizable schedule – these conditions may not be sufficient. Thus, the throughput obtained by adding all maximal clique constraints may be a strict upper bound on the maximum achievable throughput. As observed in [JPPQ03], utilization constraints for other types of subgraphs in the conflict graph may need to be added, e.g., an odd length cycle with no chords, called an *odd hole*. The sum of the utilizations of the links corresponding to the vertices of an odd hole of length a in the conflict graph can be at most $\lfloor \frac{a}{2} \rfloor$. Constraints corresponding to other types of subgraph in the conflict graph may also need to be added.

The throughput upper bound based only on maximal clique constraints is tight only for a special class of conflict graphs called *perfect graphs* [Lo72]. Perfect graphs are characterized by the following property: for all induced subgraphs, the chromatic number is equal to the clique number (size of largest clique).

8.6.4 Independent Set Constraints

Consider an independent set of vertices in the conflict graph. These correspond to links in the topology graph that can be active simultaneously. An independent set is said to be *active* at any given time if some subset of the corresponding links are active. Let $I_1, I_2, \ldots, I_p$ denote all the *maximal* independent sets in the conflict graph

(let each set consist of corresponding links in the topology graph). Let independent set I_j be active for y_j fraction of the time. Any set of actively transmitting links are contained in some independent set (in case there are multiple such independent sets, we can choose one arbitrarily). Then, we should have

$$\sum_{j=1}^{p} y_j \leq 1 \tag{8.22}$$

For an individual link, its utilization is at most the sum of the fraction of time that each independent set it belongs to is active. This can be written as

$$\frac{1}{u_e} \sum_{i,j \in N}^{p} x_e^{ij} \leq \sum_{j:I_j \ni e} y_j \tag{8.23}$$

It is east to see that constraints (8.22)-(8.22) provide a set of necessary *and* sufficient conditions for link rates to be schedulable in the protocol model of interference. Hence, these constraints, together with constraint (8.1) of the linear program in Section 8.5.1, give an exact formulation for maximizing throughput for two-phase routing in WMNs under the protocol model of interference.

Since the number of maximal independent sets in the conflict graph can be exponential in the worst case, this does not lead to a polynomial time algorithm. Moreover, the number of maximal independent sets in the conflict graph could be larger than the number of maximal cliques, since links do not interfere if they are not geographically proximal.

In practice, one can choose a small number of maximal independent sets in the conflict graph (dictated by the available computational resources) and add constraints corresponding to them in the linear program. (The set of maximal independent sets chosen should be such that their union contains all nodes in the conflict graph.) The obtained solution would produce link rates that are schedulable and hence give lower bounds on the optimal throughput. The lower bound can be improved by considering more maximal independent sets – the obtained solutions would ultimate converge

to the optimal throughput. We investigate this behavior on a medium sized WMN topology in Section 8.9.

8.7 Link Transmission Scheduling

Given the maximum achievable link rates u_e, ingress-egress bounds R_i, C_j on end-user traffic, and the node commitment vector $\vec{\eta}$, we use the algorithms in Sections 8.5 and 8.6 to obtain the traffic split ratios and routing of Phase 1 and Phase 2 paths for two-phase routing. Our optimization framework explicitly models constraints imposed on scheduling as a result of link interference. It now remains to compute the static link transmission schedule that will realize the Phase 1 and Phase paths obtained from the routing algorithm.

The routing of the Phase 1 and Phase 2 paths of fixed bandwidth give us data rates $x_e = \sum_{i,j \in N} x_e^{ij}$ that need to be achieved on each link e of the network. These data rates can be translated into a *periodic* link transmission schedule that obeys interference constraints. Suppose that the schedule has a period of M slots. If link e transmits during M_e of these slots, then we must have $\frac{M_e}{M} \geq \frac{x_e}{u_e}$ for the data rates x_e to be achievable, whence $M_e = \lceil \frac{M x_e}{u_e} \rceil$.

The schedule can be constructed using an edge coloring algorithm as follows. We begin with a sufficiently large initial guess for M. Since time slot lengths are small (order of milliseconds or smaller), we can choose M to be the number of time slots per second. We make M_e copies of link e in the topology graph. The scheduling problem is now equivalent to edge coloring this multi-graph with at most M colors subject to the following constraints:

- The M_e copies of link e must be assigned distinct colors, and
- If two links (i.e., their copies) interfere, then they must be assigned different colors.

Each color in the edge coloring problem represents a time slot. An edge is assumed to be active in the time slot which corresponds to its assigned color. Thus, all links that have the same color transmit simultaneously in that time slot (these links do not interfere with each other because of the second coloring constraint).

If the coloring requires $M' > M$ colors, then the computed schedule has a reduced throughput equal to at least $\frac{M}{M'}(< 1)$ times that obtained by the optimization methods for maximum throughput routing. One can try larger values of M in an attempt to increase the value of the ratio $\frac{M}{M'}$. Experiments show that simple greedy edge coloring algorithms work very well in practice and that for large values of M, the number of required colors M' is only a few more than M, i.e., the fraction $\frac{M}{M'}$ is of the order of $\frac{1}{M}$.

The coloring problem to determine the link transmission schedule is $\mathcal{NP}$-hard even for the directional antenna case – in this case, the problem reduces to standard edge coloring where two edges incident on the same node must be assigned different colors [GJ79]. However, it is important to note that *this hard scheduling problem can be solved in a centralized manner and the static schedule can be pre-computed and uploaded to the WMN nodes.*

8.8 Generalization to Multiple Channels and Multiple Radios

Our optimization framework for two-phase routing in WMNs can be extended to handle multiple wireless channels and multiple radios at each node. We discuss these extensions in this section.

