Traffic Engineering in IP and MPLS Networks

Gabi Nakibly
Traffic Engineering in IP and MPLS Networks

Research Thesis

Submitted in Partial Fulfillment of the Requirements

For the

Degree of Doctor of Philosophy

Gabi Nakibly

Submitted to the Senate of the Technion — Israel Institute of Technology

KISLEV 5768          HAIFA          DECEMBER 2007
I would like to thank my supervisor Reuven for his guidance and sound advice throughout the course of this research. Reuven, it was a privilege working with you. I am grateful for the loving support and help of my family that has been there for me throughout all these years. A special thanks goes to my Shiri - my wife - for her encouragement and pep talks, and for putting up with the long hours of many paper rewritings and corrections.

The generous financial help of the Technion is gratefully acknowledged.
# Contents

Abstract 1

1 Introduction 3
   1.1 The Internet and Traffic Engineering 3
   1.2 Overview of the Thesis 4

2 On the Computational Complexity and Effectiveness of N-hub Shortest-Path Routing 7
   2.1 Background 7
   2.2 N-hub Shortest-Path Routing in IP Networks: Implementation and Related Work 10
   2.3 Problem Definition and Complexity 12
      2.3.1 Problem Definition 12
      2.3.2 NP-Completeness of the N-hub routing problem 14
      2.3.3 On the Approximation Hardness of N-hub 17
   2.4 A Probabilistic Approximation Algorithm for the Offline N-hub Problem 18
   2.5 The Approximation Technique 18
   2.6 The Approximation Algorithm 19
   2.7 Online Approximation Algorithms 22
   2.8 Simulation Study 27
   2.9 Conclusions 32
   2.10 Appendix: A linear program for the general routing problem 33

3 A Traffic Engineering Approach for Placement and Selection of Network Services 35
   3.1 Background 35
   3.2 Application Scenarios 37
      3.2.1 Open Access Networks 37
      3.2.2 VoIP communication through border session controllers 38
   3.3 Related Work 40
   3.4 Problem Model 41
3.5 Algorithms for The Placement and Selection Problems .............................................................................. 44
  3.5.1 Algorithms for the Placement Problem .................................................................................................. 44
  3.5.2 Algorithms for the Selection Problem .................................................................................................. 46
3.6 Simulation Study ........................................................................................................................................... 48
3.7 Conclusions .................................................................................................................................................. 60
3.8 Appendix: A Proof for the Approximation Ratio of Algorithm 1 for the Gateway Placement Problem .............................................................................................................. 60

4 Maximizing Restorable Throughput in MPLS Networks ............................................................................... 65
  4.1 Background ................................................................................................................................................ 65
  4.2 Related Work .............................................................................................................................................. 72
  4.3 Problem Definition and Computational Complexity .................................................................................... 74
    4.3.1 The Splittable Primary-restricted Restorable Flow Problem ................................................................ 76
    4.3.2 The Unsplittable Primary-restricted Restorable Flow Problem ................................................................. 79
    4.3.3 The Unsplittable and Splittable Restorable Flow Problems .................................................................. 80
  4.4 Algorithms for U-PRFP and U-RFP ............................................................................................................. 82
    4.4.1 Approximation Algorithms ................................................................................................................. 82
    4.4.2 Heuristics .............................................................................................................................................. 85
  4.5 Simulation Study ........................................................................................................................................... 87
  4.6 Node Failure ................................................................................................................................................ 93
    4.6.1 The Splittable Problems (S-PRFP and S-RFP) ......................................................................................... 93
    4.6.2 The Unsplittable Problems (U-PRFP and U-RFP) .................................................................................. 95
  4.7 Extension to MPLS over WDM lightpaths .................................................................................................. 96
  4.8 Conclusions .................................................................................................................................................. 96

5 Conclusions .................................................................................................................................................. 97

References ....................................................................................................................................................... 98
List of Figures

2.1 An example of N-hub routing ........................................ 9
2.2 N-hub routing as a compromise between efficiency and simplicity .. 13
2.3 An example of a network topology .................................. 13
2.4 An instance for the 1-hub problem constructed from the instance of SAT given in (2.1) .................................................. 16
2.5 Performance of Algorithm-1 for 1-hub for different link densities .. 28
2.6 Performance of the Algorithm-1 for 1-hub for small network (|V| = 10, |E| = 20) ............................................................... 30
2.7 Performance of Algorithm-1 for 1-hub, 2-hub and 3-hub (|V| = 25 and |E| = 75) ............................................................... 31
2.8 Performance of a random algorithm for 1-hub for different network sizes 32
2.9 Performance on an ISP (Exodus). |V| = 80, |E| = 294 .................. 33
2.10 No. of connections vs. max. tolerated load ........................ 34

3.1 The schematic structure of a typical open access network ........... 38
3.2 VoIP communication through a border session controller ............ 39
3.3 The potential effectiveness of the various placement schemes ...... 50
3.4 The relative performance for every combination of placement and selection algorithms ........................................... 52
3.5 The admitted traffic for selected combinations of algorithms, as a function of the number of gateways ............................. 53
3.6 The admitted traffic for the best combinations of algorithms, as a function of the number of gateways, for the Exodus ISP ........... 54
3.7 The admitted traffic for the best combinations of algorithms, as a function of the number of gateways, for the gravity demand model .... 55
3.8 The admitted traffic for some combinations of algorithms, as a function of the fraction of serviced traffic .......................... 56
3.9 The relative performance of Exp, as a function of the update interval (gateways were placed using Prob) ............................. 57
3.10 The admitted traffic for selected combinations of algorithms, when random noise is added to the traffic distribution information .... 58
3.11 The effect of AS evolution on admitted traffic ........................ 59
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Illustrations of the various recovery schemes</td>
<td>66</td>
</tr>
<tr>
<td>4.2</td>
<td>An illustration of the drawback of SCA</td>
<td>70</td>
</tr>
<tr>
<td>4.3</td>
<td>Bandwidth sharing of two backup paths</td>
<td>71</td>
</tr>
<tr>
<td>4.4</td>
<td>An example of the reduction in the proof of Theorem 9</td>
<td>80</td>
</tr>
<tr>
<td>4.5</td>
<td>Relative performance for OPT-S-PRFP for various MPLS domains</td>
<td>88</td>
</tr>
<tr>
<td>4.6</td>
<td>Relative performance of the heuristics for U-PRFP for recovery schemes</td>
<td>91</td>
</tr>
<tr>
<td>4.7</td>
<td>Relative performance of the heuristics for U-PRFP as a function of the offered load</td>
<td>92</td>
</tr>
<tr>
<td>4.8</td>
<td>The relative performance of S-PRFP using RLR</td>
<td>93</td>
</tr>
<tr>
<td>4.9</td>
<td>The relative performance of U-PRFP-1 using the various recovery schemes for the Exodus ISP</td>
<td>94</td>
</tr>
</tbody>
</table>
List of Tables

4.1 Abbreviations and acronyms used in Chapter 4 .......................... 67
4.2 Characteristics of the recovery scheme ................................. 68
4.3 Summary of the computational complexities of the problems ...... 75
Abstract

Traffic engineering is a set of actions whose goal is to evaluate and optimize the performance of operational networks. The performance of an operational network is enhanced by addressing traffic-oriented performance measures, such as delay, delay variation, packet loss, and throughput, while utilizing network resources economically and reliably. One of the most important traffic engineering actions is the control of the routing function in the network so that traffic can be steered through it as effectively as possible.

This thesis contains three parts. The first two parts address traffic engineering in IP networks and the third addresses traffic engineering in MPLS networks. In the first part we study the computational complexity and effectiveness of a concept we term “N-hub Shortest-Path Routing” in IP networks. N-hub Shortest-Path Routing allows the ingress node of a routing domain to determine up to N intermediate nodes (“hubs”) through which a packet will pass before reaching its final destination. This facilitates better utilization of the network resources, while allowing the network routers to continue to employ the simple and well-known shortest-path routing paradigm. Although this concept has been proposed in the past, this thesis is the first to investigate it in depth. We apply N-hub Shortest-Path Routing to the problem of minimizing the maximum load in the network. We show that the resulting routing problem is NP-complete and hard to approximate. However, we propose efficient algorithms for solving it both in the online and the offline contexts. Our results show that N-hub Shortest-Path Routing can increase network utilization significantly even for $N = 1$. Hence, this routing paradigm should be considered as a powerful mechanism for future datagram routing in the Internet.

In the second part, we consider the application of “N-hub Shortest-Path Routing” to network services traffic. Network services are provided by means of dedicated service gateways, through which traffic flows are directed. Existing work on service gateway placement has been primarily focused on minimizing the length of the routes through these gateways. Only limited attention has been paid to the effect these routes have on overall network performance. We propose a novel approach for the service placement problem, which takes into account traffic engineering considerations. Rather than trying to minimize the length of the traffic flow detours, we take advantage of them in order to enhance the overall network performance. We divide
the problem into two sub-problems: finding the best location for each service gateway, and selecting the best service gateway for each flow. We propose efficient algorithms for both problems and study their performance. Our main contribution is showing that placement and selection of network services can be used as effective tools for traffic engineering.

In the third part, we consider the MPLS recovery mechanisms. These mechanisms are increasing in popularity because they can guarantee fast restoration and high QoS assurance. Their main advantage is that their backup paths are established in advance, before a failure event takes place. Most research on the establishment of primary and backup paths has focused on minimizing the added capacity required by the backup paths in the network. However, this so-called Spare Capacity Allocation (SCA) metric is less practical for network operators who have a fixed capacitated network and want to maximize their revenues. In this thesis we present a comprehensive study on restorable throughput maximization in MPLS networks. We present the first polynomial-time algorithms for the splittable version of the problem. For the unsplittable version, we provide a lower bound for the approximation ratio and propose an approximation algorithm with an almost identical bound. We present efficient heuristics which are shown to have excellent performance. One of our most important conclusions is that when one seeks to maximize revenue, local recovery should be the recovery scheme of choice.
Chapter 1

Introduction

1.1 The Internet and Traffic Engineering

Over the past decade the popularity of the Internet has increased dramatically. Due to its versatility, cost-effectiveness and reach, the Internet is considered the transport infrastructure of choice for many diverse applications. The Internet is used now by over one billion people all over the world as a platform for communications, information, entertainment and business. It is considered by many as a critical infrastructure, necessary to the running of our day-to-day lives. This popularity rests on the Internet’s ability to supply the quality of experience users expect of it. The quality of user experience is dependent on many traffic-oriented performance measures such as delay, delay variation, packet loss, and throughput. The aim of network operators and service providers is to consistently keep these measures at acceptable levels for as many users as possible at a minimum cost.

One of the most important tools for achieving this aim is traffic engineering. Traffic engineering is a set of actions whose goal is to evaluate and optimize the performance of operational networks while utilizing network resources as economically and as reliably as possible. This thesis focuses on traffic engineering actions involving routing controls. These routing controls augment or replace the standard shortest-path routing scheme by steering traffic through the network as effectively as possible. In particular, their objective is to distribute the traffic demand evenly throughout the network by directing traffic from hot spots to underutilized areas.

The shortest-path routing scheme has been the predominant intra-domain routing scheme in the Internet from its early days. The source of a packet specifies the destination address, and each router along the route forwards the packet to a neighbor located “closer” to the destination. Distance between nodes are determined based on pre-configured link weights. Usually these weights are static, namely the cost of a path is dependent on the network topologies rather than on the dynamics of the network traffic.
The shortest-path routing paradigm is known to be simple and efficient. It does not place a heavy processing burden on the routers. However, while this scheme finds the shortest path for each pair of nodes and thus minimizes the bandwidth consumed by every packet, it does not guarantee full utilization of the network resources under high traffic loads. When the network load is not uniformly distributed, some of the routers introduce an excessive delay while others are underutilized.

Much research has been conducted in a search for an alternative routing paradigm that would address this drawback of shortest-path routing. The sought paradigm should utilize the network resources more efficiently and minimize the probability of congestion, thereby achieving better delay-throughput behavior than traditional shortest-path routing. In addition, such a scheme should interoperate seamlessly with network routers that continue to employ the shortest-path routing paradigm.

Most of the routing schemes proposed in the past are able to employ more than one path between every source-destination pair. Generally, these schemes base their routing decisions on the load imposed on every network link. When a particular link, or an area, becomes congested, some of the routes are modified. Some routing schemes find an alternate data path only when the standard path is highly congested \[105\]. In \[19, 37, 104\], alternate routes are found for every source-destination pair even if the standard route is not heavily loaded. Several loop-free paths are found in advance and the load is distributed between them. However, due to the complexity of these schemes, their increased processing burden, and their considerable deviation from the conventional shortest-path routing paradigm, not one of them has been adopted for the Internet. A major drawback of many proposed routing schemes is that they must be deployed over the lion’s share of the routing domain in order to be effective.

### 1.2 Overview of the Thesis

In the first part of thesis we present an investigation of a novel routing scheme that takes advantage of a concept we refer to as “N-hub Shortest-Path Routing,” or simply N-hub routing. This concept can be implemented using several existing IP mechanisms. N-hub routing allows the ingress router of a routing domain to determine one or more intermediate nodes (“hubs”) that a packet will traverse before reaching its final destination. Using the concept of N-hub routing, the routing protocol gains better control over the routing process, while the network routers continue to employ the shortest-path paradigm for building their routing tables. Although this concept is not employed today in the Internet, we show it is a powerful tool that should be considered in the context of traffic engineering and QoS.

In the second part of the thesis we apply the N-hub Shortest-Path Routing scheme on network services’ traffic. As the Internet popularity increases, there is a growing
demand for services that facilitate and enhance interoperability, performance and security of communication between two or more parties. Examples for such services are voice and video conversion, protocol translation, caching, compression, QoS control, authentication, encryption and intrusion detection. Many of these services require the intervention of intermediate service gateways, like firewalls, VoIP gateways, NAT routers, VPN gateways and broadband access servers. Hence, the serviced traffic traverses the shortest path from the source to the service gateway, and then the shortest path from the gateway to the destination. In a sense the service gateways act as hubs.

Traditionally, such service gateways have been placed on the boundary of an Autonomous System (AS), since all inter-domain traffic passes through it. However, there is a growing trend to place network services inside the AS. It was first shown by [28] that FTP traffic can be significantly reduced by placing caches in strategic locations inside the AS backbone. Since then, there has been a large volume of work that demonstrates the benefits of well-planned placement strategies in a variety of service contexts [22, 43, 51, 68, 79, 89]. Such strategies concentrate mainly on placing the service gateways in a way that minimizes the average length of the traversed routes.

In this work we propose a novel approach for the service placement problem. The idea is to leverage service placement in the AS to facilitate traffic engineering. We take advantage of the ability of the source node to route traffic flows through service gateways, in order to control the load in certain areas of the AS. Instead of trying to minimize the length of detour routes, we extend and utilize these detours in order to relieve congestion in hotspots. To the best of our knowledge, this is the first work that employs such a traffic-engineering approach for the service placement and gateway selection problems.

In the third part of the thesis we address traffic engineering in restorable MPLS networks. IP networks should support real-time applications that require stringent availability and reliability, such as Voice over IP and virtual private networks. Unfortunately, failures are still common in the daily operation of networks, for reasons such as improper configuration, faulty interfaces, and accidental fiber cuts [48, 76]. Therefore, mechanisms that restore the flow of traffic quickly and efficiently after a failure are essential.

Many network operators do not rely only on IP routing protocols to restore traffic, but also employ recovery mechanisms in Layer 1 and Layer 2 protocols such as WDM, SONET/SDH, and MPLS. These recovery mechanisms guarantee fast restoration and high QoS assurance because they establish backup paths in advance, before a failure event takes place. Such recovery mechanisms are usually referred to as “protection” mechanisms [96, 97].

Most past research on the selection of backup LSPs is directed at minimizing the total bandwidth reserved for the backup LSPs. To this end, backup LSPs are routed to maximize their bandwidth sharing. This optimization metric is usually referred to as
Spare Capacity Allocation (SCA). Models that seek to optimize SCA usually consider a network whose links have unbounded capacity, and a cost function associated with bandwidth usage. However, while minimizing the cost of building the backup LSPs is an important goal, network operators usually face a different optimization problem. They have a network with finite link capacities and seek to maximize their revenue by maximizing the traffic the network can accommodate. Another drawback of the SCA optimization for network operators is that the cost associated with an established LSP does not depend on the load imposed on the selected route. In other words, there is no incentive for load balancing. Hence, in this work we present a comprehensive study of the problem of constructing primary and backup LSPs while maximizing throughput. We believe, such an optimization metric is much more relevant for network operators. We investigate the computational complexity of maximizing throughput. We offer algorithms with bounded approximation ratio as well as efficient heuristics that are shown to have excellent performance.
Chapter 2

On the Computational Complexity and Effectiveness of N-hub Shortest-Path Routing

In this chapter we study the computational complexity and effectiveness of a concept we term “N-hub Shortest-Path Routing” in IP networks. We show that applying this concept can achieve better utilization of the network resources, while allowing the network routers to continue to employ the simple and well-known shortest-path routing paradigm. We apply N-hub Shortest-Path Routing to the problem of minimizing the maximum load in the network. Our results show that N-hub Shortest-Path Routing can increase network utilization significantly even for $N = 1$.

The work described in this chapter was originally published in [23].

2.1 Background

Intra-AS routing in the Internet is based on the hop-by-hop shortest-path paradigm. The source of a packet specifies the destination address, and each router along the route forwards the packet to a neighbor located “closer” to the destination. Since the routing is usually static, i.e., the cost of a path is dependent on the network topologies rather than on the dynamics of the network traffic, a single route is used for every source-destination pair.

The shortest-path routing paradigm is known to be simple and efficient. It does not place a heavy processing burden on the routers and usually requires at most one entry per destination network in every router. However, while this scheme finds the shortest path for each pair of nodes and thus minimizes the bandwidth consumed by every packet, it does not guarantee full utilization of the network resources under high traffic loads. When the network load is not uniformly distributed, some of the
routers introduce an excessive delay while others are underutilized. In some cases this non-optimized use of network resources may introduce not only excessive delays but also incur a high packet loss rate.

Much research has been conducted in a search for an alternative routing paradigm that would address this drawback of shortest-path routing. The sought paradigm should utilize the network resources more efficiently and minimize the probability of congestion, thereby achieving better delay-throughput behavior than traditional shortest-path routing. In addition, such a scheme should be practical in terms of the volume of control information exchanged by the routers, the memory requirement, the processing burden imposed by every packet, and so forth. Finally, such a scheme should interoperate seamlessly with network routers that continue to employ the shortest-path routing paradigm.

Most of the routing schemes proposed in the past are able to employ more than one path between every source-destination pair. Generally, these schemes base their routing decisions on the load imposed on every network link. When a particular link, or an area, becomes congested, some of the routes are modified. Some routing schemes find an alternate data path only when the standard path is highly congested [105]. In [19, 37, 104], alternate routes are found for every source-destination pair even if the standard route is not heavily loaded. Several loop-free paths are found in advance and the load is distributed between them. However, due to the complexity of these schemes, their increased processing burden, and their considerable deviation from the conventional shortest-path routing paradigm, not one of them has been adopted for the Internet. A major drawback of many proposed routing schemes is that they must be deployed over the lion’s share of the routing domain in order to be effective.