For multiple wireless channels, the radio at each node can tune to any one of $W \geq 1$ available channels at a given time. We can model this by replacing each link (i, j) in the topology graph by W parallel links from node i to node j. Each such link corresponds to the use of a specific channel for communication from node i to node

j. In the conflict graph, links corresponding to different channels do not interfere. Since a single radio can tune to exactly one channel at a time, the links that have an end-node in common will mutually conflict with each other as before.

Now consider R multiple radios per node where each radio can transmit on a fixed (but different) channel. We can model this replacing each link (i, j) in the topology graph by R parallel links from node i to node j. Each such link corresponds to the use of the radio for that channel for communication from node i to node j. As before, links corresponding to different channels do not interfere. Since the channel for a given radio is fixed, the links that have an end-node in common *and* correspond to the same channel will mutually conflict with each other.

The above modifications naturally extend to the procedure for link transmission scheduling discussed in Section 8.7.

8.9 Performance Evaluation

In this section, we evaluate the performance of two-phase routing in WMNs. In a manner analogous to that for the wired network case in Section 4.6.1, we will use *throughput efficiency* to measure the effectiveness of two-phase routing in WMNs with respect to the optimal scheme. We define this first.

Throughout this section, we will consider the throughput obtained by solving the routing problem (which also models the link transmission scheduling constraints). We will, however, not be concerned with the actual scheduling of the link transmissions and hence ignore any decrease in the achievable throughput resulting from this. In practice, this decrease is small, given the small duration (order of milliseconds or smaller) of the transmission time slots (see note in Section 8.7).

8.9.1 Throughput Efficiency

Given a WMN network with maximum achievable link rates u_e, bounds R_i, C_i on the ingress-egress traffic, and the node commitment vector $\vec{\eta}$, an output λ^* of the problem formulation for two-phase routing in Sections 8.5 and 8.6 provides a guarantee that all matrices in $\lambda^* \cdot \mathcal{T}(\vec{R}, \vec{C})$ can be routed by two-phase routing. The optimal routing scheme in the context of WMNs is defined in a manner similar to that for the wired network case. That is, the optimal scheme has the flexibility of changing the routing with changes in the traffic matrix. For routing any matrix, it must also enforce additional constraints peculiar to WMNs, namely (i) mutually exclusive transmit/receive nature of wireless communication at a node, and (ii) link interference for omnidirectional antenna case. The optimal scheme admits the highest possible throughput λ_{OPT}. We use the ratio $\frac{\lambda^*}{\lambda_{OPT}}$ to define the *throughput efficiency* of two-phase routing in WMNs.

> **Definition (Throughput Efficiency):** Under given maximum achievable link rates and ingress-egress bounds on the traffic matrix in an WMN, the *throughput efficiency* of two-phase routing is given by the quantity $\frac{\lambda^*}{\lambda_{OPT}}$ (≤ 1).

Similar to that for the wired network case, we define the throughput $\lambda(T)$ for any traffic matrix T to be the maximum multiplier such that $\lambda(T) \cdot T$ can be feasibly routed subject to the additional constraints for (i) and (ii) above. For a given matrix T, the quantity $\lambda(T)$ can be computed by solving the maximum concurrent flow problem [SM90] with additional constraints corresponding to (i) and (ii) above.

With the above definitions for λ_{OPT} and $\lambda(T)$ that are similar to that for the wired network case, it should be easy to see that Lemma 4.5.1 and Theorem 4.5.2 continue to hold in the context of WMNs (the proofs are essentially the same). It follows that the throughput efficiency of two-phase routing in WMNs is at least 0.5 (or, 50%) when the ingress-egress capacities are symmetric, i.e., $R_i = C_i$ for all i. The

latter assumption holds for all the WMN topologies we use in our experiments. We will see that the throughput efficiency of two-phase routing on the evaluated WMN topologies is significantly better than the theoretical lower bound of 50%.

We use the methods discussed in Section 4.5.3 to compute a "good" upper bound $\hat{\lambda}_{OPT}$ for λ_{OPT}. The general approach in that section is to identify a matrix $T \in \mathcal{T}(\vec{R}, \vec{C})$ such that $\lambda(T)$ gives a good upper bound $\hat{\lambda}_{OPT}$. The main difference is that the computation of $\lambda(T)$ for a given T involves additional constraints for WMNs as explained above. For the case of equal ingress-ingress capacities, the argument in the proof of Lemma 4.5.3 can be extended for WMNs, since the additional constraints are linear. Thus, we can use traffic matrices corresponding to derangement permutations to upper bound λ_{OPT} in the case of WMNs also.

Since $\lambda_{OPT} \leq \hat{\lambda}_{OPT}$, we have

$$\frac{\lambda^*}{\hat{\lambda}_{OPT}} \leq \frac{\lambda^*}{\lambda_{OPT}} \leq 1$$

Thus, the quantity $\frac{\lambda^*}{\hat{\lambda}_{OPT}}$ is a lower bound on the throughput efficiency of two-phase routing in WMNs. We will use this lower bound as a conservative estimate of throughput efficiency of two-phase routing in all the experiments.

8.9.2 Topologies and Link Rates

The operating range of the model corresponds roughly to the ETSI standards for UMTS [UMTS98]. The main assumptions for our experiments are as follows:

- The system is operating in the $2.4000 - 2.4835$ GHz ISM band.
- All transmitters share the full $W = 83.5$ MHz band.
- Different power levels starting from about 100 mW to 200 mW are considered. The transmitter has limited power in real systems so as to reduce the link interference range.

- A deterministic fading model is used for link capacities. For link (i, j) with transmitter node i and receiver node j, the maximum achievable link rate is given by

$$u_{ij} = W \log_2 \left(1 + \frac{PG_tG_rL}{d^\alpha \sigma}\right)$$

where P is the transmit power of node i, $W = 83.5$ MHz is the frequency band, G_t and G_r are the transmit and receiver antenna gains respectively, and $L = 2 \times 10^{-4}$ is the path loss at unit distance (1 m) from the transmitter. We assume unit antenna gains, i.e., $G_r = G_t = 1$. The distance between the transmitter and the receiver is d, and the path loss exponent $\alpha = 3$. The background noise can be approximated by $\sigma = fkTW$, where f is the receiver noise figure, k is the Boltzmann's constant, and T is the absolute temperature of the receiver circuitry. Assuming a receiver noise figure of 10 and a temperature of $T = 290K$, we get $\sigma = 3.34 \times 10^{-12}$ as the background noise.

We generated random (connected) topologies by picking points at random on a $a \times a$ grid where $a = 500$ m. The communication range was assumed to be 100 m at a node transmission power of $P = 100$ mW. We used the corresponding link rate as the cutoff point for adding new links to the WMN topology when transmission power was increased. For the first four sets of experiments, we assume a narrow beam forming antenna so that there is no link interference. In the fifth set of experiments, we investigate the impact of link interference for omnidirectional antenna on throughput and the convergence of the throughput bounds obtained by the clique and independent set constraints. For this set of experiments, we assume that the interference range is *twice* the communication range, that is, 200 m at $P = 100$ mW. Using the cutoff link rate as explained earlier, we obtain a communication range of 126 m at $P = 200$ mW, and hence, a corresponding interference range of 252 m.

8.9.3 Experiments and Results

In the absence of any modeling information for varying the ingress-egress data rates across WMN nodes, we assumed the R_i, C_j values to be equal and normalized them to 1 Mbps, i.e., $R_i = C_i = 1$ Mbps for all i. In many envisaged applications for WMNs, the available end-user spectrum at each WMN node is comparable, hence this assumption is not unrealistic. We also fixed the η_i values to unity – this assumes that the spectrum for connectivity between the WMN nodes and the end-user devices (user links) is not shared with the spectrum providing interconnection among the WMN nodes (network links).

In all the experiments, we use the linear programming formulations and solve the problems using CPLEX [CPLEX] so as to obtain exact answers for comparison purposes. Solving the linear programming problems on 50-node WMN topologies for the directional antenna case and for up to an order of thousand clique/independent set constraints for the omnidirectional antenna case turned out to be feasible in CPLEX for the CPU clock speed and available RAM on the machines that we had access to (2.4GHz Dual Xeon, 1GB RAM).

Throughput and Throughput Efficiency

We consider ten randomly generated 50-node WMN topologies in a 500 m $\times$ 500 m square area. Each node transmits with a power of 200 mW. The throughput values are plotted in Figure 8-5 and the throughput efficiency values in Figure 8-6. Note that throughput is also the ingress-egress data rate guarantee for each WMN node since $R_i = C_i = 1$ Mbps for all i. The guaranteed data rate for each node ranges from about 10 Mbps to 18 Mbps for the WMN topologies. The throughput efficiency of two-phase routing ranges from about 0.92 to slightly less than 1.0. This implies that, for these WMN topologies, an optimal algorithm that provides traffic matrix independent bandwidth guarantee with the flexibility of changing the routing with changes in the traffic matrix cannot give more than about 1.09 times the throughput

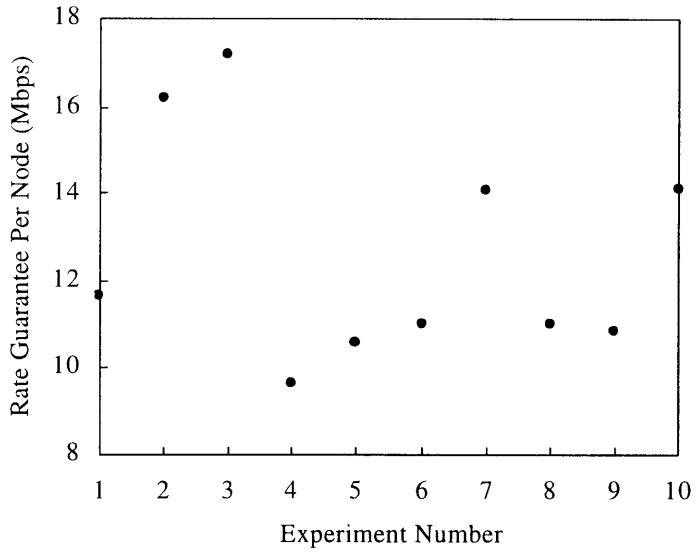


Figure 8-5: Throughput of Two-Phase Routing in WMNs: 50-node topologies, 200 mW Transmission Power.