This chapter investigates a routing scheme that takes advantage of a concept we refer to as “N-hub Shortest-Path Routing,” or simply N-hub routing. This concept can be implemented using several existing IP mechanisms, as will be discussed in Section 2.2. N-hub routing allows the ingress router of a routing domain to determine one or more intermediate nodes (“hubs”) that a packet will traverse before reaching its final destination. Fig. 2.1 illustrates this concept. The figure shows three paths for a packet whose source and destination nodes are A and D. The first path, path-1, is the shortest path. Path-2 uses node G as a single hub. Packets are routed first on the shortest path from A to G and then on the shortest path from G to D. Such a route is likely to improve the throughput if the links B – C or C – D are heavily loaded while the links B – G, G – H and H – D are underutilized. Path-3 uses 2 hubs: F and B. Packets are routed first on the shortest path from A to F, then on the shortest path from F to B, and finally on the shortest path from B to D. It is evident from the example above that N-hub routing is a generalization of shortest-path routing, because shortest-path routing is equivalent to N-hub routing with N = 0.

Using the concept of N-hub routing, the routing protocol gains better control over the routing process, while the network routers continue to employ the shortest-path
paradigm for building their routing tables. Although this concept is not employed today in the Internet, we think it is a powerful tool that should be considered in the context of traffic engineering and QoS.

It is important to note the practical benefits of N-hub Shortest-Path Routing over virtual-circuit routing. First, N-hub routing can be implemented in networks that usually do not employ virtual-circuit routing technologies (such as MPLS [96]). In particular, it can be implemented in sensor networks and ad hoc (mobile) networks. Second, when virtual-circuit routing is used, only one or two routes are usually established between every two routers. Therefore, it is not possible to react to changes in the traffic pattern before the time-consuming and labor-intensive building of new routes. In contrast, an N-hub route can be changed immediately according to changes in the link loads, without having to set up additional routes in advance. Third, N-hub routing imposes additional processing and memory burden on the hubs and the source edge routers only, while the other nodes employ regular shortest-path routing. In virtual-circuit routing this burden is imposed on all the nodes along the path. This is especially significant when each node has to maintain several thousands of explicit routes.

The ingress router of a routing domain should be responsible for determining the intermediate router(s) through which the packets of each flow will be routed. To this end, the router may use information it acquires regarding the load distribution in the network by means of a link-state flooding protocol like OSPF-TE [57]. For a typical case scenario for N-hub routing in an ISP AS, consider a DiffServ [33] domain, which supports the Expedited Forwarding (EF) Per Hop Behavior. When an edge router receives a packet of an EF flow (e.g., a Voice over IP flow), and N-hub routing is not supported, the router has no option but to forward the packet along the default (shortest) path or to drop it. With N-hub routing support, however, the edge router uses information about the load distribution in the entire domain, as can be obtained using OSPF-TE [6], in order to determine the hub(s) that define the least congested route. This list of hub(s) is added to the packet, and is also kept in the router’s local
flow table. When subsequent packets of the same flow are received by this router, it identifies them as belonging to the same flow, e.g., using the flow label of IPv6, and fetches from its table the list of hub(s) associated with this flow. Once every time-out period, the router checks if there is a better N-hub route that can be used by the considered flow.

To the best of our knowledge, this work is the first to propose a thorough theoretical and practical investigation of N-hub Shortest-Path Routing. The contribution of this work is fourfold. First, we define the N-hub shortest-path problem as an optimization problem, and show that from a computational complexity perspective, N-hub is "closer" to virtual circuit (|V|-hub) routing than to shortest path (0-hub) routing. This is because N-hub is NP-complete, and it has no polynomial approximation scheme (PTAS). Second, we develop a probabilistic approximation algorithm for the N-hub problem. Third, we show that online algorithms originally designed for multicommodity routing maintain their competitive ratio for N-hub routing. Fourth, we show that in practice, one hub for every flow is sufficient to obtain results that are almost equal to those obtained by optimal algorithms for the splittable multicommodity flow problem. These results are upper bounds for the results that can be obtained by optimal algorithms for virtual circuit routing.

The rest of this chapter is organized as follows. In Section 2.2 we discuss related work and the various mechanisms that can be employed to implement N-hub routing. In Section 2.3 we define the N-hub routing problem and review its computational complexity. In Section 2.7 we present several approximation algorithms for the online context. The competitive ratio of these algorithms is discussed, and one of them is shown to have the best competitive ratio that can be obtained for this problem. In Section 2.8 we present simulation results that show the potential effectiveness of N-hub routing in general, and the effectiveness of the various algorithms proposed in the chapter. Finally, Section 2.9 concludes the chapter.

2.2 N-hub Shortest-Path Routing in IP Networks: Implementation and Related Work

N-hub Shortest-Path Routing can be implemented using several existing mechanisms. A straightforward way is to take advantage of the IPv4 Loose Source-Routing option [88]. When this option is used, the IP header is extended by a list of the addresses of the intermediate node(s) the packet must traverse. However, this option, much like any other IPv4 option, is rarely used, mainly because of the heavy processing burden imposed on the general purpose CPU of the router when an IPv4 header contains any optional field. Moreover, there are some notable security issues related to this option [11]. In [39], it is noted that only 8% of Internet routers are source-routing capable.
As opposed to IPv4, IPv6 \cite{29} has a more “built-in” support for N-hub routing. The primary header of an IPv6 packet can be followed by flexible extension headers. These headers can, for example, indicate the IP addresses of the network routers the packet should traverse en route to its destination.

Another way to implement N-hub routing in IPv4 is to use IP-in-IP encapsulation \cite{87}. In this case, an IP header indicating the final destination is encapsulated in the payload of another IP header. The latter header contains, in its destination address field, the IP address of an intermediate router. The total number of headers is therefore equal to the number of hubs plus 1.

N-hub routing can also be implemented through an overlay network \cite{5}. In an overlay network the source sends a packet to the first hub, while adding to its payload information that identifies the next hubs and the final destination. Each hub uses this information to route the packet to the next hub.

Another powerful way to implement the N-hub routing paradigm is to use MPLS \cite{96}. MPLS is a virtual circuit technology that allows an MPLS ingress node to set up a tunnel over the shortest path or over an explicit path to an egress node. An explicit path contains a list of intermediate nodes. The route between two consecutive nodes in the list is either strict or loose. A loose route may contain other nodes. Therefore, N-hub shortest path routing can be viewed as a special case of the MPLS explicit route option. With respect to MPLS, our results imply that an explicit strict route need not be specified. Rather, it is sufficient for the ingress MPLS node to include a single loose node in the RSVP-TE Path message. If the tunnel should be established over the route whose maximum load is minimized, the routing algorithms we propose can be used.

We are not aware of any work that addresses the computational complexity and the potential effectiveness of N-hub Shortest-Path Routing, which is the core of this work. Several routing schemes that are similar in one way or another to ours have been leveraged in other works, e.g., \cite{5, 17, 35}, but their focus is entirely different. In \cite{63, 112}, the authors present a multi-path routing scheme called “two-phase routing.” In this scheme, traffic originating at a source node is routed over a set of routes in predetermined and static proportions. Each route is diverted from the source to an intermediate node before reaching the destination. This approach is shown to provide load balancing and bandwidth efficiency even with highly variable traffic. In \cite{62} the authors explore the deployment of this routing scheme in optical networks, in order to increase routing resiliency. In \cite{64} the authors study the throughput performance of that routing scheme. Our work\footnote{An early version of our work, published in Infocom 2004, predates Ref. \cite{62, 64, 112}} investigates the effectiveness and computational complexity of the general form of two-phase routing. Furthermore, we consider non-static routing in which intermediate nodes are determined according to current traffic conditions, while addressing the online setting of the problem.
In [91], the authors investigate the effectiveness of selfish routing in Internet-like environments. Selfish routing allows the host to determine the path according to a criterion that maximizes its profit. This work specifically addresses a setting where sources choose N-hub routes in an overlay network. Their main conclusion is that selfish hosts can achieve results similar to those achieved by routing with full control. There are two notable differences between [91] and our work. First, in our model, the host chooses routes that do not necessarily maximize its profit. Second, [91] assumes absolute knowledge of flow demands that do not change over time, while we deal with the more practical online scenario where flow demands are not known in advance.

As already said, the main benefit gained from determining more intermediate nodes (hubs) for a route between a source-destination pair is better control over network load distribution, with little deviation from the traditional shortest-path routing paradigm. More specifically, the routers continue building their routing tables using the shortest-path information they acquire through a conventional routing protocol. However, the network is capable of routing a packet over less congested areas. Moreover, it can be employed effectively even if a small fraction of the network routers support it. This is because traffic can be diverted to less congested areas without the support of the core routers.

The trade-off between the simplicity of traditional datagram (shortest-path) routing and the efficiency of virtual-circuit routing is well known. However, both schemes can be viewed as special cases of N-hub routing: with \( N = 0 \) for shortest-path routing and \( N = |V| \) for virtual circuit routing. Hence, N-hub routing, where \( 0 \leq N \leq |V| \), offers a compromise between these two extremes (see Fig. 2.2). As the number of allowed hubs grows, the number of possible routes between each source-destination pair increases, and the flexibility/efficiency of the routing scheme increases as well. However, we pay for the increased efficiency by sacrificing some of the inherent simplicity of shortest-path routing at each hub. In practice, as shown in Section 2.8, the performance achieved with a single hub is very close to the optimal performance of virtual-circuit routing. Hence, 1-hub routing can be viewed as a routing protocol that offers the performance of virtual-circuit routing with only slight deviation from traditional shortest-path routing.

### 2.3 Problem Definition and Complexity

#### 2.3.1 Problem Definition

In this chapter we focus on applying the N-hub Shortest-Path Routing paradigm to a traffic engineering task. Our specific aim is to minimize the maximum load in the network. We deal with the routing problem of minimizing the maximum load imposed on a single link by determining up to \( N \) intermediate nodes through which
Figure 2.2: N-hub routing as a compromise between efficiency and simplicity

the packets of each flow will be routed. Note that we do not assume any constraint regarding the criteria used for classifying packets to flows.

A similar objective – minimizing the maximum load imposed on a single link – was addressed in the past mainly in the context of the multicommodity flow problem [30, 66] and the Virtual Circuit Routing problem [32, 42, 75]. Maximizing the load on a single link does not always guarantee perfect load balancing and minimum average delay. However, it was shown in the past to yield good performance because the delay on a link grows exponentially with the load. Moreover, this objective is easier to analyze from a theoretical point of view. As a counter-example, consider the topology in Fig. 2.3 and suppose there are 3 flows as follows:

1. A flow from node A to node E, with a bandwidth demand of 1.
2. A flow from node A to node B, with a bandwidth demand of 2.
3. A flow from node B to node E, with a bandwidth demand of 1.

An algorithm that minimizes the maximum load may produce a solution that routes flows 1 and 3 via node C. This solution yields a greater delay of the packets of flow 1 and flow 3 than a solution obtained by an algorithm that tries to minimize the

Figure 2.3: An example of a network topology

13
average delay. The latter solution might route flow 1 through router $C$ and flow 3 through router $D$.

One may consider the average load over all the edges in the graph as a better objective for minimizing the average delay of the packets. However, this objective is achieved with static shortest-path routing which, as mentioned above, is known to be inefficient for non-uniform traffic patterns in which some areas in the AS are more congested than others. Another possible objective is minimizing the variance of the loads on the network links. However, this objective does not take into account the actual load on the links. It may therefore yield very long and possibly non-simple routes in order to ensure that all the links will be equally utilized.

In our model the network is represented by a directed graph. The routers in the network are represented by the vertices of the graph and the links by the edges. The bandwidth of a link is represented by the capacity of the corresponding edge. The source and destination of each flow are represented by their edge routers. For every flow there is a traffic demand.

We now give a formal definition of the N-hub routing problem. Let $G = (V, E)$ be a directed graph. Each edge, $e \in E$, has a capacity $u(e)$, where $u : E \rightarrow \mathbb{R}^+$. Let $\mathcal{F} \subseteq V \times V$ be a set of flows between pairs of source and destination nodes. Each flow $f \in \mathcal{F}$ has a traffic requirement $T(f)$, where $T : \mathcal{F} \rightarrow \mathbb{R}^+$. Let $s^f$ and $d^f$ denote the source and destination of flow $f$ respectively. For each flow $f \in \mathcal{F}$, find an ordered sequence of $N$ hubs, denoted by $h^f_1, h^f_2, \ldots, h^f_N$, where $h^f_i \in V$, such that the packets of $f$ are routed over $s^f \rightarrow h^f_1 \rightarrow h^f_2 \rightarrow \ldots \rightarrow h^f_N \rightarrow d^f$, where $a \rightarrow b$ denotes the shortest path from node $a$ to node $b$ on $G$, and the maximum relative load imposed on every edge in $E$ is minimized. The relative load on edge $e$ is defined as $\frac{\sum_{f|e \in P^f} T(f)}{u(e)}$, where $P^f$ is the path chosen to route flow $f$.

### 2.3.2 NP-Completeness of the N-hub routing problem

The 1-hub routing problem is a special case of N-hub routing. In what follows we formulate the 1-hub problem with uniform capacities as a decision problem and prove that this problem is NP-complete. It is easy to see that if 1-hub with uniform capacities is NP-complete, then the more general N-hub problem with arbitrary capacities is NP-complete as well. An instance for the 1-hub problem is a directed graph $G = (V, E)$, a set $\mathcal{F} \subseteq V \times V$ of flows, a function $T$ of bandwidth demand for each flow, and a positive real $K$. The question is whether there exists a hub $h^f \in V$ for each flow $f \in \mathcal{F}$ such that if the required traffic volume for $f$, namely $T(f)$, is routed over $s^f \rightarrow h^f \rightarrow d^f$, the total traffic routed through every link $e \in E$ does not exceed $K$.

**Theorem 1** 1-hub is NP-complete.
Proof

It is easy to see that 1-hub ∈ NP. To prove that 1-hub is NP-complete we will show a reduction from SAT to 1-hub. Consider the following instance for SAT. Let $U = \{u_1, u_2, \ldots, u_N\}$ be a set of variables and $C = \{c_1, c_2, \ldots, c_L\}$ a set of clauses. A valid hub assignment for the 1-hub problem is an assignment that does not impose a traffic volume greater than $K$ on any edge. We shall now transform the instance for SAT into an instance for the 1-hub problem such that a valid hub assignment for the 1-hub problem exists if and only if $C$ is satisfiable.

For every variable $u_i \in U$ of SAT, $1 \leq i \leq N$, the following three sets are defined:

\[
V_i^u = \{u_i^1, u_i^2, u_i^3, \bar{u}_i^1, \bar{u}_i^2, u_i^4\},
\]

\[
E_i^u = \{(u_i^1, u_i^2), (u_i^1, \bar{u}_i^3), (u_i^2, u_i^3), (\bar{u}_i^1, \bar{u}_i^2), (\bar{u}_i^1, u_i^4), (\bar{u}_i^2, u_i^4)\},
\]

\[
\mathcal{F}_i^u = \{(u_i^3, u_i^4)\}.
\]

For every clause $c_j \in C$ of SAT, $1 \leq j \leq L$, the following three sets are defined:

\[
V_j^c = \{c_j^1, c_j^2\},
\]

\[
E_j^c = \{(c_j^1, u_i^1), (u_i^2, c_j^2) \mid u_i \in c_j\} \cup \{(c_j^1, \bar{u}_i^1), (\bar{u}_i^2, c_j^2) \mid \bar{u}_i \in c_j\},
\]

\[
\mathcal{F}_j^c = \{(c_j^1, c_j^2)\}.
\]

An instance for the 1-hub problem is defined as follows:

\[
V = \left( \bigcup_{i=1}^{N} V_i^u \right) \cup \left( \bigcup_{j=1}^{L} V_j^c \right),
\]

\[
E = \left( \bigcup_{i=1}^{N} E_i^u \right) \cup \left( \bigcup_{j=1}^{L} E_j^c \right),
\]

\[
\mathcal{F} = \left( \bigcup_{i=1}^{N} \mathcal{F}_i^u \right) \cup \left( \bigcup_{j=1}^{L} \mathcal{F}_j^c \right),
\]

\[
T(f) = \begin{cases} 
L & \text{if } f \in \mathcal{F}_i^u 1 \leq i \leq N \\
1 & \text{if } f \in \mathcal{F}_j^c 1 \leq j \leq L 
\end{cases}
\]

\[
K = L.
\]

As an example, Fig. 2.4 shows the graph of the corresponding 1-hub instance for the following SAT instance:

\[
C = \{c_1, c_2\},
\]

where $c_1 = \{\bar{x}, y\}$ and $c_2 = \{x, \bar{y}, z\}$.

The flows of the 1-hub instance are: $\mathcal{F}_1^u = (x^s, x^d)$, $\mathcal{F}_2^u = (y^s, y^d)$, $\mathcal{F}_3^u = (z^s, z^d)$, $\mathcal{F}_1^c = (c_1^1, c_1^2)$, and $\mathcal{F}_2^c = (c_2^1, c_2^2)$. Their bandwidth demands are as follows: $T(\mathcal{F}_i^u) = 2$ for $i = 1, 2, 3$ and $T(\mathcal{F}_j^c) = 1$ for $j = 1, 2$. 

15
Figure 2.4: An instance for the 1-hub problem constructed from the instance of SAT given in (2.1)

It is easy to see that an instance for the 1-hub problem can be constructed in polynomial time.

We proceed by showing that a valid hub assignment for the 1-hub problem exists if and only if there exists a truth assignment that satisfies $C$ in the corresponding SAT problem. Let us assume that $C$ is satisfiable. Let $g : U \rightarrow \{\text{TRUE}, \text{FALSE}\}$ be a truth assignment for $C$. We now assign a hub for each $f \in \mathcal{F}$ as follows. For each $f \in \mathcal{F}_i^u$, where $f = (u_i^1, u_i^d)\ 1 \leq i \leq N$, we assign the vertex $u_i^1$ as a hub if $g(u_i) = \text{FALSE}$ and the vertex $\overline{u}_i^1$ otherwise. For each $f \in \mathcal{F}_j^v$, where $f = (c_j^1, c_j^d)\ 1 \leq j \leq L$, we assign the vertex $z_i^1$ as a hub, where $z_i \in c_j$ and $g(z_i) = \text{TRUE}$. Note that such a literal must exist since $g$ satisfies $c_j$. Clearly, the traffic volume between the pairs that correspond to the clauses is routed through the vertices corresponding to the literal whose value, as determined by $g$, is TRUE. Furthermore, the traffic volume of the pairs that correspond to the variables is routed through the vertices that correspond to the literals whose value is set to be FALSE. Hence, no edge in $G$ has a load greater than $L$.