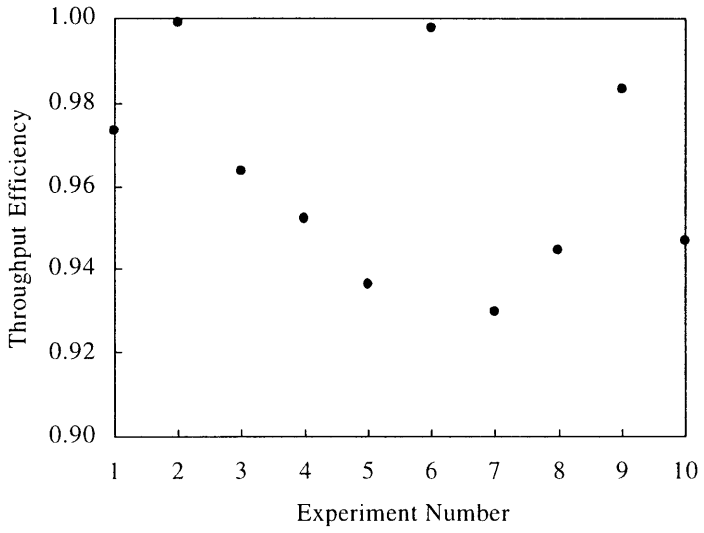


Figure 8-6: Throughput Efficiency of Two-Phase Routing in WMNs: 50-node topologies, 200 mW Transmission Power.

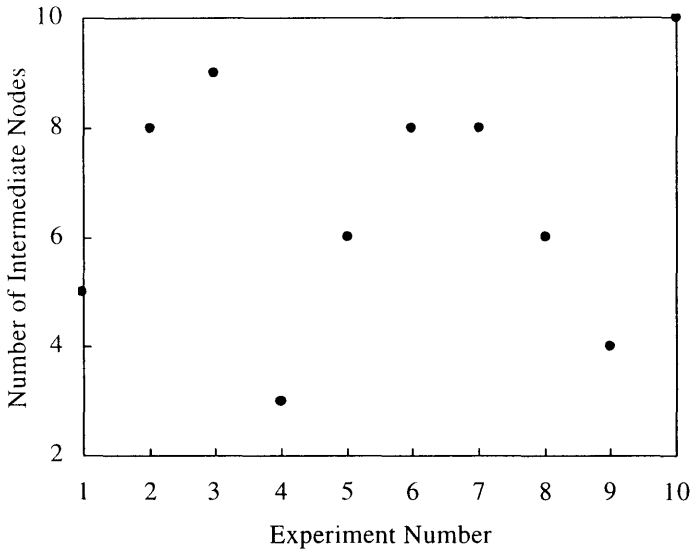


Figure 8-7: Number of Intermediate Nodes in Two-Phase Routing in WMNs: 50-node topologies, 200 mW Transmission Power.

guaranteed by two-phase routing.

Number of Intermediate Nodes

The number of intermediate nodes i with $\alpha_i > 0$ for each topology is shown in Figure 8-7. Interestingly, as was observed for the wired network case, 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 the applications mentioned in Section 8.2, particularly for providing rendezvous based services in WMNs. In the latter application, the intermediate nodes are sites for locating the rendezvous points.

Increasing Transmission Power

In Figure 8-8, we investigate the effect of increasing transmission power on the throughput. For one of the randomly generated 50-node topologies above, we in-

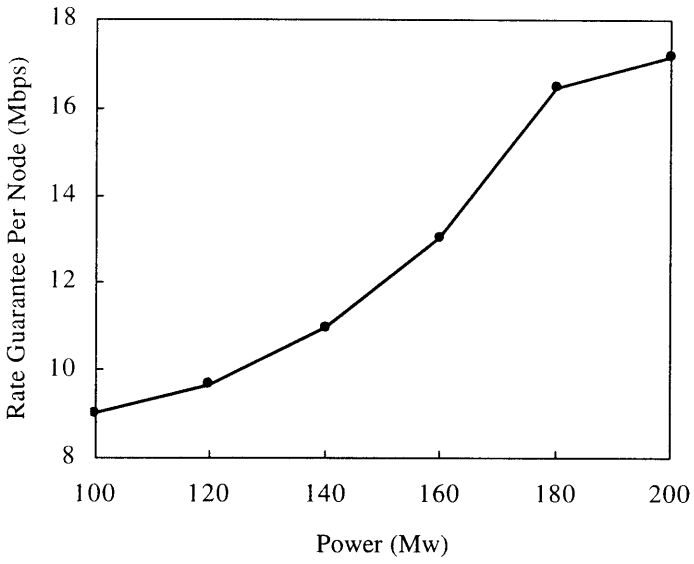


Figure 8-8: Throughput vs. Transmission power for Two-Phase Routing in WMNs: 50-node topology.

creased the transmission power P from 100 mW to 200 mW in steps of 20 mW. As the transmission power is increased, three effects come into play, namely, (i) the achievable rate of each existing link increases, (ii) new links appear in the topology (these had rates below the cutoff point at lower power values), and (iii) the link interference range increases (since it is assumed to be twice the communication range). The first two effects have a favorable effect on the throughput, while the third has an opposite effect. However, because we assumed a narrow barrow beam forming antenna for this set of experiments, the third effect is absent. The throughput increases as a result of the first two effects, as shown in Figure 8-8. The third effect comes into play when we consider link interference in the fifth set of experiments – in that case also, the throughput increases with increase in transmission power.

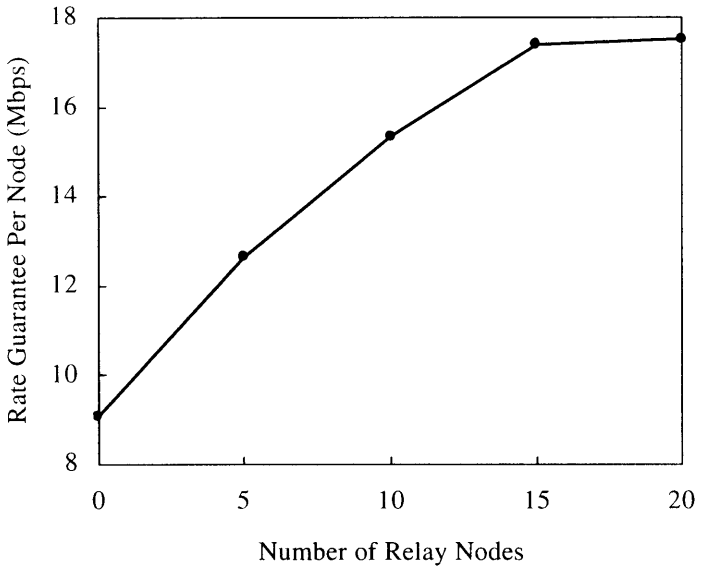


Figure 8-9: Throughput vs. Number of relay nodes for Two-Phase Routing in WMNs: 50-node base topology, Transmission Power 100 mW.

Effect of Relay Nodes

The throughput of the network can also be increased by introducing relay nodes that participate in network routing but do not provide connectivity to end-user devices. This effect is plotted in Figure 8-9 for one of the randomly generated 50-node topologies above, where the number of relay nodes is increased from 0 to 20. The transmission power was fixed at 100 mW. Thus, relay nodes can naturally augment the topology of WMNs and increase throughput as the number of users increases.

Impact of Link Interference for Omnidirectional Antenna

We investigate the impact of link interference for omnidirectional antenna on the throughput of two-phase routing in WMNs. Because the routing problem with the link scheduling constraints is $\mathcal{NP}$-hard in this case, we designed our experiments to show the convergence of the upper and lower bounds on throughput using clique and independent set constraints respectively. The number of cliques and independent

sets in the conflict graph could be exponential in the graph size in the worst case – generating all of them could also require exponential amount of time in the worst case. Hence, we use an iterative heuristic described below to generate these and plot the convergence of the upper and lower bounds as a function of the number of iterations the heuristic is executed.

The heuristic runs iteratively and attempts to generate a new clique (or, independent set) in each iteration in a greedy manner. During each iteration, the nodes in the conflict graph are considered in some random (permutation) order. The clique set (or, independent set) is initialized to consist of the first node in this order. Thereafter, the remaining nodes are considered in this order and each node is added to the set if it forms a clique (or, independent set) with the nodes already in the set. If the resulting clique (or, independent set) has not been generated in earlier iterations, it is added to the collection of cliques (or, independent sets) generated.

This procedure may not produce a new clique (or, independent set) after each iteration. We plot the convergence of the upper and lower bounds as a function of the number of iterations this heuristic is run, i.e., the number of attempts to generate cliques (or, independent sets) in the conflict graph.

The collection of cliques and the collection of independent sets are initialized as follows before running the above iterative heuristic. We initialize the collection of cliques to contain cliques that correspond to transmit/receive diversity constraints for the directional antenna case – each such clique consists of the set of links that are incident at a given WMN node (incoming or outgoing). The total number of such cliques is n. The collection of independent sets must be initialized so that their union contains all the links in order to give a valid lower bound. For this, we consider the links in some arbitrary order. For each link, if it is not already included in some independent set, we form a singleton independent set consisting of this link and grow it in a manner similar to an iteration of the above heuristic. Note that the number of attempts to generate cliques (or, independent sets), as plotted in the experiments

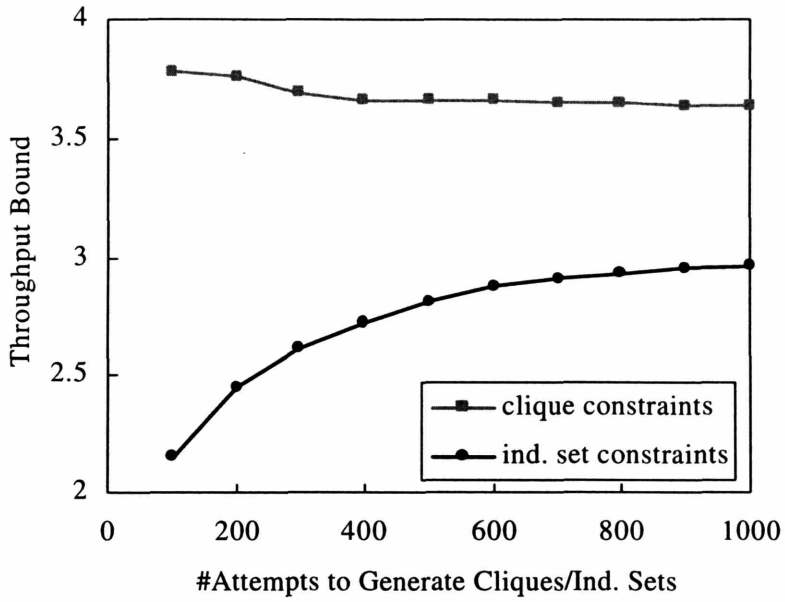


Figure 8-10: Upper and Lower Bounds for Throughput of Two-Phase Routing in WMN: 50-node topology, 100 mW Transmission Power.

below, does not include the described initialization above.