Let $h : \mathcal{F} \rightarrow V$ be a valid hub assignment in $G$. We now show how to construct from $h$ an assignment $g$ that satisfies $C$. From the way $G$ is constructed, it follows that the traffic volume required for every flow $f \in \mathcal{F}_i^u$, $f = (u_i^1, u_i^d)\ 1 \leq i \leq N$ can be routed in two ways: either through $u_i^1$ or through $\overline{u}_i^1$. If this traffic is routed through $u_i^1$, we set $g(u_i) = \text{FALSE}$. If this traffic is routed through $\overline{u}_i^1$, we set $g(u_i) = \text{TRUE}$. Obviously, each variable in $U$ is assigned one value. From the construction of $G$, it follows that the traffic volume required for a flow $f \in \mathcal{F}_j^v$, $f = (c_j^1, c_j^d)\ 1 \leq j \leq L$ must
be routed through a vertex corresponding to a literal in $c_j$. From the construction of $T$, it follows that this literal is set by $g$ to TRUE. Otherwise, $h$ would not be a valid hub assignment. Hence, for every clause in $C$ there is a literal whose value, as determined by $g$, is TRUE. Therefore, $g$ satisfies $C$. 

\begin{corollary}
N-hub is NP-complete. Moreover, a restricted version of N-hub referred to as Integer N-hub, where all the capacities are 1 and all the bandwidth demands are integers, is also NP-complete.
\end{corollary}

\begin{proof}
From the proof of Theorem 1 it follows that Integer 1-hub is NP-complete. Integer N-hub is NP-complete since it is a generalization of Integer 1-hub. Finally, N-hub is NP-complete since it is a generalization of Integer N-hub. \hfill \square
\end{proof}

\subsection{On the Approximation Hardness of N-hub}

One common way to get around an NP-complete problem is to develop a polynomial time algorithm that finds a near-optimal solution for the problem, namely an approximation algorithm. Usually, when the worst-case performance of an approximation algorithm is bounded, the average-case performance is very close to the optimum.

Algorithm $A$ is an approximation algorithm for an optimization problem $\Pi$ if for any input $I$ it runs in polynomial time in the length of $I$ and outputs a feasible solution $A(I)$ for the problem. In the context of N-hub, a feasible solution is a solution where the route between each source-destination pair traverses at most $N$ hubs, while the route between two consecutive hubs is the shortest path between them. An algorithm $A$ for a minimization problem (like N-hub, where we seek to minimize the maximum load) is said to have an approximation ratio of $\rho$, if for any input $I$, $\text{value}(A(I))/\text{value}(\text{OPT}(I)) \leq \rho$.

An algorithm $A$ for a minimization problem is an approximation scheme for $\Pi$ if it takes as an input not only the instance $I$ of the problem, but also a value $\epsilon > 0$ such that for any fixed $\epsilon$, $\text{value}(A(I))/\text{value}(\text{OPT}(I)) \leq 1 + \epsilon$. An approximation scheme $A$ is said to be a polynomial time approximation scheme (PTAS) \cite{38} if for each fixed $\epsilon$ there is a polynomial approximation algorithm derived from $A$ with an approximation ratio of $1 + \epsilon$. If the running time of the approximation algorithm is also polynomial in the value of $1/\epsilon$, then $A$ is said to be a fully polynomial approximation scheme (FPTAS) \cite{38}.

It can be easily shown that there is no FPTAS for N-hub unless $\mathcal{P} = \mathcal{NP}$. However, in what follows we show a stronger inapproximability result.

\begin{definition}
Let $\Pi$ be a minimization problem. The decision problem $\Pi_K$ is the problem of deciding for a given instance $I$ whether the optimum value of $\Pi_K(I) \leq K$.
\end{definition}
Corollary 2 Unless $\mathcal{P}=\mathcal{NP}$, N-hub does not permit a PTAS and cannot be approximated within $2-\epsilon$ for $\epsilon > 0$.

Proof
Let $\Pi$ be an integer minimization problem. Suppose that the decision problem $\Pi_K$ is NP-hard for some constant $K$. Then, from [38] we know that unless $\mathcal{P}=\mathcal{NP}$, there is no PTAS for $\Pi$ and there is no polynomial algorithm with an approximation ratio that is strictly less than $1+1/K$. Consider the Integer N-hub problem defined earlier. Obviously, in a feasible solution of this problem the maximum load has an integer value equal to 1 or more. However, by Corollary 1, the problem of deciding whether the optimum value of Integer N-hub is equal to 1 is also NP-hard. Hence, Integer N-hub, and subsequently N-hub, does not permit a PTAS and cannot be approximated within $2-\epsilon$ for any $\epsilon > 0$.

2.4 A Probabilistic Approximation Algorithm for the Offline N-hub Problem

In section 2.3 we saw that N-hub is NP-complete and difficult to approximate. In this section we describe a probabilistic approximation algorithm for 1-hub. This algorithm is based upon an approximation technique presented in [94]. This algorithm is then extended for the more general N-hub problem. The algorithm is the basis of an Asymptotic PTAS for N-hub. We begin by describing the general technique we use in the algorithm and then present the algorithm itself.

2.5 The Approximation Technique

The idea is to formulate 1-hub as an integer program, and then to transform it into a rational linear program that can be solved in polynomial time [59] by relaxing the integrality constraints of its variables. After the relaxed program is solved, the values of the relaxed variables are rounded either to 0 or to 1 in a randomized manner. Thus, with a certain probability, the value of the objective function, namely the maximum load in the network, is “close” to the optimum of the linear relaxation. It is therefore “close” to the optimum of the original integer programming problem. This concept was introduced in [94]. It is effective for problems whose objective function is an upper bound of sums of the problem’s binary variables.

Let $\Pi_I$ be an integer linear program and $\Pi_R$ be its rational relaxation. Let the variables of the problem be $x_1, x_2, \ldots, x_n$. Note that in $\Pi_I$, $x_i \in \{0,1\}$ whereas in $\Pi_R$, $x_i \in [0,1]$. The basic algorithm, as presented in [94], consists of the following two phases:
1. Solve $\Pi_R$. Let the value assigned to every variable $x_i$ be $\alpha_i$, where $\alpha_i \in [0, 1]$.

2. Set every variable $x_i$ to 1 or 0 randomly, such that $\text{Prob}(x_i = 1) = \alpha_i$.

In some problems the constraints dictate that the variables should be partitioned into several sets, and the sum of the variables of each set must be 1. In these problems the variables in each set are still randomly rounded to 1, but in a mutually exclusive manner.

As mentioned above, this technique is suitable for problems whose objective function is an upper bound of the sums of its binary variables. Therefore, in order to approximate the objective function, an upper bound for these sums should be found. It was observed in [94] that the sum of the rounded variables is actually a sum of independent Bernoulli random variables, where each variable may be associated with a different probability. In order to find an upper bound for sums of this kind, [94] uses results from [45] and [20]. From these results the following is derived:

$$\text{Prob}(\Psi \geq m) < e^{-\beta^2 N p}$$

where $\Psi$ is the sum of the independent Bernoulli variables, $N$ is the number of the variables and $m = (1 + \beta)Np$, where $0 < \beta \leq 1$ and $p = \sum_{k=1}^{N} p_k$ ($p_k$ is the success probability for the $k$th Bernoulli variable).

This upper bound is applicable only for Bernoulli random variables and not for other random variables with a more general distribution. In the problem we consider the objective function is not necessarily an upper bound of sums of Bernoulli variables; it is actually an upper bound for sums of flow demands passing through the links. Hence, in the following we use a different probabilistic analysis.

### 2.6 The Approximation Algorithm

We now apply the approximation technique presented above to the 1-hub problem. We start by formulating 1-hub as an integer programming problem. For every flow $f$, and for every node $i$ that can serve as a hub for the traffic of this flow, the following binary variable is defined:

$x_{if} - A$ binary variable whose value is 1 if node $i$ is assigned as a hub for the traffic of flow $f$ and 0 otherwise.

Parameters:

- $T_f$ - For every flow $f$, $T_f$ indicates the traffic volume demanded by $f$.
- $u_e$ - For every edge $e$, $u_e$ indicates the capacity offered by $e$.
- $z_{ij}^e$ - For every flow $f = (s, d)$, node $i$ and link $e$, let $z_{ij}^e = 1$ if $e$ is on the shortest path from $s$ to $i$ or from $i$ to $d$ and 0 otherwise.
The target function, Minimize $L$, is subject to the following constraints:

(a) $\forall f \quad \sum_{i} x_{if} = 1$
(b) $\forall e \quad \sum_{i,f} \frac{x_{if} \cdot z_{if} \cdot T_{f}}{u_{e}} \leq L$
(c) $\forall i, f \quad x_{if} \in \{0, 1\}$.

Constraint (a) ensures that exactly one node serves as a hub for $f$. Constraint (b) ensures that no edge will carry a relative traffic load greater than $L$. Constraint (c) ensures that the traffic of each flow is not split (i.e., it is routed on a single route).

The linear relaxation of the above program allows each variable $x_{if}$ to be assigned any real value in $[0, 1]$. This implies that we actually relax the requirement that for every flow there must be exactly one route that carries $T_f$ (constraint (c)).

After obtaining an optimal solution for the relaxed linear program, we have for every flow $f = (s, d)$ a set of hubs $\Gamma_f$ through which $T_f$ is routed. Each hub $h \in \Gamma_f$ defines a route from $s$ to $d$, that consists of the shortest paths from $s$ to $h$ and from $h$ to $d$. Each such route carries a fraction of the traffic volume, $T_f$. Each hub $h \in \Gamma_f$ is associated with a weight equals to that fraction of $T_f$. The sum of the weights for every $\Gamma_f$ is, of course, 1.

The next step is to convert the solution of the relaxed linear program into a solution of the original integer program by rounding the weight of one selected hub in every $\Gamma_f$ to 1 and rounding the weights of the other hubs to 0. In other words, the entire traffic volume of $f$ will be routed through the route defined by the selected hub from $\Gamma_f$. The hub is selected randomly, with a probability that is equal to its weight. Note that these random choices are made independently for each flow $f$. The following theorem shows that the presented approximation algorithm has an absolute performance factor of $O(\log(|E|))$. Namely, $|A(I) - OPT(I)| \leq O(\log(|E|))$.

**Theorem 2** Let $\varepsilon$ be a positive real such that $0 < \varepsilon < 1$. Let $L_{opt}$ be the optimum value of $L$ obtained by the relaxed linear program. After a single hub is chosen for every $f$ using the approximation algorithm, there is a probability greater than $1 - \varepsilon$ that the load on each edge is upper bounded by:

$$L_{opt} + \left[ \frac{n \ln(|E|/\varepsilon)}{2} \right]^{1/2} \frac{T_{max}}{u_{min}},$$

where $n$ is the number of flows. $E$ is the set of links in the network, $T_{max}$ is the maximum bandwidth demand of a flow, and $u_{min}$ is the minimum capacity of an edge.

**Proof**

Consider an edge $e \in E$. Let $L_e$ be the relative load imposed on $e$ in the optimal solution as determined by the linear program. The load imposed on $e$ by the approximation algorithm is a sum of $n$ independent random variables, $X_{ef}$ for $1 \leq f \leq n$. 

20
The value of $X_{ef}$ indicates the contribution of the traffic generated by $f$ to the load imposed on $e$. Hence, the distribution of $X_{ef}$ is as follows:

$$X_{ef} = \begin{cases} 
0 & \text{with probability } 1 - p_{ef} \\
\frac{T_f}{u_e} & \text{with probability } p_{ef},
\end{cases}$$

where $p_{ef}$ is the fraction of flow $f$ routed over $e$ according to the solution found by the linear program. Recall that the linear program is likely to split $T_f$ between multiple routes. Some of these routes (or none of them) might include link $e$. Hence, $p_{ef}$ is equal to the aggregated traffic of flow $f$ carried by these routes. Namely, $p_{ef} = \sum_i x_{ef} \cdot z_{ef}$.

We know the following from [46]. Let $X_1, X_2, \ldots, X_n$ be independent random variables, where $0 \leq X_i \leq 1$. Let $\mu = E(S)$, where $S = X_1 + X_2 + \ldots + X_n$. Then, for every $t$, $0 < t < 1 - \mu/n$, the following holds:

$$\text{Prob}(S \geq nt + \mu) \leq e^{-2nt^2}.$$

Let $S_e$ be a random variable such that $S_e = \sum_f X_{ef}$. Note that $E(S_e) = L_e$. In order to use the upper bound of [46], the random variables $X_{ef}$ should take values in the range $[0,1]$. We therefore multiply $X_{ef}$ by $u_{min}/T_{max}$ for every $f$. Now, the load imposed by the algorithm on link $e$ is a sum of random variables, $\hat{X}_{ef}$, of the following type:

$$\hat{X}_{ef} = \begin{cases} 
0 & \text{with probability } 1 - p_{ef} \\
\frac{T_f}{u_e} \cdot \frac{u_{min}}{T_{max}} & \text{with probability } p_{ef}.
\end{cases}$$

Let us denote this sum by $\hat{S}_e$. Note that $\hat{S}_e = S_e \frac{u_{min}}{T_{max}}$. Therefore, $E(\hat{S}_e) = L_e \frac{u_{min}}{T_{max}}$.

This value does not exceed, of course, $L_{opt} \frac{u_{min}}{T_{max}}$.

Applying the upper bound of [46] mentioned above yields:

$$\text{Prob} \left( \hat{S}_e \geq nt + L_e \frac{u_{min}}{T_{max}} \right) \leq e^{-2nt^2}$$

for $0 < t < 1 - \frac{L_e u_{min}}{nt_{max}}$. Choosing

$$t = \left[ \frac{\ln(|E|/\epsilon)}{2n} \right]^{1/2}$$

where $\epsilon$ is a positive real smaller than 1, yields

$$\text{Prob} \left( \hat{S}_e \geq nt + L_e \frac{u_{min}}{T_{max}} \right) \leq \frac{\epsilon}{|E|}. \quad (2.2)$$
Let $\hat{S} = \text{MAX}_{e \in E}\{ \hat{S}_e \}$. Hence, $\hat{S}$ is the normalized maximal load imposed on any link according to the solution obtained by the approximation algorithm. Note that $L_{opt} \geq L_e$. From Eq. (2.2) we get:

$$\text{Prob} \left( \hat{S} < nt + L_{opt} \frac{u_{min}}{T_{max}} \right) > 1 - \epsilon. \quad (2.3)$$

We now return to the original problem with the original bandwidth demands. Let $S = \text{MAX}_{e \in E}\{ S_e \}$. Hence, $S$ is the non-normalized maximal load imposed on any link according to the solution obtained by the approximation algorithm. From (2.3) we get:

$$\text{Prob} \left( S < nt \frac{T_{max}}{u_{min}} + L_{opt} \right) = \text{Prob} \left( \hat{S} < nt + L_{opt} \frac{u_{min}}{T_{max}} \right) > 1 - \epsilon, \quad (2.4)$$

which concludes the proof.

The presented approximation algorithm and Theorem 2 are also applicable to the more general N-hub with the obvious modifications. We consider an ordered N-tuple of nodes as a supernode. Instead of assigning to every flow a single node as its hub, we assign a supernode. There are $|V|^N$ supernodes in $G = (V, E)$. The formulation of N-hub as an integer programming problem is the same as 1-hub, except that $x_{if}$ equals 1 if supernode $i$ is assigned as a “hub” to flow $f$ and that $z_{ef}$ equals to 1 if $e$ is on the route defined by the end nodes of $f$ and supernode $i$. The rest of the analysis is similar to the 1-hub analysis.

In some cases it would be desirable to guarantee with a high probability that the solution of the approximation algorithm will not exceed the optimal solution by a certain factor. Let us consider the case where this factor is 2. From Eq. 2.4 it follows that in order to ensure that the solution will not exceed the optimal solution by this factor, $nt \frac{T_{max}}{u_{min}} \leq L_{opt}$ must hold. This yields the following constraint on $L_{opt}$:

$$\left[ \frac{n \ln(|E|)}{2} / \epsilon \right]^{1/2} \frac{T_{max}}{u_{min}} \leq L_{opt}.$$ 

Note that this does not impose a rigid upper bound but rather a probabilistic one. Furthermore, it should be noted that the approximation ratio of the above algorithm will be as small as we want it to be, provided that we increase the maximum load in the network. It represents, therefore, an asymptotic PTAS [38].

### 2.7 Online Approximation Algorithms

We now consider the more practical online version of N-hub, where routing decisions for the flows are performed one at a time without prior knowledge of future flows.
We consider three online approximation algorithms, originally developed for the unsplittable multicommodity flow problem [38]. We slightly modify these algorithms in order to apply them to the N-hub Shortest-Path routing problem. We prove that their competitive ratios for the unsplittable multicommodity flow problem is the same as for the N-hub Shortest-Path routing problem. The competitive ratio of an online algorithm is defined as the worst case ratio, over all sequences of flows, between the value of the solution found by the algorithm and the value of the solution found by an optimal offline algorithm. See [55] for further details.

For the sake of completeness we give a formal definition of the unsplittable multicommodity flow problem. Let $G = (V, E)$ be a directed graph. Each edge, $e \in E$, has a capacity of $u(e)$, where $u : E \to \mathbb{R}^+$. Let $F \subseteq V \times V$ be an ordered set of flows between pairs of source and destination nodes. Each flow $f \in F$ has a traffic requirement $T(f)$, where $T : F \to \mathbb{R}^+$. Route every flow $f \in F$, in the order the flows are received, on a single arbitrary route in $G$, while minimizing the maximum relative load imposed on every edge. This problem, also known as Routing of Permanent Virtual Circuits, is NP-complete. The splittable version of this problem, which allows the traffic of each flow to be split over multiple routes, is known to be in $P$.

The only difference between the N-hub problem and the unsplittable multicommodity flow problem is that in the former the set of possible routes for each source-destination pair is restricted while in the latter it is not. Hence, the unsplittable multicommodity flow problem can be viewed as a $|V|$-hub routing problem.

**Corollary 3** The best competitive ratio that can be achieved by an online algorithm for N-hub has a lower bound of $\Omega(\log |V|)$.

**Proof**
In [32] this lower bound is proven for the unsplittable multicommodity flow problem. In this proof a specific network and a specific sequence of flows are considered. For this specific instance, the maximum load imposed on an edge by an offline algorithm is 1, whereas the maximum load imposed by an online algorithm is at least $\frac{\log |V|}{2}$. Since all the routes in the considered network have a length of at most three edges, each of them can be represented as a 1-hub route. Hence, this proof is also valid for the 1-hub problem, and for the general N-hub problem, as well. $\square$

However, if we consider a more practical variant of the online version, where termination of flows is permitted, i.e., the lifetime of each flow is finite, we can show that no routing algorithm can do better or worse than a competitive ratio of $\Theta(|E|)$.

**Theorem 3** For the online version of N-hub, when flow termination is allowed, the competitive ratio that can be achieved by any algorithm is $\Theta(|E|)$.

**Proof**
Let us consider a directed graph that has a single source $s$, connected to a single
target $t$ via $n$ directed edges, each with capacity $C$. We construct a sequence of $n^2$ flow requests, each with a traffic demand $T$. After all the $n^2$ flows are routed, the maximum load in the network is $m^2T$ where $m \geq n$. Let $e$ be the edge with the maximum load. We now terminate all the flows that do not pass through $e$ and some $(m - n)$ flows that do pass through $e$. The maximum load in the network now is $n^2T$. The optimal offline algorithm in this situation can maintain a maximum load of $\frac{T}{n}$ by routing each of the $n$ remaining flows on a separate edge. Hence, the best competitive ratio a routing algorithm can achieve is at least $\Omega(\sqrt{n})$.

We now show that the worst competitive ratio any routing algorithm can achieve is $O(|E|)$. Consider a graph with $|E|$ edges, each with capacity $C$, and a sequence of $n$ flow requests with traffic demand $T$. The maximum load that can be produced by an online algorithm in the worst case is $\frac{nT}{C}$. The maximum load that can be produced by the optimal offline algorithm is at least $\frac{n^2T}{|E|C}$. Hence, the worst competitive ratio that can be obtained is $O(|E|)$. Using Theorem 4, this result can be extended to networks with non-uniform edge capacities. \hfill \Box

From Theorem 3 it follows that not much can be done if we want to guarantee some competitive ratio when flow termination is considered. However, from Corollary 3 it follows that when flow termination is not considered, it is possible to meet the challenge of designing an algorithm that has a competitive ratio of $O(\log |V|)$. In what follows we present some online algorithms for the problem.

These algorithms are similar in structure, as follows. Let $f$ be a new flow to be routed. Let $T_f$ be the bandwidth demand of $f$. Let $L_e$ and $U_e$ be the current load and capacity of link $e \in E$, respectively. From all feasible $N$-hub routes, the algorithm chooses the one that satisfies a given criterion as follows:

- **Algorithm-1:** minimize
  \[
  \sum_{e \in P} a \left( \frac{L_e + T_f / U_e}{C} - \frac{L_e}{C} \right),
  \]
  where $a \in (1, 2)$ and $C$ is as explained below

- **Algorithm-2:** minimize
  \[
  \text{MAX}_{e \in E} \left\{ \begin{array}{ll}
  L_e & e \notin P \\
  L_e + \frac{T_f}{U_e} & e \in P
  \end{array} \right.
  \]

- **Algorithm-3:** minimize
  \[
  \text{MAX}_{e \in P} L_e + \frac{T_f}{U_e}.
  \]

In all cases, $P$ denotes a possible path for the considered flow.

These algorithms were presented in [32] (Algorithm-1) and in [42] (Algorithm-2 and Algorithm-3) for the unsplittable multicommodity flow problem, and are applied
in this work for the N-hub routing problem. We next show that the competitive ratios of the above algorithms for the N-hub problem are the same as for the unsplittable multicommodity flow problem, and that Algorithm-1 is the best online algorithm for the N-hub problem.

In Algorithm-1, $\Lambda$ is an estimate for the value of the optimal solution. A simple doubling technique is used to estimate its value. The algorithm starts with some initial estimate. If, during execution, the maximum load exceeds $\Lambda$ by $\log(|V|)$, the estimate is doubled and the algorithm is invoked. The algorithm assigns to each edge a weight that increases exponentially in the load that will be imposed if this edge is part of the route selected for the considered flow. The algorithm chooses from all possible routes for the considered flow the one with the minimum weight. A route’s weight is the sum of the weights of all its edges. The intuition behind the exponential function weight is that as the load on an edge increases, the weight of the edge increases exponentially. Consequently, the algorithm prefers a long non-congested route over an exponentially shorter, but congested, route. The algorithm achieves a competitive ratio of $O(\log |V|)$ for the unsplittable multicommodity flow problem. To prove this, [32] uses the following auxiliary potential function:

$$\Phi(j) = \sum_{e \in E} a^{\frac{L_e(j)}{\Lambda}} (\gamma - L_e^*(j)/\Lambda), \quad (2.5)$$

where $L_e(j)$ and $L_e^*(j)$ are the load imposed on edge $e$ by Algorithm-1 and by an optimal offline algorithm, respectively, after the first $j$ flows are routed, and $a = 1 + 1/\gamma$. Function $\Phi(j)$ is non-increasing in $j$ since the weight of the route chosen by the algorithm for every flow is not greater than the weight of the route chosen by an optimal offline algorithm. Since $\Phi(0) \leq \gamma|E|$ and $L_e^*(j)/\Lambda \leq 1$, $\gamma|E| \geq \sum_{e \in E} (\gamma - 1)a^{\frac{L_e(j)}{\Lambda}}$ holds, and the competitive ratio follows. For the N-hub problem, the weight of the route chosen by Algorithm-1 is still not greater than the weight of the route chosen by an optimal offline algorithm. This implies that the potential function in Eq. 2.5 is also non-increasing in $j$. Hence, the competitive ratio of $O(\log(|V|))$ holds for N-hub as well.

Algorithm-2 uses a simple greedy approach. It chooses a route such that the maximum load imposed on any edge is minimized after the flow is routed. When all edge capacities are equal, this algorithm has a competitive ratio of $O(\sqrt{|D|E})$, where $D$ is the maximum ratio, over all flows, between the length of the longest and shortest routes that can be assigned to the flow. We now show that this competitive ratio is also valid for N-hub (when all the edges have equal capacities). In [42], where this competitive ratio is proven for the unsplittable multicommodity flow problem, the values of the loads are divided into levels. The load $L_e$ on edge $e$ is said to be in the $i$'th level if $i \cdot T_{\text{max}}/w \leq L_e \leq (i+1) \cdot T_{\text{max}}/w$, where $T_{\text{max}}$ is the maximum bandwidth requirement and $w$ is the capacity of the edges. The level of route $P$ is the maximum level over all the edges in $P$. The crux of the proof is that when the
maximum load in the network moves up to level \( i \), then all the edges in the network, including the edges of the route chosen by the optimal offline algorithm, are at least in level \( i-1 \). Since this claim is also valid for N-hub, the competitive ratio is valid for N-hub as well. Theorem 4, which will be presented later in this section, shows how to adapt this competitive ratio to the general case where the edge capacities are not necessarily equal.

Algorithm-3 always chooses the route with the minimum load. The load of a route is defined as the maximum load over all the route’s edges. The basic idea is to make the route selection criterion stricter than in Algorithm-2. To understand the difference between the two criteria, consider a network with two nodes connected by three edges with equal capacities. Suppose that the loads imposed on these edges by existing flows are 1, 4 and 6. Suppose also that the next flow to be routed has a bandwidth demand of 2. Algorithm-2 may route this flow either on the first edge or on the second edge, because in both cases the maximum load remains 6. In contrast, Algorithm-3 chooses the first edge because it is the least loaded. This implies that every route chosen by Algorithm-3 is also a valid choice for Algorithm-2, but not vice versa. In order to increase the attractiveness of Algorithm-2 over that of Algorithm-3, we have modified it in the following way. When Algorithm-2 finds several routes that do not increase the maximum load imposed on any edge, it does not choose one arbitrarily, as proposed in [42], but chooses the shortest one.

When Algorithm-3 is employed in networks with equal capacities, it has a competitive ratio of \( O(d \log |V|) \), where \( d \) is the longest route that can be assigned to a flow. For reasons similar to those stated earlier for Algorithm-2, the same competitive ratio is guaranteed when Algorithm-3 is used for N-hub. Once again, we can use Theorem 4 to extend this competitive ratio to the case where edge capacities are not necessarily equal.

**Theorem 4**  Let \( A(I) \) be an online algorithm for N-hub that achieves a competitive ratio of \( C \) in networks whose edges have the same capacity. Then, \( A(I) \) achieves a competitive ratio of \( \frac{u_{\text{max}}}{u_{\text{min}}} \cdot C \) in networks whose minimum edge capacity and maximum edge capacity are \( u_{\text{min}} \) and \( u_{\text{max}} \) respectively.

**Proof**

Let \( G = (V, E) \) represent a network, and let \( u : E \rightarrow \mathbb{R}^+ \) be the edge capacity function. Let \( OPT \) be the value of an optimal offline solution. Let \( G' \) represent another network with the same structure but with a different edge capacity function \( u' \), such that for every edge \( e \) \( u'(e) = u(e)/\alpha \). Let \( OPT' \) be the value of an optimal offline solution for \( G' \). We first prove that

\[
OPT' = \alpha OPT . \tag{2.6}
\]

Assume that \( OPT'/\alpha < OPT \). Let \( S' \) be the solution corresponding to \( OPT' \). Since the capacity of each edge in \( G \) is \( \alpha \) times larger than the corresponding edge in \( G' \),
applying the solution $S'$ to the original graph $G$ would yield a maximum load of $OPT'/\alpha$. This maximum load is strictly lower than $OPT$, in contradiction to our assumption. A similar contradiction applies when $OPT'/\alpha > OPT$.

Let $G_{min}$ be a graph similar to $G$ whose edge capacities are equal to $u_{min}$. Let $A_{min}$ and $OPT_{min}$ be the values of the solutions found by the online algorithm and the optimal offline algorithm, respectively, for $G_{min}$. Since the edge capacities do not increase, $A \leq A_{min}$, where $A$ is the value of a solution found by the online algorithm for $G$. Since the capacity of each edge in $G_{min}$ is divided by a factor that is not greater than $\frac{u_{max}}{u_{min}}$, by Eq. 2.6 we get that $OPT_{min} \leq \frac{u_{max}}{u_{min}} OPT$. Since $A_{min} \leq C \cdot OPT_{min}$ holds, we conclude that $A \leq C \cdot \frac{u_{max}}{u_{min}} OPT$. □

We now discuss the time complexity of the three algorithms. Each algorithm has to review the entire set of N-hub routes before choosing one. There are $O(|V|^N)$ such routes. In a naive implementation, the algorithm metric is calculated independently for each route. Assuming that the maximum length of the shortest path between two nodes in the graph is $D$, the longest N-hub route is $D(N+1)$. Hence, each algorithm has to make $O(|V|^N D(N+1))$ metric calculations. When $N = 1$, and $D = O(|V|)$, the time complexity is $O(|V|^2)$. A faster approach for algorithms 1 and 3 is to precalculate the total metric for every possible shortest path between the graph nodes, using Dijkstra's algorithm, for example. In this case, the time complexity is $O(|V|^N (N+1)+|V|^3)$, which is smaller than the former time complexity for $|N| > 2$. The dominant operations in the metric calculation of Algorithm 1 are division and exponential computations for real values. In contrast, algorithms 2 and 3 require only division operations. Hence, Algorithm 1 has a higher time complexity and a longer expected running time.

### 2.8 Simulation Study

In this section we present simulation results for the routing algorithms discussed in the previous section. We generated router-level networks with random capacity edges, using Waxman's model [110] and the BRITE simulator [77]. We randomly chose a sequence of source-destination nodes. Each pair represents a flow to be routed in the network. The sequence of flows was generated using Zipf. A random network topology and a random sequence of flows form one instance of the N-hub routing problem. Using an event-driven simulator, we find for each instance the maximum load in the network under the following schemes:

1. The standard shortest-path routing scheme (SP) used today in IP. This is also known as minimum hop routing.

2. The hypothetical optimal routing (OPT) scheme. In this scheme we find a solution for the splittable multicommodity flow problem presented in Section
2.7. Recall that this version of the problem is in $\mathcal{P}$. An algorithm for OPT that is based on linear programming is presented in Appendix 2.10. This scheme allows the traffic of a flow to be split over multiple routes. OPT's performance is a theoretical lower bound for N-hub, and therein lies its importance.

3. Algorithm-1, Algorithm-2 and Algorithm-3, as presented in Section 2.7.

To solve the linear programs for OPT, we used the Lp_Solve software [12].

Throughout the simulation study, we assigned a random demand with a fixed average to each flow. Hence, there is a strong correlation between the number of flows the routing protocol has to handle and the load imposed on the network. We therefore use the number of flows as our offered load metric.

Figure 2.5 depicts simulation results of the routing schemes OPT, SP, and the first online algorithm (Algorithm-1) presented in Section 2.7. These simulations were carried out in a medium size backbone network (50 routers). Algorithm-1 is implemented with $N = 1$. The most important finding in these graphs, and probably in the research so far, is that the performance of 1-hub is very close to that of OPT, and the improvement over SP is significant. Algorithm-1 reduces the maximum load in the network by up to 73%. We also simulated Algorithm-2 and Algorithm-3 with $N = 1$. However, the performance of these algorithms is slightly lower than that of Algorithm-1. The inferior performances of Algorithm-2 and Algorithm-3 can be attributed to the fact that they do not take into account the length of the chosen routes. Longer routes impose, of course, greater load on the network.

We now compare the performance of the various algorithms in networks with different topologies. Fig. 2.5(a) shows simulation results for backbone networks with low link density ($|E|/|V| = 2$), whereas Fig. 2.5(b) shows the results for backbone
networks with higher link density ($|E|/|V| = 5$). Note that as the link density increases, the number of routes between two nodes also increases. As expected, the maximum load produced by all the routing schemes decreases as the link density increases. However, while the maximum load produced by the shortest-path routing decreases on the average by only 25%, the maximum loads produced by the optimal routing scheme and by Algorithm-1 for 1-hub decrease by 65%. Since the shortest-path routing scheme uses only one path for a source-destination pair, the increase in the number of routes between two nodes is insignificant. In contrast, the optimal routing scheme and the 1-hub based routing algorithms can route different flows of a source-destination pair over different routes, in response to traffic conditions. Note that the ability to use various routes for a single source-destination pair is especially important for networks with hot-spots.

Figure 2.6 depicts simulation results for a small routing domain having $|V| = 10$ and $|E| = 20$. This routing domain has the same link density as the routing domain corresponding to Figure 2.5(a). This allows us to compare the performance of N-hub routing in routing domains with different numbers of routers. Note first that the increase in the number of routers has only a negligible effect on the difference in performance between the 1-hub and the optimal routing schemes. This is despite the fact that the number of unrestricted routes between two end nodes increases exponentially with the number of routers, whereas the number of 1-hub routes increases only linearly. One might expect the maximum loads produced by the various routing schemes to be higher in small routing domains than in larger ones, and the relative difference in the performances of the N-hub routing scheme and shortest-path routing to decrease. Interestingly, however, the maximum loads produced are actually smaller than in Figure 2.5(a) and the relative difference between 1-hub and shortest-path is similar to that of Figure 2.5(a). This is attributed to the fact that the average number of links a flow has to traverse decreases for smaller routing domains. Hence, each flow consumes fewer network resources, thereby reducing the maximum loads produced by the various routing schemes.

We now examine the performance of the N-hub routing scheme with different values for $N$, i.e., with different numbers of possible hubs. Figure 2.7 depicts simulation results of Algorithm-1 for the 1-hub, 2-hub and 3-hub schemes for a routing domain having $|V| = 50$ and $|E| = 250$. The most important finding is that the differences in performance for different values of $N$ are negligible (less than 1%). We therefore use a single curve for $N = 1$, $N = 2$ and $N = 3$. This result is attributed to the flexibility of 1-hub routing. Adding flexibility by allowing more hubs to the routing process does not contribute to its effectiveness. However, for much larger routing domains, we expect a visible performance difference because the flexibility of 2-hub and 3-hub routing schemes increases polynomially (by powers of 2 and 3 respectively) with the number of routers, whereas the flexibility of a 1-hub scheme increases only linearly.
Figure 2.6: Performance of the Algorithm-1 for 1-hub for small network (|V| = 10, |E| = 20)

We wanted to investigate not only the specific algorithms proposed in the chapter, but also the pure concept of N-hub routing. To this end, we tested the performance of 1-hub routing with a random algorithm (RAND). This algorithm does not take into account the aggregated load imposed on every network link or the load imposed by every connection. Rather, it selects a random hub for every new connection. We compared the performance of RAND with the performance of Shortest-Path (SP) and the performance of Algorithm-1 for different routing domain sizes. Figure 2.8(a) depicts simulation results for a small routing domain with 10 routers and 30 links. It is evident that RAND performs poorly in such domains, because the maximum load it imposes is even higher than the maximum load imposed by SP. This is attributed to the fact that like SP, RAND selects the routes without knowing the distribution of link loads. Since the routes selected by RAND are longer than those selected by SP, the bandwidth consumed by RAND is higher and the maximum load imposed in the routing domain increases. However, this is not the case for a large routing domain. Figure 2.8(b) depicts the performance of the various routing schemes for a domain with 200 nodes and 4000 links. The maximum load imposed by RAND is about 40% lower than the maximum load imposed by SP. This reduction is attributed to the fact that RAND is a symmetry-breaking procedure which better balances an offered load created with a Zipf distribution. This load reduction is not possible in a small routing domain in which the load is approximately uniform and there is no advantage in routing through distant hubs.
Figure 2.7: Performance of Algorithm-1 for 1-hub, 2-hub and 3-hub ($|V| = 25$ and $|E| = 75$)

To validate our findings regarding the effectiveness of N-hub routing, we have also used for our simulations an actual ISP topology as mapped by the RocketFuel project [99]. The bandwidth for each link is determined according to [74]. Figure 2.9 depicts simulation results for the Exodus ISP from [99]. It is evident that the results are similar to those achieved using the Waxman model. Again, Alg-1 using 1-hub routing performs very close to the theoretical optimum and achieves 80% improvement over the results achieved by shortest-path routing. We found similar results when implementing other ISP topologies from [99].

We conclude this section by looking at the problem from a different angle. Figure 2.10 depicts the maximum number of flows the network can accommodate under each routing algorithm as a function of the maximum load that can be imposed on a single link. Instead of routing all the flows and finding the maximum load, we now determine the maximum number of flows that can be routed, subject to a maximum load constraint. A flow is rejected if routing it over the chosen route causes the maximum load in the network to exceed the maximum tolerated load. The simulation stops when the network is saturated. The network is assumed to be saturated when 100 consecutive flows are rejected. Fig. 2.10 depicts simulation results for networks with $|V| = 50$ and $|E| = 250$. We can see that the 1-hub version of Algorithm-1 achieves the best results: it can accommodate on the average 51% more flows than SP. Algorithm-3 achieves 48% improvement over SP, and Algorithm-2 achieves only 34% improvement. These results suggest that although the three routing algorithms
produce similar maximum loads, as shown in the previous simulation, the difference in the quality of their routing is distinct. The higher number of flows accepted by Algorithm-1 and Algorithm-3 indicates their ability to better balance the load in the network, thereby achieving a higher throughput.

\begin{figure}[h]
\centering
\includegraphics[width=0.4\textwidth]{figure1.png}
\caption{Performance of a random algorithm for 1-hub for different network sizes}
\end{figure}

2.9 Conclusions

In this chapter we studied the effectiveness of the N-hub Shortest-Path Routing concept in IP networks. We have demonstrated that this concept offers an excellent compromise between the simplicity of shortest-path routing and the efficiency of virtual circuit routing. We applied this concept to the problem of minimizing the maximum load in the network. We defined the corresponding optimization problem, and proved that it is NP-Complete even for $N = 1$. We also showed that it does not permit a PTAS and cannot be approximated within $2 - \epsilon$ for $\epsilon > 0$. However, we present in Section 2.4 a probabilistic asymptotic PTAS for the offline version of N-hub.