In Figure 8-10, we plot the upper and lower bounds for the throughput of two-phase routing as a function of the number of attempts to generate maximal cliques and independent sets respectively up to 1000 such attempts. The WMN has a randomly generated 50-node topology with a node transmission power $P = 100$ mW. The upper bound flattens quickly to a value of about 3.63 corresponding to 400 attempts to generate maximal cliques in the conflict graph. The lower bound flattens less rapidly – with 1000 attempts to generate maximal independent sets in the conflict graph, the value is about 2.97. We were able to improve the lower bound further to 3.36 using 10000 attempts. The gap between the upper and lower bounds thus obtained is about 8%. There is no way to verify whether the gap will get significantly smaller with more attempts to generate cliques and independent sets other than carrying out the computation. Since the independent set constraints produce a lower bound for

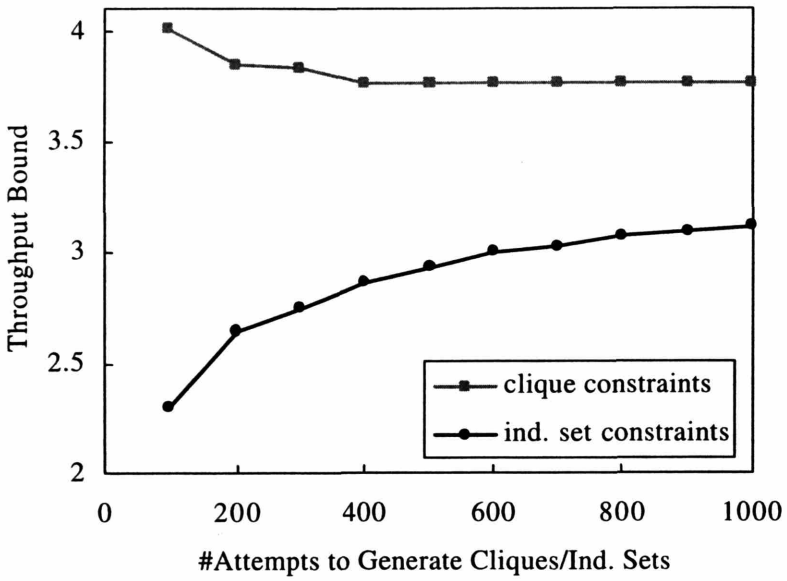


Figure 8-11: Upper and Lower Bounds for Throughput of Two-Phase Routing in WMN: 50-node topology, 200 mW Transmission Power.

the maximum throughput that is achieved by a feasible solution, we can use this as an estimate of the throughput for the omnidirectional antenna case (we know that it is within 8% of the maximum throughput). Using this estimate, the throughput reduction as result of link interference is by a factor of 2.7 compared to the directional antenna case.

We repeat the experiment on the same 50-node topology but with a node transmission power $P = 200$ mW. The results are plotted in Figure 8-11. In this example also, the upper bound flattens quickly – the value is about 3.75 corresponding to 400 attempts to generate maximal cliques in the conflict graph. The lower bound is about 3.12 corresponding to 1000 attempts to generate maximal independent sets in the conflict graph. We were able to improve the lower bound further to 3.53 using 10000 attempts. The gap between the upper and lower bounds thus obtained is about 6%. Using the lower bound as an estimate, the throughput reduction as result of link interference in this case is by a factor of 4.8 compared to the directional antenna case.

Thus, the impact of interference on reducing throughput is greater when the node transmission power (and, hence interference range) is higher.

Bibliography

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, February 1993.

[A98] S. M. Alamouti, "A Simple Transmit Diversity Technique for Wireless Communications," *IEEE JSAC*, Vol. 16, No. 8, March 1998, pp. 1451–1458.

[ACFKR03] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Räcke, "Optimal oblivious routing in polynomial time", *35th ACM Symposium on the Theory of Computing (STOC)*, 2003.

[B95] F. Baker, "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[BPSK97] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links", *IEEE/ACM Transactions on Networking*, December 1997.

[BK03] W. Ben-Ameur and H. Kerivin, "New Economical Virtual Private Networks", Communications of the ACM, Vol. 46, No. 6, pp. 69-73, June 2003.

[BPS99] J. C. R. Bennett, C. Partridge, and N. Shectman, "Packet Reordering is Not Pathological Network Behavior", *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 789-798, December 1999.

[BA02] E. Blanton and M. Allman, "On Making TCP More Robust to Packet Reordering", *ACM Computer Communication Review*, vol. 32, no. 1, pp. 20-30, January 2002.

[BCG05] R. Bruno, M. Conti, and E. Gregori, "Mesh Networks: Commodity Multihop Ad Hoc Networks", *IEEE Communications Magazine*, March 2005, pp. 123-131.

[CFSD90] J. Case, M. Fedor, M. Schoffstall, J. Davin, "Simple Network Management Protocol (SNMP)", RFC 1157, May 1990.

[CLJ02] C. -S. Chang, D. -S. Lee, and Y. -S. Jou, "Load balanced Birkhoff-von-Neumann switches, Part I: one-stage buffering", *Computer Communications*, vol. 25, pp. 611-622, 2002.

[COSS05] C. Chekuri, G. Oriolo, M. G. Scutella, F. B. Shepherd, "Hardness of Robust Network Design", *International Network Optimization Conference (INOC) 2005*, March 2005.

[Cisco97] "Configuring OSPF", Cisco Systems Product Documentation, http://www.cisco.com/univercd/home/home.htm.

[D00] Douglas E. Comer, *Internetworking with TCP/IP Vol.1: Principles, Protocols, and Architecture*, Prentice Hall, 4th edition, January 2000.