We have addressed the online version of N-hub, where the set of the input flows is not known in advance. We showed that the best competitive ratio an online N-hub algorithm may achieve is $\Omega(\log |V|)$. We then presented an online algorithm that achieves this lower bound, and two additional online algorithms that have less attractive competitive ratios, but are also less computationally intensive.

We then used simulations to study the practical effectiveness of N-hub routing in general, and of the specific algorithms presented in the chapter. Our main findings are as follows:

- The performance of N-hub Shortest-Path Routing is very close to the performance of a hypothetical optimal algorithm that splits the traffic of the same
flow among multiple routes.

- The N-hub Shortest-Path Routing scheme can produce much better quality routing than shortest-path routing, without the need to incorporate complicated logic into the routing process or even make the effort to learn the link load distribution throughout the routing domain.

- The effect of $N$ on the performance of $N$-hub is very small. Hence, even the performance of 1-hub is very close to optimal.

- Although the competitive ratio of an online algorithm is $\Omega(\log |V|)$, all three online algorithms proposed in this chapter perform very well in practice.

We therefore conclude that N-hub Shortest-Path Routing, and in particular the $N = 1$ version, should be considered as a powerful mechanism for future datagram routing in the Internet.

2.10 Appendix: A linear program for the general routing problem

We describe a general routing problem, expressed in the form of a linear program, for the optimal routing scheme discussed in Section 2.8. Let $G = (V, E)$ be a directed
Figure 2.10: No. of connections vs. max. tolerated load

graph representing the network. Let $\mathcal{F}$ be a set of flows in the network. Let $T_f$ denote the bandwidth demand of flow $f$, and let $s^f$ and $d^f$ denote the source and destination of flow $f$ respectively. For every flow $f$ and link $e$, let $l_e^f$ represent the traffic load imposed on link $e$ due to flow $f$.

The linear program is as follows:

\[
\text{Minimize} \quad L \\
\text{subject to the following constraints:} \\
\begin{align*}
\text{(a)} \quad & \sum_{e \in E_v^{in}} l_e^f - \sum_{e \in E_v^{out}} l_e^f = \left\{ \begin{array}{ll}
T_f & \text{if } v = d^f \\
-T_f & \text{if } v = s^f \\
0 & \text{otherwise}
\end{array} \right. \\
\forall v \in V \text{ and } \forall f \in \mathcal{F}
\end{align*}
\]

where $E_v^{in}$ and $E_v^{out}$ are the sets of incoming and outgoing links of vertex $v$ respectively, and

\[
\begin{align*}
\text{(b)} \quad & \sum_{f \in \mathcal{F}} l_e^f \leq L \quad \forall e \in E.
\end{align*}
\]

The first constraint ensures that the traffic flow is conserved in each vertex and it is routed from its source to its destination. The second constraint ensures that the load on each link does not exceed $L$. 

34
Chapter 3

A Traffic Engineering Approach for Placement and Selection of Network Services

In this chapter we study how the concept of “N-hub Shortest-Path Routing” is applied to the engineering of network services’ traffic. Network services are provided by means of dedicated service gateways, through which traffic flows are directed. Existing work on service gateway placement has been primarily focused on minimizing the length of the routes through these gateways. Only limited attention has been paid to the effect these routes have on overall network performance. In this chapter we propose a novel approach for the service placement problem, which takes into account traffic engineering considerations. We divide the problem into two sub-problems: finding the best location for each service gateway, and selecting the best service gateway for each flow. We propose efficient algorithms for both problems and study their performance. The main contribution of this chapter is showing that placement and selection of network services can be used as effective tools for traffic engineering.

The work described in this chapter was originally published in [25].

3.1 Background

As the Internet becomes more prevalent and diverse, there is a growing demand for services that facilitate and enhance interoperability, performance and security of communication between two or more parties. Examples for such services are voice and video conversion, protocol translation, caching, compression, QoS control, authentication, encryption and intrusion detection. Many of these services require the intervention of intermediate service gateways, like firewalls, VoIP gateways, NAT routers, VPN gateways and broadband access servers.
These services are sometime referred to as session-oriented services [22], because they operate on traffic flowing between pairs of source and destination nodes. In the case of a stub AS, some of these source and destination nodes are likely to be edge routers connected to the hosts, whereas in the case of a large transit AS, these nodes are two border routers in the ingress and egress of the AS. The serviced traffic traverses the shortest path from the source to the service gateway, and then the shortest path from the gateway to the destination.

Traditionally, such service gateways have been placed on the boundary of an Autonomous System (AS), since all inter-domain traffic passes through it. However, there is a growing trend to place network services inside the AS. It was first shown by [28] that FTP traffic can be significantly reduced by placing caches in strategic locations inside the AS backbone. Since then, there has been a large volume of work that demonstrate the benefits of well-planned placement strategies in a variety of service contexts [22,43,51,68,79,89]. Such strategies take into account the distribution of traffic as well as the topology of the AS.

Research on service placement has concentrated mainly on placing the service gateways in a way that minimizes the average length of the traversed routes. Therefore, each flow always selects the service gateway that imposes the shortest possible route. However, this approach does not take into account the reciprocal effect of individual flows, the load imposed on the network links, and the possible existence of hotspots (congested areas) in the network.

Another important trend in recent years is the adoption of traffic engineering inside large ASs. Traffic engineering [6] is related to a set of actions dealing with performance evaluation and performance optimization of operational IP networks. The performance of an operational network is enhanced by addressing traffic oriented performance requirements, such as delay, delay variation, packet loss, and throughput, while utilizing network resources economically and reliably. Most importantly, traffic engineering is used to control and optimize the routing function so that traffic can be steered through the network in the most effective way.

In this work we propose a novel approach for the service placement problem. The idea is to leverage service placement in the AS to facilitate traffic engineering. We take advantage of the ability of the source node to route traffic flows through service gateways, in order to control the load in certain areas of the AS. Instead of trying to minimize the length of detour routes, we extend and utilize these detours in order to relieve congestion in hotspots. To the best of our knowledge, this is the first work that employs such an approach for the service placement and gateway selection problems.

We address the problem by dividing it into the following two sub-problems:

1. The service placement problem: finding the best location for each service gateway.
2. The gateway selection problem: selecting the best service gateway to accommodate each flow.

The service placement problem is addressed in the offline context, by considering the long term average distribution of the source-destination traffic for each service type, which can be obtained using traffic matrix estimation techniques [78, 103]. We use this information to decide on the best location for each service gateway. In contrast, the gateway selection problem is addressed in the online setting. That is, each flow is associated with a service gateway, which is determined by current network conditions.

The rest of the chapter is organized as follows. In Section 3.2 we present detailed scenarios for which our placement and selection schemes are applicable. In Section 3.3 we present related work. Section 3.4 presents formal definitions of the problems we address. In Section 3.5 we present approximation algorithms, with performance guarantees, as well as efficient heuristics for both problems. In Section 3.6 we present extensive simulation results that demonstrate the significant performance gain achieved by our approach for real and synthetic topologies and various load settings. Finally, Section 3.7 concludes the chapter.

3.2 Application Scenarios

In this section we present two important application scenarios for the problem discussed in the chapter.

3.2.1 Open Access Networks

Users today are connected to the Internet almost from everywhere, using a variety of wireless and wireline technologies. There is often a distinction between the operator of an access network and the service provider (SP) who provides Internet access and/or other services (like telephony) to the end users. This concept, known as Open Access, is motivated by the following factors:

- The desire for regulations to protect the SPs from anti-competitive practices on the part of the access operators, in order to ensure that access to the Internet and other broadband services remain open to free competition.

- The expertise and assets required from an access operator such as a local or cable company are completely different from those required from an SP.

Figure 3.1 depicts a typical Open Access network that employs MPLS technology. This network connects the broadband link termination systems (BLTSs) such as cable-modem CMTSs, ADSL DSLAMs, and WiFi access points to the networks of the service providers. One of the most important functionalities of the Open Access
network is to decide to which service provider each packet sent by a user should be forwarded. This decision is different from a typical routing decision, because it is not performed according to the IP destination address. This address indicates the final target of the packet, which could be, for example, a web server, that is not necessarily located in the service provider’s domain.

The information on which the decision is based is not included in the BLTS, but rather in another network device, referred to as a broadband remote access server (BRAS). This is for two main reasons:

- There might be hundreds of BLTSs and tens of service providers. Establishing one tunnel, for best-effort traffic, or several tunnels, for multiple QoS classes, between every BLTS and service provider pair, would require too many tunnels.

- The decision as to which service providers a packet should be forwarded is sometimes very complicated, and might change in real-time (e.g., if the user is allowed to select an SP online). Such decisions are not in the scope of the BLTSs, which mainly function as traffic aggregators.

There is at least one MPLS tunnel between every BLTS and BRAS, and one tunnel between every BRAS and service provider. More tunnels can be used to address different service classes. When the BLTS identifies a new flow of a certain user, it needs to select the best BRAS to which this flow will be forwarded. Obviously, traffic engineering is one of the most important selection criteria, especially for traffic flows that require better than best-effort QoS. This has motivated the model and solutions presented in this work.

3.2.2 VoIP communication through border session controllers

Enterprises and operators that provide VoIP services over their IP networks often use a special device called a border session controller (BSC). When a local user initiates
Figure 3.2: VoIP communication through a border session controller

a VoIP call to a remote peer, the call is established through one of the local BSCs\(^1\), to the destination peer. In such a case, two RTP (Real Time Protocol) connections are established for the considered call: between the caller and the BSC, and between the BSC and the remote peer (see Figure 3.2).

The setup of a VoIP call through an intermediate device, rather than peer-to-peer, might increase the end-to-end delay. However, it has some important — and usually more significant — advantages:

- The codec used by the caller can be changed to another codec, either because the new codec is more efficient, or because the older one is not supported by the remote peer. This case is shown in Figure 3.2(a).

- The NAT (Network Address Translation) problem can be overcome if the local user uses a non-globally unique IP address. This is because the IP packets sent out of the local network will carry the globally-unique IP address of the BSC, rather than the non-globally unique IP address of the calling user. This case is shown in Figure 3.2(b).

- The firewall traversal problem of VoIP sessions can be overcome. Since firewalls are usually closed for UDP traffic, the local administrator can open the firewalls

\(^1\)In certain scenarios, a BSC is involved only in the control plane. However, here we consider the case where it is involved both in the control and data planes.
only for UDP connections that are originated in or are destined for the local BSCs, when these are employed. This case is also shown in Figure 3.2(b).

- Providing QoS from a small set of BSCs to the target peers is easier than providing QoS from each end user to these peers. For example, if MPLS tunnels are used, then the number of required tunnels is smaller by several orders of magnitude.

The concepts presented in this chapter in the considered BSC-based VoIP architecture can be taken advantage of as follows. Each VoIP client is configured with a local redirect proxy server. This server only participates in the signaling of the VoIP calls. To set up a new call, the client sends a SIP INVITE message to the proxy. The proxy takes into account the location of the client and of the potential BSCs. It selects one of these BSCs (e.g., using the algorithms proposed in this chapter), and responds to the client with a 3XX Redirect message, asking this client to establish the call through the selected BSC. The clients then sends an INVITE message to the selected BSC, and the BSC sends such a message to the remote peer.

### 3.3 Related Work

The problem of placing network intermediate devices has been extensively addressed in a vast range of fields. Most notably, much work has been done on the placement of caches, Web proxies [52, 69] and mirror servers [50, 92, 93]. Usually, these devices respond to the service request themselves; they do not forward any traffic to a destination node. However, as noted by [22], they can also be considered as private cases of session-oriented services in which the volume of traffic changes after it passes through the service gateway. Hence, our results are applicable to such services as well.

Another field where intermediate device placement has been researched is stream processing systems [1, 101]. These distributed systems are composed of autonomous devices that operate on continuous data streams. Each device performs a single function and then passes the traffic to other operators or to the final consumers. Applications include sensor networks, network management, and location-tracking. The goal here is to select appropriate devices while dynamically adjusting to the network load, thereby minimizing service latency and improving resiliency.

The main focus of all the above work is placement and selection of intermediate devices while minimizing bandwidth consumption or minimizing the distance (average or maximum) between the devices and the end users. The most notable algorithms proposed in these works are graph-theoretic algorithms, which are based on approximation algorithms for the $K$-center and $K$-median problems [22, 92]. Other algorithms are based on a greedy strategy [50, 92] or on the connectivity degree of the routers and the ASs [50, 93]. However, no work has yet addressed the reciprocal effect of
the serviced traffic flows. Ignoring this important issue may result in the creation of network hotspots, which may lead to exponential degradation in the service latency, or even to denial of service.

Some papers, such as [51, 80], suggest placing the service gateways in a distributed manner. This approach is claimed to be scalable, efficient, and adaptable to dynamically changing network conditions. However, it may be more prone to deployment difficulties, as stateful services have to be forwarded from one gateway to another, and clients need to continuously relocate the gateways. Furthermore, it was shown in [68, 85] that Web traffic in the long term is fairly stable, with relatively moderate changes. This implies that determining the location of gateways using long-term traffic distribution statistics can be very effective.

Placement and selection of relay nodes in a network is also addressed in the field of anycast servers. Anycast is a routing mechanism that forwards a packet to at least one node from a set of possible destinations. In [53], the authors investigate selection algorithms of already deployed anycast relay servers which forward the received traffic to the final destination. Their main conclusion is that the best selection algorithm is dependent on the placement algorithm for the servers. Other papers, such as [34, 109], suggest selection algorithms that minimize response time.

Relay nodes are also deployed in sensor and wireless networks. Relay nodes with higher energy capacity are deployed in these networks in order to shorten the transmission ranges of regular, more energy-constrained, nodes and to alleviate the need for them to relay data for other nodes. In [47, 56, 58], the authors suggest placement algorithms that optimize various metrics such as average node lifetime and average node congestion.

Finally, we remind you of the N-hub shortest-path routing scheme presented and explored in Chapter 2. In this routing scheme, a traffic flow can be routed through up to N intermediate nodes (hubs) before reaching its destination, while traversing the shortest paths between them. N-hub shortest-path routing was shown to be an effective approach for ensuring load balancing and achieving better utilization of network resources. In this chapter we build on this result and leverage the benefits of the N-hub shortest-path routing scheme.

### 3.4 Problem Model

The optimization metric we consider maximizes the volume of admitted traffic, subject to an upper bound on the load imposed on each link. This criterion is relevant for service providers that must provide guaranteed bandwidth to some of the flows, such as video or voice streams, which have strict QoS constraints. Our approach can also be applied to several other optimization metrics, such as minimizing the delay, minimizing the average packet loss, or minimizing the maximum load.
We now formulate the two problems described in Section 3.1: placement of service gateways, and selection of service gateways. We assume that the network offers several services, each of which is offered by a set of service gateways. For the sake of simplicity, we assume that each flow demands at most one service, which is rendered by one gateway from a set of gateways that can deliver this service. Still, all the algorithms described in the next sections can be extended to address the case where a flow may demand multiple services and have, therefore, to traverse multiple gateways. Note that a flow may require no service at all. In this case, it should not be routed through a gateway, but through the direct path from its source to its destination. We assume that between every two nodes in the network (end nodes or service gateways) there exists a single predetermined route, to be used by the traffic between them. However, we make no specific assumptions regarding this route. Therefore, such a route can be the “standard” shortest path between the two nodes, or a pre-established MPLS tunnel. To avoid having to forward states from one service gateway to another, and to avoid packet reordering or route oscillation, we require that the same gateway serve each flow for the entire flow duration. A flow serviced by a gateway is said to be routed through that gateway. We call the assignment of hubs (service gateways) to flows in \( \mathcal{F} \) “1-hub routing” of \( \mathcal{F} \).

**Definition 2** Let \( \mathcal{H} \subseteq V \). An instance of 1-hub routing is \( \mathcal{H} \)-limited for a given set of flows \( \mathcal{F} \), if only nodes from \( \mathcal{H} \) serve as hubs for the flows in \( \mathcal{F} \).

A route in an \( \mathcal{H} \)-limited 1-hub routing is denoted a 1-hub(\( \mathcal{H} \)) route. The routing domain is represented by a directed graph, \( G = (V, E) \), where \( V \) is a set of routers and \( E \) is a set of directed links. A bidirectional link is represented by two counter edges. Let \( u(e) \forall e \in E \) be the bandwidth capacity of link \( e \). Let \( \mathcal{F} \subseteq V \times V \) be a set of flows and \( t(f) \forall f \in \mathcal{F} \) be the bandwidth demand of flow \( f \). Let \( S \) be the set of service types deployed in the network. Let \( \mathcal{F}_s \subseteq \mathcal{F} \), \( s \in S \), be the subset of flows that require a service of type \( s \). \( \forall s_1, s_2 \in S, \mathcal{F}_{s_1} \cap \mathcal{F}_{s_2} = \phi \) and \( \bigcup_{s \in S} \mathcal{F}_s \subseteq \mathcal{F} \). Let \( k_s \leq |V| \) is the maximum number of service gateways for service of type \( s \). Let \( p_{uv}, \forall u, v \in V \), be a predetermined path between nodes \( u \) and \( v \). A flow \((o,d)\) serviced by a gateway \( h \) must be routed on the path \( p_{oh} \) and then on the path \( p_{hd} \). A flow \((o,d)\) that does not demand a service must be routed on the path \( p_{od} \). Each gateway can be deployed on one of the network routers. The load of a link is the total traffic the link carries divided by its capacity.

The **placement problem** is defined as follows. For every \( s \in S \), find \( \mathcal{H}_s \subseteq V \), where \( |\mathcal{H}_s| \leq k_s \), for which there exists an \( \mathcal{H}_s \)-limited 1-hub routing of \( \mathcal{F}_s \subseteq \mathcal{F}_s \), and a direct routing of subset \( \mathcal{F}' \subseteq \mathcal{F} \setminus \mathcal{F}_s \), such that the total load imposed on each link does not exceed the threshold \( L \), and the total bandwidth of \( \bigcup_{s \in S} \mathcal{F}_s' \cup \mathcal{F}' \) is maximized.

The **selection problem** is defined as follows. For every \( s \in S \), let \( \mathcal{H}_s \subseteq V \) be the set of (already placed) service gateways. For every \( s \in S \), for every \( f \in \mathcal{F}_s \), either
assign a service gateway $h \in \mathcal{H}_s$ or reject it, and for every flow $f \notin \bigcup_{s \in S} \mathcal{F}_s$, either route $f$ directly to its destination or reject it, such that the sum of the bandwidth demands of the admitted flows is maximized, and the load imposed on every link does not exceed the threshold $L$.

Both problems are NP-complete even for the special case where $|S| = 1$, $\mathcal{F}_1 = \mathcal{F}$ and $\forall u, v \in V$ $p_{uv}$ is the shortest-path between $u$ and $v$. To prove that the decision variant of the placement problem is NP-complete, consider a slightly different problem: Find a set $\mathcal{H} \subseteq V$, where $|\mathcal{H}| \leq k$, for which there exists an $\mathcal{H}$-limited 1-hub routing of $\mathcal{F}$, such that the maximum link load is minimized. The decision variant of this problem is deciding for a given instance whether there exists a suitable set of hubs $\mathcal{H}$ for which there is an $\mathcal{H}$-limited 1-hub routing for all the flows while ensuring that the maximum load on a link does not exceed a threshold $L$. In Chapter 2 it is shown that finding “1-hub routing” under a similar optimization criterion but with is no upper bound on $k$ is NP-complete. This implies that the problem with $k = |V|$ is NP-complete as well. Hence, the problem with an arbitrary value of $k$ must also be NP-complete.

Getting back to our original problem, the decision variant of this problem is deciding for a given instance whether there exists a set $\mathcal{H} \subseteq V$ of hubs, where $|\mathcal{H}| \leq k$, for which there exists an $\mathcal{H}$-limited 1-hub routing for a subset of flows whose total bandwidth demand is not less than a threshold $T$, such that the load imposed on each link does not exceed $L$. Let us assume that there is an optimal polynomial algorithm for this problem. If we take $T$ to be the sum of the bandwidth demands of all the flows in $\mathcal{F}$, we can decide in a polynomial time whether there exists a suitable 1-hub routing that routes every flow in $\mathcal{F}$ while keeping the maximum load imposed on each link below $L$. However, such an algorithm can also solve in polynomial time the decision variant of the problem shown earlier to be NP-hard.

It can be easily shown that the offline decision variant of the selection problem is NP-complete, using, again, a reduction from the routing problem presented in Chapter 2.

As already said, we address the selection problem in the online context. A competitive ratio for an online algorithm is defined as the worst case ratio, over all sequences of flows, between the value of the solution found by the algorithm and the value of the solution found by an optimal offline algorithm [55]. In Chapter 2 it is shown that if the optimization criterion is to minimize the maximum load, and the flows have a limited and unknown duration, the best and worst competitive ratios are $O(|E|)$. It can be shown that a similar proof also applies for our selection problem. Hence, in the following discussion we assume that each flow has an unlimited duration or, alternatively, known duration. This assumption allows us to differentiate between the performance of the various algorithms by their competitive ratios.
3.5 Algorithms for The Placement and Selection Problems

3.5.1 Algorithms for the Placement Problem

In this subsection we present approximation algorithms and heuristics for the placement problem. We first propose an approximation algorithm whose worst-case performance is upper bounded.

Algorithm 1 We start by formulating the problem as an integer linear problem. Let \( F_S = \bigcup_{s \in S} F_s \). The following variables and parameters are defined:

- \( x_{if} \) - a binary variable, whose value is 1 if node \( i \in \{V, \phi\} \) is assigned as a hub for the traffic of flow \( f \), and 0 otherwise. The symbol \( \phi \) denotes a dummy hub. Namely, if \( \phi \) is assigned to \( f \), then \( f \) does not pass through any hub.

- \( X_f \) - a binary variable, whose value is 1 if flow \( f \) is admitted, and 0 otherwise.

- \( h^s_i \) - a binary variable, whose value is 1 if node \( i \) is used as a hub for some flow in \( F_s \), and 0 otherwise.

- \( z^e_{if} \) - for every flow \( f = (o, d) \), node \( i \in V \) and link \( e \), \( z^e_{if} = 1 \) if \( e \) is on \( p_{od} \) or \( p_{do} \), and 0 otherwise. If \( i = \phi \) \( z^e_{if} = 1 \) if \( e \) is on \( p_{od} \), and 0 otherwise.

The target function is:
maximize \( \sum_f t(f) \cdot X_f \)
subject to the following constraints:

\[
\begin{align*}
(a) \quad & \forall f \quad \sum_{i \in \{V, \phi\}} x_{if} \geq X_f \\
(b) \quad & \forall f \in F_S \quad x_{\phi f} = 0 \\
(c) \quad & \forall f \notin F_S \quad x_{\phi f} = 1 \\
(d) \quad & \forall e \quad \sum_{i \in \{V, \phi\}, f} x_{if} \cdot z^e_{if} \cdot t(f) \leq Lu(e) \\
(e) \quad & \forall i \in V, s, f \in F_s \quad x_{if} \leq h^s_i \\
(f) \quad & \forall s \quad \sum_{i \in V} h^s_i \leq k_s \\
g) \quad & \forall i \in \{V, \phi\}, f, s \quad x_{if} \in \{0, 1\}, \quad X_f \in \{0, 1\}, \quad h^s_i \in \{0, 1\}
\end{align*}
\]

The linear relaxation of the program allows each variable to be assigned any real value in \([0, 1]\). This implies that the requirement to route the entire bandwidth demand of every flow \( f \in F_S \) on a single route is now relaxed. Note, however, that a flow \( f = (o, d) \in F \setminus F_S \) (i.e., a flow served by no gateway) still can be routed only on \( p_{od} \). Though not all it’s bandwidth demand may be accommodated. After finding an optimal solution for the relaxed linear program, we have for every flow \( f = (o, d) \in F_S \) a set
of hubs $\Gamma_f$ through which $t(f)$ or part of it is routed. An empty $\Gamma_f$ indicates that $f$ is not admitted. Every hub $h \in \Gamma_f$ defines a route from $o$ to $d$, which consists of $p_{oh}$ and $p_{hd}$. Each such a route carries a fraction of $x_{if}$ from $t(f)$. In order to convert the solution to the relaxed linear program into a solution to the original integer program, each hub $h \in \Gamma_f$ is associated with a weight that equals to that fraction. The sum of the weights for every $\Gamma_f$ equals $X_f$.

The weight of hub $i$ in every $\Gamma_f$ is rounded to 1 with probability $x_{if}/\gamma$, where $\gamma$ is some constant larger than 1 whose value will be discussed latter, and to 0 with the complementary probability. There might be flows for which more than one hub is chosen for routing the whole demand, and other flows for which no hub is chosen. In the former case, one of the hubs is arbitrarily selected, whereas in the latter case the flow is not admitted for routing. These decisions are made independently for each flow $f$. A flow $f \in F \setminus F_S$ that requires no service is admitted with probability $x_{df}/\gamma$. 

In appendix 3.8 we prove that there exists a value $\gamma$ for which the above probabilistic algorithm can be transformed to a deterministic one that yields a feasible solution. That is, the number of service gateways does not exceed $k$ and the load imposed on every link does not exceed 1. We also show that the admitted bandwidth of the solution is not smaller than $\Omega(\max\{(y^*)^2/|E|, y^*/\sqrt{|E|}\})$, where $y^*$ is the bandwidth admitted by an optimal solution for the relaxed problem.

Note that in certain cases, when the number of the service gateways is small compared to the number of flows, the service gateways may become congestion hotspots. In these cases, the traffic that can be admitted into the network is bounded by the capacity of the links surrounding the service gateways. This gives rise to the following heuristic, which maximizes this bandwidth.

**Algorithm 2** Let $D^v = \min\{D^v_{in}, D^v_{out}\}$, where $D^v_{in} = \sum_{e \in \Gamma_{(s,v)}} u(e)$ and $D^v_{out} = \sum_{e \in \Gamma_{(v,d)}} u(e)$. We call $D^v$ the capacity of $v$, and $\sum_{f \in F} t(f)$ the bandwidth demand of $s$. Set all vertex capacities and service bandwidth demands as unassigned. First, assign the vertex with the highest unassigned capacity to the service with the highest unassigned bandwidth demand. Then, subtract the unassigned capacity of the vertex and the unassigned bandwidth demand of the service accordingly. These two steps iterate until each service is assigned $k_s$ vertices.

Note that Algorithm 2 does not take into account the distribution of the flows in the network. It only maximizes the capacity of the service gateways, in order to reduce their load.

As mentioned above, the goal of most placement algorithms proposed in the past is to minimize the average or maximum routes of the flows. Algorithms based on approximations for the $K$-center and $K$-median problems have been proposed in [22, 92]. They can be summarized as follows.
**Algorithm 3** For every service \( s \in S \), construct an instance of the \( K \)-median/\( K \)-center problem from an instance of the placement problem. For every service \( s \in S \), every flow \( f = (o, d) \in \mathcal{F}_s \) is considered as a client. Each router \( v \in V \) is considered as a site. The distance between a client and a site is set to the length of the path in \( G \) between the source of the corresponding flow and the site, plus the length of the path in \( G \) between \( v \) and the destination of the corresponding flow. Then, use an approximation algorithm for the \( K \)-median or the \( K \)-center problem (e.g., [72]) that selects a set of \( k_s \) vertices to be used as the locations for the gateways of service \( s \).

These algorithms are considered computationally intensive. Hence, a greedy placement strategy was proposed as an efficient alternative [50, 92].

**Algorithm 4** For every service \( s \in S \), choose the \( k_s \) locations in \( k_s \) iterations. In each iteration select the location which, in conjunction with the locations selected in previous iterations, imposes the smallest average of route lengths for the flows in \( \mathcal{F}_s \).

### 3.5.2 Algorithms for the Selection Problem

We now present algorithms for the online selection problem. When a flow is introduced into the network, the selection algorithm can either reject it or admit it. If necessary, it can also assign to the flow a service gateway. The service gateway is selected from a set \( \mathcal{H}_s \) of gateways that offer the required service and whose locations have been determined in advance. The rejection and assignment decisions are made without being aware of forthcoming flows. Furthermore, it is assumed that an active flow cannot be stopped or reassigned to a different gateway.

The selection problem can be translated into a special case of a well-known traffic engineering problem called the **online unsplittable flow problem** [8]. In this problem, every flow may be routed over an arbitrary route. In our problem, a flow may only be routed over a limited set of routes that must pass through one of the service gateways, or over a single direct path. In the following, we extend the results for the general unsplittable flow problem to address our selection problem.

We first present gateway selection schemes employed in previous work (e.g., [22, 50, 92]).

**Algorithm 5** If \( f \in \mathcal{F}_s \), select from \( \mathcal{H}_s \) the gateway that imposes the shortest path on the flow.

The main advantages of this algorithm are its simplicity and its ability to minimize the consumption of network resources. Indeed, the scheme performs very well in practice when traffic is distributed evenly across the network.

Another example of an algorithm employed in previous work is the following closest gateway algorithm.

46
Algorithm 6 If $f \in F_s$, select the gateway from $H_s$ that is closest to the source of the flow.

This scheme does not require the source to know the network topology, but only its distance to every service gateway.

The above algorithms belong to a class of algorithms that do not take into account the load distribution but only the network topology. These algorithms employ only one path between a source-destination pair, regardless of possible network hotspots. Consequently, they might create congestion if the traffic is not evenly distributed.

A more sophisticated class of algorithms takes into account not only the network topology but also the current load on every link, [7, 8]. We now adapt the algorithm presented in [8] to our selection problem. The algorithm assumes that $t_{\text{max}} \leq u_{\text{min}}/P$, where $t_{\text{max}}$ and $u_{\text{min}}$ are the maximal bandwidth demand and minimal edge capacity, respectively, and $P \geq 2$. The algorithm yields a competitive ratio of $O(P|V|^{1/P}).$

Algorithm 7 Let $D$ denote the length of the longest 1-hub route in $G$. Choose one of the following two routing methods with equal probability, and route each incoming flow request accordingly.

1. Set $\mu = 2D$. If the flow’s bandwidth demand is larger than $\frac{u_{\text{min}}}{P+1}$, then route the flow over $\pi$ and set $\forall e \in \pi$ $L(e) = L(e) + \frac{1}{\frac{u_{\text{min}}}{P+1}}$, if one of following conditions holds:
   - $f \in F_s$ and there exists a 1-hub($H_s$) route $\pi$ for $f$ for which $\sum_{e \in \pi} (\mu^L(e) - 1) < D$ holds,
   - $f = (o, d) \notin F_S$ and $\sum_{e \in \pi} (\mu^L(e) - 1) < D$ holds for $\pi = p_{od}$.
   Otherwise, reject this flow.

2. Set $\mu = (2D)^{1+\frac{1}{P+1}}$. If the flow’s bandwidth demand is less than $\frac{u_{\text{min}}}{P+1}$, then route the flow over $\pi$ and set $\forall e \in \pi$ $L(e) = L(e) + \frac{u(e)}{\mu^L(e)}$, if one of following conditions holds:
   - $f \in F_s$ and there exists an 1-hub($H_s$) route $\pi$ for the considered flow for which $\sum_{e \in \pi} (\mu^L(e) - 1) < D$ holds,
   - $f = (o, d) \notin F_S$ and $\sum_{e \in \pi} (\mu^L(e) - 1) < D$ holds for $\pi = p_{od}$.
   Otherwise, reject this flow.

It can be shown that this algorithm achieves a competitive ratio of $O(P|V|^{1/P})$ for our selection problem. The proof is similar to the one presented in [8] because the following claims always hold: 1) An admitted flow is routed on a path whose cost is
less than $D$; 2) A rejected flow has no path whose cost is smaller than $D$. As in [8], a cascade network, in which all simple routes can also be viewed as 1-hub routes, can be used to show that the above algorithm achieves the best competitive ratio.

One drawback of this algorithm is that the current load on every network link cannot be delivered to the nodes in real time. If this information is not available, the algorithm's performance might be impaired as a result.

To overcome this drawback, algorithms with some a priori knowledge about forthcoming flows are presented in [54] and [102]. The algorithm in [54] knows only the source-destination pairs of future flows; it predicts where hotspots are and diverts traffic to less congested areas. The algorithm in [102] also knows the average traffic between every pair of nodes, additional information which leads to improved performance over [54]. We take advantage of this result, and present the following heuristic for the selection problem.

Algorithm 8 Given a set of service gateways $\mathcal{H}_s$, $s \in S$, solve the relaxed linear program presented in Section 3.5.1 while setting the variable $h^*_i$ to 1 if $i \in \mathcal{H}_s$, and to 0 otherwise. The target value of the solution is a lower bound for the optimal offline selection problem. Let $x^*_i$ and $X^*_j$ be the values assigned by the solution to variables $x_{if}$ and $X_{jf}$, respectively. Admit a flow $f$ with probability $X^*_j$. If $f$ requires a service, gateway $i$ is selected for the flow with probability $x^*_i / X^*_j$.

In order to solve the relaxed linear program, a source node need only know the long-term average traffic distribution. It does not require up-to-date information regarding the load of the links.

### 3.6 Simulation Study

In this section we evaluate the performance of the placement and selection algorithms presented in Section 3.5. We use router-level AS topologies, generated with the Barabasi-Albert model [9], using the BRITE simulator [77]. The Barabasi-Albert model captures two important characteristics of AS topologies: incremental growth and preferential connectivity of new routers to well-connected existing routers. These characteristics yield a power-law degree distribution of the routers.

To validate our results, we also use actual ISP topologies, as inferred by the RocketFuel project [99]. The bandwidth for each link is based on [74]. For each synthetic or real AS topology, we generate a traffic matrix according to a power-law distribution. A traffic matrix and a network topology form together one instance of the placement problem. For each such an instance, we determine the locations of the service gateways using the following algorithms:

1. The probabilistic approximation algorithm, Algorithm 1, referred to in the following as Prob.
2. The maximum bandwidth location heuristic, Algorithm 2, referred to in the following as Max-BW.

3. The K-median algorithm, Algorithm 3, referred to in the following as K-median.

4. The greedy algorithm, Algorithm 4, referred to in the following as Greedy.

5. A random algorithm, which chooses the location for each gateway randomly from the entire set of network nodes, referred to in the following as Rand-Loc.

To solve the linear programs in algorithm Prob, we use the Lp_Solve software [12].

Most of our simulations focus on the case where all the flows require the same service. At the end of the section we validate these results we with simulation results for the more general case. A sequence of flows is generated using the average traffic distribution given by the traffic matrix. Taking the average traffic between two vertices, we determine the average life-time of every flow, the average bandwidth demand for every flow, and the average flow inter-arrival times. Throughout the simulation study we change the offered load in the network by adjusting the network-wide average of these three parameters.

Each flow is associated with the following parameters: a source node, a destination node, bandwidth demand, arrival time, and time duration. The network topology, the sequence of flows, and the locations of the k service gateways form together one instance of the selection problem. For each such instance, we determine the allowable traffic volume in the network using each of the following selection algorithms:

1. Shortest Path, Algorithm 5, referred to in the following as SP.

2. Closest gateway, Algorithm 6, referred to in the following as Closest.

3. Algorithm 7, which uses the exponential link weights, referred to in the following as Exp.

4. The optimal estimation heuristic, Algorithm 8, referred to in the following as Est-Opt.

5. A random algorithm that chooses a random service gateway, referred to in the following as Rand-Sel.

For each selection scheme, the flow is admitted if the route through the selected gateway does not violate any of the network capacity constraints. Otherwise, it is rejected.

Every simulation scenario in this section is tested about 400 times: 10 different network topologies were generated and 40 traffic matrices used for each topology. For all simulation results, the widths of the confidence intervals of the expected admitted traffic (using a confidence level of 95%) were less that 5% of the average value.
We start by evaluating the placement algorithms. For each set of gateway locations selected by a placement scheme, we solve the relaxed offline selection problem described in Section 3.5.2. The optimal solution for this problem gives an upper bound on the traffic volume that can be admitted into the AS for the specific gateway locations. This allows us to evaluate the potential effectiveness of every placement scheme, regardless of the selection scheme. Figure 3.3 depicts the results for a medium-size AS, with 50 routers whose average degree is 4 links. Each placement scheme produces a set of 7 gateway locations. The y-axis represents the total accepted traffic divided by the maximum admitted traffic from all simulation scenarios. The x-axis represents the offered load, which is the average amount of bandwidth that needs to be serviced at any given time between two nodes (i.e., the average bandwidth demand per flow multiplied by the average flow life-time divided by the average flow inter-arrival time).

It is evident from the graph that Prob and Max-BW have the best potential effectiveness. The potential effectiveness of Prob is attributed to its ability to place the gateways while taking the traffic distribution into account. The fact that Max-BW has roughly the same potential effectiveness is somewhat unexpected, because it does not take into account traffic distribution. In this simulation setting, however, we note that the service gateways create bottlenecks in the network. Therefore, by increasing the bandwidth around the gateways, Max-BW is able to increase the admitted traffic. To verify this, we also measure what percentage of the rejected flows were rejected because of congestion on the links of the service gateways. On the average, this
percentage is 75% for Max-BW, and 85% for every other algorithm. This proves that a majority of the rejected flows is indeed attributed to congestion around the service gateways.

The potential effectiveness of Greedy and K-Median is, respectively, 20% and 45% less than the potential effectiveness of Prob and Max-BW. This is because these algorithms aim to minimize the average length of the chosen routes. Hence, they place the gateways close to congested areas in the AS. K-median performs worse than Greedy because K-median is much more successful in minimizing the average route length. The Rand-Loc scheme performs better than K-median, since it places the gateways uniformly throughout the AS, thereby allowing some degree of load balancing.