[CPLEX] ILOG CPLEX, http://www.ilog.com.

[DABM03] D. S. J. De Couto, D. Aguayo, J. Bicket, R. Morris "A High-Throughput Path Metric for Multi-hop Wireless Routing", *ACM MOBICOM* 2003, pp. 134–146.

[DPZ04] R. Draves, J. Padhye, B. Zill, "Comparison of Routing Metrics for Static Multi-hop Wireless Networks", *ACM SIGCOMM* 2004. pp. 133–144.

[DGGMR99] 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.

[EGOS05] F. Eisenbrand, F. Grandoni, G. Oriolo, and M. Skutella, "New approaches for Virtual Private Network Design", Proceedings of *ICALP 2005*, LNCS vol. 3580, pp. 1151.

[ER04] T. Erlebach and M. Rüegg, "Optimal Bandwidth Reservation in Hose-Model VPNs with Multi-Path Routing", *IEEE Infocom 2004*, March 2004.

[FST97] J. Andrew Fingerhut, Subhash Suri, and Jonathan S. Turner, "Designing Least-Cost Nonblocking Broadband Networks", *Journal of Algorithms*, 24(2), pp. 287-309, 1997.

[F00] L. K. Fleischer, "Approximating Fractional Multicommodity Flow Independent of the Number of Commodities", *SIAM Journal on Discrete Mathematics*, Vol. 13, No. 4, ppp. 505-520, May 2000.

[FG98] G. J. Foschini, M. J. Gans, "On the Limits of Wireless Communication in a Fading Environment When Using Multiple Antennas," *Wireless Personal Communications*, Vol. 6, No. 3, March 1998, pp. 311–335.

[GJ79] M. R. Garey and D. S. Johnson, *Computers and Intractability : A Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979.

[GK02] N. Garg and R. Khandekar, "Fast Approximation Algorithms for Fractional Steiner Forest and Related Problems", Proceedings of *43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 500-509, 2002.

[GK98] 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.

[G02] M. Gast, *802.11 Wireless Networks: The Definitive Guide*, O'Reilly and Associates, 2002.

[GRKL01] A. Goel, K. G. Ramakrishnan, D. Kataria, and D. Logothetis, "Efficient Computation of Delay-Sensitive Routes from One Source to All Destinations", *IEEE Infocom 2001*, April 2001.

[GLS88] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, 1988.

[G03] W. D. Grover, *Mesh-based Survivable Transport Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking*, Prentice Hall, 2003.

[GKKRY01] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, B. Yener, "Provisioning a Virtual Private Network: A Network Design Problem for Multicommodity Flow", *ACM Symposium on Theory of Computing (STOC) 2001*, July 2001.

[GK00] P. Gupta, and P. R. Kumar, "The Capacity of Wireless Networks", *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 910-917, 2000.

[HS88] B. Hajek and G. Sasaki, "Link Scheduling in Polynomial Time", *IEEE Transactions on Information Theory*, 34(5), pp. 910-917, 1988.

[H92] R. Hassin, "Approximation schemes for the restricted shortest path problem", *Mathematics of Operations Research 17*, 36-42, 1992.

[HKS05] C. A. J. Hurkens, J. C. M. Keijsper, and L. Stougie, "Virtual Private Network Design: A Proof of the Tree Routing Conjecture on Ring Networks", Proceeedings of *IPCO 2005*, LNCS 3509, pp. 407-421, 2005.

[ILO02] G. Italiano, S. Leonardi, and G. Oriolo, "Design of networks in the hose model: the balanced case", Proceedings of *ARACNE 2002*, pp. 65-76, 2002.

[IBTM03] S. Iyer, S. Bhattacharyya, N. Taft, C. Diot, "An approach to alleviate link overload as observed on an IP backbone", *IEEE Infocom 2003*, March 2003.

[JPPQ03] K. Jain., J. Padhye, V. N. Padmanabhan, L. Qiu, "Impact of Interference on Multi-hop Wireless Network Performance", *ACM MobiCom 2003*, September 2003.

[JIDKT03] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone", *IEEE Infocom 2003*, March 2003.

[JMB01] David B. Johnson, David A. Maltz, and Josh Broch, "DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks", in *Ad Hoc Networking*, edited by Charles E. Perkins, Chapter 5, pp. 139-172, Addison-Wesley, 2001.

[JMH04] D. B. Johnson, D. A. Maltz, Y-C. Hu, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)", IETF Internet draft <draft-ietf-manet-dsr-10.txt>, July 2004.

[KSV00] Y. B. Ko, V. Shankarkumar, N. H. Vaidya "Medium Access Protocols Using Directional Antennas in Adhoc Networks", *IEEE INFOCOM*, 2000, pp. 13–21.

[KLOS06a] M. Kodialam, T. V. Lakshman, J. B. Orlin, and S. Sengupta, "A Versatile Scheme for Routing Highly Variable Traffic in Service Overlays and IP Backbones", To Appear in *IEEE Infocom 2006*, April 2006.

[KLOS06b] M. Kodialam, T. V. Lakshman, J. B. Orlin, and S. Sengupta, "Preconfiguring IP-over-Optical Networks to Handle Router Failures and Unpredictable Traffic", To Appear in *IEEE Infocom 2006*, April 2006.

[KLS04a] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Efficient, Robust Routing in Highly Dynamic Environments", Stanford Workshop on Load-Balancing, May 2004.

[KLS04b] 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.

[KLS06] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Maximum Throughput Routing of Traffic in the Hose Model", To Appear in *IEEE Infocom 2006*, April 2006.