Next, we evaluate the combined performance of the placement and selection algorithms. Again, we consider a medium-size AS with 7 service gateways and 50 routers whose average degree is 4. For every set of service gateways determined by the placement algorithms, we apply each of the selection algorithms presented above. The graphs in Figure 3.4 depict the results. To compare the performance of the various combinations, we use the following relative performance metric: the ratio between the bandwidth of flows admitted by the placement and selection algorithms and the bandwidth of flows admitted by the same placement algorithm combined with the optimal solution of the relaxed offline selection. This relative performance forms the y-axis of all the graphs in Figure 3.4, while the offered load forms the x-axis.

First, it is evident from all the graphs that the relative performance of each selection algorithm is roughly the same for all placement algorithms. The selection algorithm that exhibits the best performance is Est-Opt. For every placement algorithm, Est-Opt admits flows whose bandwidth is almost 80% of the bandwidth of flows admitted by the optimal solution. This can be explained by the fact that Est-Opt is the only selection algorithm that is aware of the long-term average traffic distribution. Surprisingly, Rand-Sel performs roughly the same as Exp, despite the fact that Exp is aware of the AS topology and the current link loads. This can be attributed to the fact that all gateway links are highly congested. Therefore, it does not matter which gateways are selected, as long as the load is evenly distributed among them.

It is also evident that SP and Closest exhibit the worst performance. This is, of course, because both schemes select only one gateway for each source-destination pair, which leads to poor load balancing. Closest is inferior to SP since its selection imposes longer routes, which consumes more network resources.

We can conclude from Figure 3.4 that the placement algorithms Prob and Max-BW combined with the selection algorithms Est-Opt, Exp and Rand-Sel exhibit better performance than every other combination.

We now take a closer look at the performance of the above mentioned combinations, by changing the number of service gateways and the link density. Again, we
Figure 3.4: The relative performance for every combination of placement and selection algorithms
consider two AS types, each with 50 routers. One type has an average link degree of 4 while the other has an average link degree of 7. We change the number of service gateways for every AS and measure the traffic admitted by the algorithms. Figure 3.5 depicts the results. The y-axis in the graphs represents the admitted traffic divided by the offered load, while the x-axis represents the number of gateways.

It is evident from both graphs that when the number of service gateways increases, so does the volume of admitted traffic. This indicates, again, that when the offered load is sufficiently high, the gateways are likely to create bandwidth bottlenecks. As the number of gateways increases, the bandwidth they can service increases as well, and their immediate vicinity becomes less congested. When the link density is 4, the increase in traffic admitted as a result of the increase in the number of gateways is more moderate. This is because when there are many service gateways, the bottlenecks are shifted from the gateway areas to other areas of the AS. However, when the link density is 7, the AS has much greater capacity, and the gateways impose again bandwidth bottlenecks. From Figure 3.5(a) we can see that when the service gateways no longer constitute the bandwidth bottleneck, Max-BW does not perform as well as Prob, and Exp performs better than Rand-Sel. This is because the gateways are less congested, so that judicious placement and selection algorithms that take into account the traffic load – Prob and Exp, for example – perform better. Another important point is that when the link density increases, Exp performs better relatively to the other algorithms. This is related to the network links being less congested. Hence, a prudent decision that takes into account the varying network loads achieves better load balancing.

To validate the results from the synthetic graphs, we run similar simulations,
but this time over real AS topologies as inferred by the RocketFuel project [99]. The results for these simulations are shown in Figure 3.6. The topology studied in Figure 3.6 is of the Exodus ISP, which consists of 80 routers and 147 links (link density = 1.8). Comparing Figure 3.6 to Figure 3.5 reveals that the considered combinations of algorithms have similar performance to what we found in the synthetic graphs. To further validate our results, we run simulations with a different demand model, called the gravity model [67,111]. In this model the offered load from node \( u \) to node \( v \) is proportional to the product of the total traffic volume exiting \( u \) and the total traffic volume entering \( v \). The results for these simulations are shown in Figure 3.7. The AS topology in this study has 30 routers and link density of 4. It is generated using the BRITE simulator just as described previously. For each router in the AS we use the Zipf distribution to determine the total amount of traffic entering and exiting the router. By comparing Figure 3.7 to Figure 3.5 we see that the considered combinations of placement and selection algorithms have similar performance to what we found before.

We now examine the case where not all the traffic in the AS demands a service. Such flows are not routed through a service gateway, but traverse the shortest-path between their source and destination. Such flows restrict our ability to balance the load in the AS. To investigate the effect of these “direct flows” on the total admitted traffic, we run simulations where the ratio of the traffic that requires a service varies
Figure 3.7: The admitted traffic for the best combinations of algorithms, as a function of the number of gateways, for the gravity demand model

from 20% to 100% of the total traffic. We run the three best combinations of algorithms as previously found: Prob+Exp, Prob+Est-Opt and Max-BW+Est-Opt. In addition, we examine the two best placement algorithms, Max-BW and Prob, with the commonly used SP selection algorithm, and the most commonly used combination of algorithms: K-median and SP. The AS in these simulations has 30 routers, link density of 4, and 7 service gateways. Figure 3.8 depicts the results. The y-axis in the graphs represents the admitted traffic divided by the maximum traffic admitted during the simulation. The x-axis represents the fraction of traffic that has to be serviced.

In Figure 3.8 it is most important to note that even when only 20% of the traffic needs to be routed through a service gateway, the performance differs significantly for two sets of combinations of algorithms. The first set includes the three combinations of algorithms that do not use SP for routing. The relative admitted traffic for this set is over 0.7. The second set includes the combinations of algorithms that do utilize SP. For this set the relative admitted traffic is less than 0.5. This implies that judicious placement and selection of service gateways in the AS may yield a significant performance gain even if only a small portion of traffic needs to be routed through them. As the fraction of serviced flows increases, the performance difference between the two sets of algorithms increases as well. In fact, the traffic admitted by the first set of combinations increases while the traffic admitted by the second set does not. This is due to the fact that even though the gateways are placed judiciously in the
AS, the SP selection algorithm does not try to load balance the traffic.

As noted above, the main drawback of Exp is that it requires every node to have information on the current load of every network link. This information can be obtained via an extension to a link state routing protocol, like OSPF-TE [57]. However, this information cannot be always obtained in real-time, mainly because update packets cannot be sent immediately after every change in the link load. We now study the performance of Exp under this practical constraint. Figure 3.9 depicts the performance of Exp as a function of the update interval. As in the previous simulations, short-term traffic flows are created between every two nodes in accordance with the long-term average offered load imposed by the traffic matrix. The average long-term offered load between every two nodes is determined from the following parameters: the bandwidth demand per flow, the lifetime of the flows, and their inter-arrival time. These parameters are generated for each flow using an exponential distribution. The $y$-axis in the graph represents the admitted traffic normalized to the case where this update interval is 0. The $x$-axis represents the update interval normalized to the average inter-arrival time of flows with the same source and destination. In this simulation, Exp is used in conjunction with Prob. However, we saw similar behavior of Exp with the other placement algorithms.

First, it is evident that as the update interval increases, the admitted traffic decreases. This is due to the fact that nodes have a greater probability to choose routes that are no longer available. As the update interval increases, the nodes make such
wrong decisions for longer periods of time. It can be seen that when the normalized update interval is 32, the admitted traffic drops by half. However, when the normalized update interval is less than 4, the performance loss is less than 4%. This implies that Exp can tolerate inaccurate information for short periods of time while achieving roughly the same performance level as with accurate information.

As discussed earlier, the selection algorithm Est-Opt and the placement algorithm Prob rely on a priori knowledge of the long-term average traffic distribution in the AS. The simulation results presented above are based on the assumption that these algorithms have accurate information regarding this distribution. However, in reality these algorithms will probably have only a rough estimate of this distribution. In the following we evaluate the effect of inaccurate information on the effectiveness of these algorithms. We add uniformly distributed random noise to the long-term average distribution of the traffic. Figure 3.10 depicts the performance of the three combinations: Prob and Est-Opt, Max-BW and Est-Opt, and Prob and Exp. We consider an AS with 30 routers whose link density is 4, and with 7 gateways. The y-axis in the graphs represents the admitted traffic divided by the admitted traffic when accurate information regarding the long-term traffic distribution is available. The x-axis represents the noise magnitude. A noise magnitude of $X$ means that if the real volume of traffic between a source-destination pair is $t$, then the algorithms know an inaccurate estimate of that volume that ranges uniformly between $t/X$ and $tX$. 

Figure 3.9: The relative performance of Exp, as a function of the update interval (gateways were placed using Prob)
Figure 3.10: The admitted traffic for selected combinations of algorithms, when random noise is added to the traffic distribution information.

As expected, Figure 3.10 reveals that as the random noise increases, the effectiveness of the algorithms decreases. Furthermore, the performance decrease is more drastic when Est-Opt is used as a selection algorithm. For example, the admitted traffic is reduced by almost 66% for a noise factor of 5. This suggests that the sensitivity of Est-Opt to traffic estimation errors is high. When Prob is used with Exp, the performance is less drastically affected. For example, when the noise factor is 5, the admitted traffic decreases by 40%. This is because Exp does not rely on traffic estimates.

Next, we evaluate the effect of AS evolution on the admitted traffic if the gateways are not dynamically relocated in accordance with topological changes. Figure 3.11 depicts the performance of three combinations of placement and selection algorithms: Max-BW and Est-Opt, K-median and Est-Opt, and K-median and SP. We consider an AS with 30 routers with average link degree of 4 and 7 gateways. The y-axis represents the admitted traffic as the AS evolves, divided by the admitted traffic for the baseline AS. The x-axis represents the number of nodes added to the AS since the last placement of the gateways. The new nodes are added to the AS topology according to the Barbaši-Albert model [9]. As the AS topology evolves, so does the traffic distribution, because the new nodes inject more traffic into the network.

From Figure 3.11 it follows that as the number of new nodes and the offered load in the AS increase, so does the admitted traffic. Adding 10 new nodes to the
network increases the admitted traffic by 67%, 132%, and 107% when using the algorithm combinations of K-median and SP, K-median and Est-Opt, and Max-BW and Est-Opt, respectively. The combination of K-median and SP has the most moderate increase. This is because the load on the service gateways is not properly balanced, and hot-spots in the AS are not relieved. The other two combinations utilize the capacity of the gateways more efficiently, with a sharper increase in the admitted traffic. Hence, we conclude that algorithms Max-BW and Est-Opt not only outperform the other algorithms, but also utilize the network resources more efficiently when the AS evolves.

Finally, note that in the model as presented in Section 3.4, the bandwidth demand of the flows is rigid. However, there are cases where the bandwidth demand of a flow is flexible. For example, during the startup of a multimedia session, it can change its demand by using different codecs. To examine the effect of such flexibility on the performance of the various algorithms, we run some simulations where bandwidth demand of a flow is halved if the original demand cannot be satisfied. This process repeats up to 3 times or until the demand can be satisfied. The profit gained by admitting a flow is relative to the flow’s satisfied bandwidth. Our results show that the combinations of Prob+Est-Opt, Prob+Exp, and Max-BW+Est-Opt have the best performance while the K-median+SP combination have the worst. These results are in agreement with the results shown earlier for the original model.
3.7 Conclusions

We proposed a novel approach for addressing the problems of placement and selection of service gateways. Rather than considering the need to route traffic to its destination through the gateways as a burden that has to be minimized, we take advantage of this need for the sake of traffic engineering. We translated the problems of gateway placement and selection to optimization traffic engineering problems whose objective is to maximize the admitted throughput. In this context, both problems are NP-complete. For the placement problem, we presented a probabilistic approximation algorithm (Prob) and an efficient heuristic (Max-BW). For the selection problem we presented an algorithm whose competitive ratio is bounded (Exp), as well as a simpler heuristic (Est-Opt).

We then conducted a detailed simulation study to examine the performance of these algorithms. We showed that when the service gateways create bandwidth bottlenecks in the network, a simple placement heuristic that maximizes the connectivity of the gateways (Max-BW) yields the best results. When the service gateways are not bandwidth bottlenecks, a placement algorithm that takes into account the expected traffic distribution (Prob) yields the best results. The selection algorithm that was shown to have the best performance is Est-Opt. This algorithm needs to know only the long-term average of the traffic distribution. Finally, the combinations of Est-Opt with Max-BW or with Prob yield significant improvement over algorithms that only try to minimize the length of the traffic routes through the gateways.

Our main conclusion is that placement and selection of network services can be employed as an effective tool for traffic engineering.

3.8 Appendix: A Proof for the Approximation Ratio of Algorithm 1 for the Gateway Placement Problem

We now prove that Algorithm 1 guarantees that the number of service gateways for every \( s \in S \) does not exceed \( k_s \), that the load imposed on every link does not exceed 1, and that the admitted bandwidth is not smaller than \( \Omega(\max\{(y^*)^2/|E|, y^*/\sqrt{|E|})\), where \( y^* \) is the bandwidth admitted by an optimal solution for the relaxed problem. The proof draws on the proof presented in [100] for the Unsplittable Flow Problem. The main difference is that here we also need to prove that the number of service gateways for every \( s \in S \) in the solution does not which exceed \( k_s \). In order to focus on our contribution we avoid reiterating some technical details in the proof. These details are fully presented in [100]. However, all information which is crucial to the understanding of the proof is included here.
Let \( x_{i,f}^* \), \( X_f^* \) and \( h_i^* \) be the values assigned to the corresponding variables by an optimal solution to the relaxed problem. Let \( L_e \) be a random variable representing the load on link \( e \in E \) after the randomized rounding procedure. Let \( \mathcal{H}_s \), where \( s \in S \), be the sets of nodes used as hubs in the final solution.

As in [100], we need to find well-behaved estimators for each of the following two “bad” events: edge \( e \) is overloaded, namely \( L_e > 1 \), and the objective function is not smaller than \( \Omega(\max\{|y|^2/|E|, y/\sqrt{|E|}\}) \). Let these two bad events be denoted as \( \mathcal{E}_e \) and \( \mathcal{E}_y \), respectively. Due to the nature of our problem, we also need to find a well-behaved estimator for additional \(|S| \) “bad” events, namely, that more than \( k_s \) hubs for service \( s \) are chosen (\(|\mathcal{H}_s| > k_s \)). Let these events be denoted by \( \mathcal{E}_h^s \).

The first two estimators are identical to those presented in [100]. We include them here for the sake of completeness. If all the demands are less than \( 1/2 \), we have:

\[
E[L_e] = \sum_{i,f} x_{i,f}^* \cdot z_{i,f}^e \cdot t'(f) \leq 2/\gamma,
\]

and

\[
\text{Prob}(\mathcal{E}_e) = \text{Prob}\left(\sum_{i,f} x_{i,f}^* \cdot z_{i,f}^e \cdot t'(f) \geq 2\right),
\]

where \( t'(f) = 2t(f) \). Hence, a possible estimator for \( \mathcal{E}_e \) is:

\[
\chi_i^e = \prod_{i,f} (1 + \delta)^{x_{i,f}^* \cdot z_{i,f}^e \cdot t'(f)} \frac{(1 + \delta)^\mu}{(1 + \delta)^{\mu(1 + \delta)}}
\]

where \( \mu = 2/\gamma \) and \( \delta = \gamma - 1 \). We know that \( E[\chi_i^e] \leq G(\mu, \delta) \), where

\[
G(\mu, \delta) = \left(\frac{e^\delta}{(1 + \delta)^{\mu(1 + \delta)}}\right)^\mu.
\]

If all the demands are greater than \( 1/2 \), we have:

\[
\text{Prob}(\mathcal{E}_e) = \text{Prob}\left(\sum_{i,f} x_{i,f}^* \cdot z_{i,f}^e \geq 2\right).
\]

Hence, a possible estimator for \( \mathcal{E}_e \) is:

\[
\chi_2^e = \Psi_2\left(\frac{x_{i,f}^* z_{i,f}^e}{\gamma} : \forall i, f\right)
\]

where

\[
\Psi_2(z) = \sum_{1 \leq i_1 < i_2 \leq n} z_{i_1} z_{i_2}, \text{ for } (z_1, z_2, \ldots, z_n) \in \mathbb{R}^n
\]

We know that \( E[\chi_2^e] \leq 2/\gamma^2 \).
Let us now consider the routing of only $U \subseteq F$. We construct a proper estimator for the bad event where the bandwidth of the admitted traffic is less than $y^*_U (1 - 1/e)/(2\gamma)$, where $y^*_U = \sum_{f \in U} t(f) \cdot X^*_f$. By [100], a possible estimator for this bad event is:

$$
\chi^y_U = \prod_{i=1}^n (1 - \delta_1) x^*_f / (1 - \delta_1) \mu_1 (1 - \delta_1),
$$

where $\mu_1 = y^*_U (1 - 1/e)/\gamma$ and $\delta_1 = 1/2$. We know that $E[\chi^y_U] \leq H(\mu, \delta)$, where $H(\mu, \delta) = e^{\mu \delta^2/2}$.

Next, we find a well-behaved estimator $\chi^h_i$ for the last bad events. For every $i \in V$, let $H^*_i$ be a random variable whose value is 1 if $i$ is chosen as a hub and 0 otherwise. Note that these random variables are independent of each other, since each route for each flow is chosen independently of the others.

We now find an upper bound on the expected value of each of these random variables:

$$
E[H^*_i] = \Pr(H^*_i = 1) = 1 - \prod_{f \in F} (1 - x^*_f / \gamma)
\leq 1 - \prod_{f \in F} (1 - h^*_i / \gamma) = 1 - (1 - h^*_i / \gamma)^{|F|}
\leq |F|h^*_i / \gamma.
$$

The last inequality follows from the fact that for all $x \in [0, 1]$, $1 - (1 - x)^n \leq nx$. We can now calculate the expected number of hubs $\mathcal{H}_s$:

$$
E[\mathcal{H}_s] = E[\sum_i H^*_i] = \sum_i E[H^*_i] \leq |F|h^*_i / \gamma
\leq |F|h^*_i / \gamma.
$$

Using similar considerations to those presented in [100] for constructing $\chi^t_e$, we conclude that

$$
\chi^h_s = \prod_{i=1}^n (1 + \delta_2) h^*_i / (1 + \delta_2) \mu_2 (1 + \delta_2), \text{ and } E[\chi^h_s] \leq G(\mu_2, \delta_2),
$$

where $\mu_2 = k_s |F| / \gamma$ and $\delta_2 = \gamma / |F| - 1$.

We define $\chi^h_{U, s}$ as a well-behaved estimator for $\mathcal{E}_s$ while only the subset $U \subseteq F$ is considered for routing. It is obvious that $E[\chi^h_{U, s}] \leq E[\chi^h_s]$.

Before proceeding the approximation ratio of the algorithm, we present the following theorem from [100].

**Theorem 5** Let $E_1, E_2, \ldots, E_t$ be events and $r, s$ be non-negative integers with $r + s \leq t$ such that:

- $E_1, E_2, \ldots, E_t$ are all increasing, with respective well-behaved estimators $g_1, g_2, \ldots, g_t$,
• $E_{r+1}, \ldots, E_{r+s}$ are all decreasing, with respective well-behaved estimators $g_{r+1}, \ldots, g_{r+s}$,
• $E_{r+s+1}, \ldots, E_t$ are arbitrary events, with respective proper estimators $g_{r+s+1}, g_2, \ldots, g_t$,
• all $E_i$ and $g_i$ are completely determined by $\bar{X}$.