[KN03] M. Kodialam and T. Nandagopal, "Characterizing Achievable Rates in Multi-hop Wireless Networks: The Joint Routing and Scheduling Problem", *ACM MOBICOM 2003*, November 2003.

[KRSY01] A. Kumar, R. Rastogi, A. Silberschatz , B. Yener, "Algorithms for provisioning VPNs in the hose model", *ACM SIGCOMM 2001*, August 2001.

[LAJ98] C. Labovitz, A. Ahuja, and F. Jahanian, "Experimental Study of Internet Stability and Backbone Failures", Proceedings of *29th International Symposium on Fault-Tolerant Computing (FTCS)*, Madison, Wisconsin, pp. 278-285, June 1999.

[La72] E. L. Lawler, "A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem", *Management Science*, vol. 18, pp. 401-405, 1972.

[LMS92] C. Li, S. T. McCormick, D. Simchi-Levi, "Finding Disjoint Paths with Different Path Costs: Complexity and Algorithms", *Networks*, vol. 22, 1992, pp. 653-667.

[Lo72] L. Lovasz, "Characterization of Perfect Graphs", *Journal of Combinatorial Theory*, Series B, vol. 13, pp. 95-98, 1972.

[MSWA02] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring link weights using end-to-end measurements", *2nd ACM Internet Measurement Workshop*, 2002.

[MTSBD02] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, C. Diot, "Traffic Matrix Estimation: Existing Techniques and New Directions", *ACM SIGCOMM 2002*, August 2002.

[MNS04] Mesh Networking Summit 2004, http://research.microsoft.com/meshsummit/techprogram.aspx, June 2004.

[N01] W. B. Norton, "Internet Service Providers and Peering", Equinix white paper, 2001, http://www.equinix.com.

[P93] C. A. Phillips, "The network inhibition problem", *Proc. 25th Annual ACM Symposium on Theory of Computing (STOC)*, 776-785, 1993.

[RS02] R. Ramaswami and K. N. Sivarajan, *Optical Networks: A Practical Perspective*, Morgan Kaufmann Publishers, 2002.

[RS99] M. Reardon and S. Saunders, "Terabit Trouble", *Data Communications*, August 1999, pp. 11-16.

[RVC01] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

[S86] A. Schrijver, *Theory of Linear and Integer Programming*, John Wiley & Sons, 1986.

[SSK03] S. Sengupta, D. Saha, and V. P. Kumar, "Switched Optical Backbone for Cost-effective Scalable Core IP Networks", *IEEE Communications Magazine*, June 2003.

[SM90] F. Shahrokhi and D. Matula, "The Maximum Concurrent Flow Problem", *Journal of ACM*, 37(2):318-334, 1990.

[SMW02] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP Topologies with Rocketfuel", Proceedings of *ACM SIGCOMM 2002*, pp. 133-145, August 2002.

[SMWH] N. Spring, R. Mahajan, D. Wetherall, and H. Hagerstrom, Rocketfuel: An ISP topology mapping engine, http://www.cs.washington.edu/research/networking/rocketfuel/.

[SR03] A. Spyropoulos, C. S. Raghavendra, "Asymptotic Capacity Bounds for Adhoc Networks Revisited: The Directional and Smart Antenna Cases", *IEEE GLOBECOM*, 2003, pp. 1216–1220.

[SAZSS02] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, S. Surana, "Internet Indirection Infrastructure", *ACM SIGCOMM 2002*, August 2002.

[SAHS03] K. Sundaresan, V. Anantharaman, H.Y. Hsieh, R. Sivakumar, "ATP: A Reliable Transport Protocol for Adhoc Networks", *ACM MOBIHOC*, 2003, pp. 64–75.

[TSGR04] R. Teixeira, A. Shaikh, T. Griffin, J. Rexford, "Dynamics of Hot-Potato Routing in IP Networks", *ACM SIGMETRICS 2004*, June 2004.

[UMTS98] "Universal Mobile Telecommunications System (UMTS): Selection Procedures of the Choice of Radio Transmission Technologies of the UMTS" (UMTS 30.03 Version 3.2.0), European Technical Standards Institute (ETSI), Technical Report TR 101 112 V3.2.0, April 1998.

[V82] L. G. Valiant, "A scheme for fast parallel communication", *SIAM Journal on Computing*, 11(7), pp. 350-361, 1982.

[XPMS01] G. Xylomenos, G. C. Polyzos, P. Mahonen, M. Saaranen, "TCP Performance Issues over Wireless Links", *IEEE Communications Magazine*, 39(4), 2001, pp. 52–58.

[Y71] J. Y. Yen, "Finding the k shortest loop less paths in a network", *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[Z95] G. Ziegler, *Lectures on Polytopes*, Graduate Texts in Mathematics, Vol. 152, Springer- Verlag, New York, 1995.

[ZM04] R. Zhang and N. McKeown, "Designing a Predictable Internet Backbone Network", *Third Workshop on Hot Topics in Networks (HotNets-III)*, November 2004.

[ZRDG03] Y. Zhang, M. Roughan, N. Duffield, A. Greenberg, "Fast Accurate Computation of Large-Scale IP Traffic Matrices from Link Loads", *ACM SIGMETRICS 2003*, June 2003.

[ZRLD03] Y. Zhang, M. Roughan, C. Lund, and D. Donoho, "An Information-Theoretic Approach to Traffic Matrix Estimation", Proceedings of *ACM SIGCOMM 2003*, pp. 301-312, August 2003.