Then, if
\[
1 - \left( \prod_{i=1}^{r} (1 - E[g_i]) \right) + 1 - \left( \prod_{i=r+1}^{r+s} (1 - E[g_i]) \right) + \sum_{i=r+s+1}^{t} E[g_i] < 1
\]
holds, we can efficiently construct a deterministic assignment for $\bar{X}$ under which none of $E_1, E_2, \ldots, E_t$ holds. (Empty products are taken to be 1. If there is $g_i$ such that $E[g_i] > 1$, then the entire product is equal to 0.)

We shall use the above theorem and the well-behaved estimator for the “bad” events mentioned above to prove the following.

**Theorem 6** A deterministic placement of $k$ service gateways that facilitates an admitted traffic of no less than $\Omega(\max\{(y^*)^2/|E|, y^*/\sqrt{|E|})$ can be found efficiently.

**Proof**

We start by showing the $\Omega((y^*)^2/|E|)$ bound. Let $F_0$ be the subset of flows for which traffic demand is at most $1/2$ and $F_1 = F \setminus F_0$. Let $y^*_U$ be the optimal objective function when only the subset $U \subseteq F$ is considered for routing. We first assume that $y^*_U \leq y^*_F$, i.e., $y^*_F \geq y^*/2$. Since the events $E_h$ and $E_e \forall e \in E$ are increasing and the event $E_y$ is decreasing, in order to avoid these events we must have,
\[
1 - \left( \prod_{h} E[\chi_{F_0}^h] \prod_{e} (1 - E[\chi_{F_1}^e]) \right) + E[\chi_{F_0}^y] \leq 1,
\]
where $E'(\cdot) \triangleq \min\{E(\cdot), 1\}$.

Since
\[
E[\chi_{F_0}^y] \leq H(y_U^*(1 - 1/e)/\gamma, 1/2) \leq H(y^*(1 - 1/e)/(2\gamma), 1/2),
\]
for a suitably large constant $e$ it can be shown that $\gamma = e|E|/y^*$ satisfies the above inequality. Thus, from Theorem 5 follows that if $y_{F_1}^* \leq y_{F_0}^*$, we can efficiently select feasible paths for $F_0$ with objective function value $\Omega((y_{F_0}^*)^2/|E|) = \Omega((y^*)^2/|E|)$.

Next, we consider the case where $y_{F_1}^* \geq y_{F_0}^*$. As in the previous case, in order to avoid all of the “bad” events we must have
\[
1 - \left( \prod_{h} E'[\chi_{F_1}^h] \prod_{e} (1 - E'[\chi_{F_2}^e]) \right) + E[\chi_{F_1}^y] \leq 1.
\]
Again, for a suitably large constant $c$, and from the same considerations as above, it can be shown that $\gamma = c|E|/y^*$ satisfies the above inequality. Thus, from Theorem 5 follows that if $y_{F_1}^* \geq y_{F_0}^*$, we can efficiently select feasible paths for $F_1$ with objective function value $\Omega((y_{F_1}^*)^2/|E|) = \Omega((y^*)^2/|E|)$.

To conclude the proof we need to show the $\Omega(y^*/\sqrt{|E|})$ bound. If $y^* \geq \sqrt{|E|}$, this immediately follows from the $\Omega((y^*)^2/|E|)$ bound. If $y^* < \sqrt{|E|}$, we simply choose to admit a flow $f$ for which $t(f) = 1$ holds. \qed
Chapter 4

Maximizing Restorable Throughput in MPLS Networks

In this chapter we study traffic engineering in MPLS restorable networks and present a comprehensive study on restorable throughput maximization in these networks. We present the first polynomial-time algorithms for the splittable version of the problem. For the unsplittable version, we provide a lower bound for the approximation ratio and propose an approximation algorithm with an almost identical bound. We present efficient heuristics which are shown to have excellent performance. One of our most important conclusions is that when one seeks to maximize revenue, local recovery should be the recovery scheme of choice.

The work described in this chapter was originally published in [24, 26].

4.1 Background

IP networks should support real-time applications that require stringent availability and reliability, such as Voice over IP and virtual private networks. Unfortunately, failures are still common in the daily operation of networks, for reasons such as improper configuration, faulty interfaces, and accidental fiber cuts [48, 76]. Therefore, mechanisms that restore the flow of traffic quickly and efficiently after a failure are essential.

The IP routing protocols are not suitable for fast restoration. Using these protocols, a node first detects a failure and then disseminates routing updates to other nodes. These updates are used for calculating new paths. This process takes several seconds before proper routing of data can resume [3, 48]. During this time, packets destined to some destinations may be dropped, and applications might be disrupted. Moreover, when QoS is supported, the routing protocol cannot guarantee that the alternate path will provide the same QoS as the failed one.
Figure 4.1: Illustrations of the various recovery schemes

For these reasons, many network operators do not rely only on IP routing protocols to restore traffic, but also employ recovery mechanisms in Layer 1 and Layer 2 protocols such as WDM, SONET/SDH, and MPLS. These recovery mechanisms guarantee fast restoration and high QoS assurance because they establish backup paths in advance, before a failure event takes place. Such recovery mechanisms are usually referred to as “protection” mechanisms, as opposed to “rerouting” mechanisms, which establish backup paths only after a failure occurs.

In this chapter we focus on MPLS-based protection mechanisms [96,97]. However, our results are also applicable to other Layer 1 and Layer 2 protection mechanisms. In keeping with MPLS terminology, we refer to the path that carries the traffic before a failure as a primary LSP, and the path that carries the traffic after the primary LSP fails as a backup LSP. Throughout the chapter we consider only bandwidth guaranteed protection. For this kind of protection, the backup LSP must be able to provide the same amount of guaranteed bandwidth provided by the primary LSP. To this end, resources should be reserved upon the establishment of each backup LSP, to be used only when the protected element – link or node – fails.

Many MPLS recovery schemes have been proposed. We classify these schemes as follows (see Table 4.2):

1. Global recovery (GR) schemes [97] (Figure 4.1(a)): In this class, each primary LSP has one backup LSP. The primary and backup LSPs share the same end
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSP</td>
<td>Label Switched Path, an established route in an MPLS domain</td>
</tr>
<tr>
<td>LSR</td>
<td>Label Switch Router, a routing device in an MPLS domain</td>
</tr>
<tr>
<td>GR</td>
<td>Global Recovery, one of the studied MPLS recovery schemes (see Section 4.1)</td>
</tr>
<tr>
<td>LR</td>
<td>Local Recovery, one of the studied MPLS recovery schemes (see Section 4.1)</td>
</tr>
<tr>
<td>RLR</td>
<td>Restricted Local Recovery, one of the studied MPLS recovery schemes (see Section 4.1)</td>
</tr>
<tr>
<td>FLR</td>
<td>Facility Local Recovery, one of the studied MPLS recovery schemes (see Section 4.1)</td>
</tr>
<tr>
<td>EkFLR</td>
<td>Extended k Facility Local Recovery, one of the studied MPLS recovery schemes (see Section 4.1)</td>
</tr>
<tr>
<td>UR</td>
<td>Unrestricted Recovery, one of the studied MPLS recovery schemes (see Section 4.1)</td>
</tr>
<tr>
<td>SCA</td>
<td>Spare Capacity Allocation, a widely used optimization metric for restorable traffic (see Sections 4.1 and 4.2)</td>
</tr>
<tr>
<td>S/U-PRFP</td>
<td>Splittable/Unsplittable Primary-restricted Restorable Flow Problem, a new optimization problem described in this chapter (see Sections 4.3.1 and 4.3.2)</td>
</tr>
<tr>
<td>S/U-RFP</td>
<td>Splittable/Unsplittable Restorable Flow Problem, a new optimization problem described in this chapter (see Sections 4.3.3)</td>
</tr>
<tr>
<td>S/U-FP</td>
<td>Splittable/Unsplittable Flow Problem, a well-known optimization problem [41,100]</td>
</tr>
<tr>
<td>S/U-PFP</td>
<td>Splittable/Unsplittable Primary-restricted Flow Problem, a restricted version of S/U-FP (see Sections 4.3.2)</td>
</tr>
</tbody>
</table>

Table 4.1: Abbreviations and acronyms used in Chapter 4
<table>
<thead>
<tr>
<th>Recovery scheme class</th>
<th>Protected elements</th>
<th>Start node</th>
<th>End node</th>
<th>Protected primary LSPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>GR</td>
<td>all elements of primary LSP</td>
<td>head of primary LSP</td>
<td>tail of primary LSP</td>
<td>single LSP</td>
</tr>
<tr>
<td>LR</td>
<td>single element</td>
<td>immediately upstream of protected element</td>
<td>tail of primary LSP</td>
<td>single LSP</td>
</tr>
<tr>
<td>RLR</td>
<td>single element</td>
<td>immediately upstream of protected element</td>
<td>immediately downstream of protected element</td>
<td>single LSP</td>
</tr>
<tr>
<td>FLR</td>
<td>single element</td>
<td>immediately upstream of protected element</td>
<td>immediately downstream of protected element</td>
<td>all LSPs traversing the element</td>
</tr>
<tr>
<td>EkFLR</td>
<td>single element</td>
<td>immediately upstream of protected element</td>
<td>immediately downstream of protected element</td>
<td>a subset of LSPs traversing the element</td>
</tr>
<tr>
<td>UR</td>
<td>arbitrary</td>
<td>arbitrary</td>
<td>arbitrary</td>
<td>single LSP</td>
</tr>
</tbody>
</table>

Table 4.2: Characteristics of the recovery scheme

nodes. The backup LSP protects against all link/node failures along the primary LSP, and it does not share any link/node with the primary LSP. A failure notification must propagate from the node that detects the failure to the head of the LSP. These recovery schemes are sometimes referred to as path recovery schemes.

2. Local recovery (LR) schemes [83, 97] (Figure 4.1(b)): In this class, a separate backup LSP is constructed to protect against a possible failure of each element along the primary LSP. Each backup LSP starts at the immediate upstream node of the protected element, and ends at the tail of the primary LSP. If such a local path does not exist, we assume that the backup LSP starts at the closest possible upstream node. A backup LSP may share links with the primary LSP upstream of the failure. The recovery in this class is faster than in the GR class, since the node that detects the failure is usually also the one that diverts the traffic to the backup LSP. However, more backup LSPs are needed to protect
each primary LSP.

3. Restricted local recovery (RLR) schemes (Figure 4.1(c)): As in the LR scheme, a backup LSP starts at the immediate upstream node of the protected element but ends at the immediate downstream node. This makes the recovery process more local, since the route downstream of the failure does not change. Hence, unlike in the LR scheme, resources need not be released downstream of the primary LSP failure. The RLR schemes are sometimes referred to as link recovery schemes.

4. Facility local recovery (FLR) schemes [83] (Figure 4.1(d)): Backup LSPs are constructed as in the RLR schemes. However, a single backup LSP protects all the primary LSPs that traverse the protected element. This makes the process of restoring the traffic to the backup LSP simpler, using MPLS label stacking [96]. In addition, with this recovery scheme, the number of backup LSPs and the incurred state overhead are significantly reduced.

5. Extended k-facility local recovery (EkFLR) schemes [4] (Figure 4.1(e)): Backup LSPs are constructed as in RLR. However, there might be up to \( k \) backup LSPs that protect each element. Obviously, this scheme is more flexible than FLR, and permits the preferred trade-off between higher routing efficiency (\( k \) is larger) and lower administration overhead (\( k \) is smaller).

6. Unrestricted recovery (UR) schemes (Figure 4.1(f)): In this class, each primary LSP may be protected by any number of backup LSPs. Moreover, each backup LSP may start and end at any point along the primary LSP, and may protect against failures of any number of elements. This scheme incurs the highest administrative overhead while being the most flexible.

Table 4.1 summarizes the abbreviations and acronyms used throughout the chapter.

A failure is frequently limited to a single network element – a link or a node. Hence, it is customary to compare recovery schemes by measuring their performance under the assumption that a failure may occur only after the network has recovered from the previous failure. An important implication of this assumption is that two backup LSPs protecting against different failures may share their reserved bandwidth, as depicted in Figure 4.3. The single failure assumption may not hold in an optical WDM network, where the MPLS links are circuit-switched optical channels, called lightpaths [21]. Since a single physical fiber link may carry several lightpaths, a single failure in the optical layer may induce several MPLS link failures, an event known as “failure propagation”. Since the algorithms presented in this paper are complicated, we prefer to present them under the single failure assumption. However, in Section 4.7 we show how these algorithms can be extended to address MPLS tunnels over WDM lightpaths.
Most past research on the selection of backup LSPs is directed at minimizing the total bandwidth reserved for the backup LSPs. To this end, backup LSPs are routed to maximize their bandwidth sharing. This optimization metric is usually referred to as Spare Capacity Allocation (SCA). Models that seek to optimize SCA usually consider a network whose links have unbounded capacity, and a cost function associated with bandwidth usage. However, while minimizing the cost of building the backup LSPs is an important goal, network operators usually face a different optimization problem. They have a network with finite link capacities and seek to maximize their revenue by maximizing the traffic the network can accommodate. Another drawback of the SCA optimization for network operators is that the cost associated with an established LSP does not depend on the load imposed on the selected route. In other words, there is no incentive for load balancing.

Figure 4.2 illustrates the drawback of using the SCA metric for finite capacity networks. Suppose that all capacities are equal to 1 and have an equal cost of 1. We also have the following three flow demands: $f_1 = (5, 6)$, $f_2 = (3, 8)$, and $f_3 = (3, 4)$, each requesting one unit of bandwidth. Let us assume that the primary LSPs of the three flows are routed on their corresponding shortest paths, namely 5-6, 3-5-8, and 3-4. A routing that optimizes the SCA metric will route the backup LSPs of $f_1$ and $f_2$ through nodes 7 and 9, respectively. Now, in order to maximize bandwidth sharing, the backup LSP of flow $f_3$ will be routed on path 3-5-7-6-4 rather than on the shorter path 3-1-2-4. The former path is chosen since the backup LSP can share the bandwidth of links 2-7 and 7-6 with the backup LSP of $f_1$. Thus, the added cost of this path is only 2, whereas the shorter alternative incurs a cost of 3. However, this routing is not feasible, because the capacity constraint of link 3-5 is violated. Hence, one of the flows $f_2$ and $f_3$ must be rejected. If, however, our goal is not to minimize SCA but rather to maximize throughput, we would route the backup LSP
of $f_3$ through the underutilized path 3-1-2-4. This would allow all the three flows to be admitted.

In light of the above, SCA is not the most practical criterion for network operators. Hence, in this work we present a comprehensive study of the problem of constructing primary and backup LSPs while maximizing throughput. The main contributions of this work are as follows:

1. We show that the splittable version of the problem is in $\mathcal{P}$ and we offer the first polynomial time algorithm for it. In particular, we improve the results presented in [13], where only an FPTAS was shown.

2. We show that the unsplittable version of the problem is $\mathcal{NP}$-complete and has no approximation algorithm with a ratio of $|E|^{1/2-\varepsilon}$.

3. We propose an approximation algorithm with the ratio of $O(|E|^{1/2})$ for the case where the traffic demand of an individual flow does not exceed half of the edge capacities.

4. We present efficient heuristics that are shown to have excellent performance.

5. We compare the various recovery schemes with respect to the throughput maximization criterion. We show that UR, GR and, LR differ only marginally in their performance. Since LR has the fastest restoration time of the three schemes, it should be the scheme of choice.

6. We show that EkFLR with $k = 2$ has almost the same performance as RLR and should be preferred over it for its lower administrative overhead (fewer backup LSPs).

The rest of the chapter is organized as follows. Section 4.2 discusses related work. In Section 4.3 we formally define the problems addressed in the chapter and discuss their computational complexity. Section 4.4 presents algorithms for the problems.
In section 4.5 we conduct a simulative comparison of algorithm performance for the various recovery schemes. In Section 4.6 we extend the discussion to address node failures. Finally, Section 4.8 concludes the chapter.

4.2 Related Work

Research on the performance of recovery schemes for virtual circuit routing has been conducted not only for MPLS, but also for ATM and optical networks. The recovery principles in most of these schemes are similar: a primary path is protected by one or more pre-established backup paths to which the traffic is restored. As mentioned in Section 4.1, most previous work is directed at minimizing the Spare Capacity Allocation (SCA) metric. Many papers develop an Integer Linear Program (ILP) whose output is the set of backup paths that can fully restore the traffic on the primary paths. Since finding an optimal solution to an ILP is computationally hard, most of the papers focus on approximation algorithms.

The approximation algorithm in [73] chooses backup paths, one at a time, according to a GR scheme, and updates them iteratively. The approximation in [44] uses a rounding process for the relaxed LP. It then uses hop-limited restoration routes to round the LP solution. Other algorithms that address the SCA problem using the ILP formulation can be found in [15,16,36], and [106].

Some works use non-ILP methods to address the SCA problem. The approach employed by [2] is based on a genetic algorithm. The algorithm utilizes crossover and mutation operators to evolve “good” solutions toward optimality. These operators force disjoint backup paths to share their bandwidth. The algorithm finds GR paths and can also deal with a non-linear cost function. In [98], a local search algorithm which adopts the tabu search technique is proposed. In [108], a two-phase approach is proposed. First, a set of link-disjoint paths for a given set of demands is found. Then, using this set, an ILP-based selection of primary and backup paths is made.

In the online version of the SCA problem, each flow demand is handled independently, assuming no knowledge about future demands. Usually, each flow is assigned a primary and backup paths according to a specific recovery scheme. In [82], the authors show that finding primary and backup paths that consume the least amount of bandwidth is \( \mathcal{NP} \)-hard. For practical considerations, many of the online algorithms utilize only partial information regarding the network status. For example, in [60] an online algorithm based on the GR scheme is proposed. For every link, the algorithm needs to know only the loads imposed by the primary and the backup paths. The performance of this algorithm is shown to be close to that of an online algorithm with complete information on all primary and backup paths. The same authors apply the same techniques for the RLR scheme in [61]. The algorithms proposed in [70,90]
improve the results of [60] by utilizing more detailed information regarding the network status. In [90], each node also utilizes complete information about its adjacent links. The algorithm in [70] considers the load imposed by the backup paths on each link when the primary path fails. In [81], the authors propose to maintain even more detailed state information in what they call a backup load distribution matrix. This matrix maintains not only the primary and backup loads imposed on each link, but also, for every pair of links, the fraction of primary load on one link that is backed up on the other.

Only a few papers have proposed approximation algorithms with performance guarantees for the SCA problem or its variants. In [18], offline and online approximations for the FLR scheme are presented. These algorithms are based on approximation algorithms for the Steiner network problem. They present an $O(1)$ approximation for the case where the primary paths are predetermined. They also propose an algorithm for finding both primary and backup paths with an approximation ratio of $O(\log(n))$, where $n$ is the number of nodes in the network. An $\frac{14}{15}$-approximation algorithm that finds several backup paths for a single flow according to the GR scheme is presented in [14]. This algorithm reserves an integral part of the whole bandwidth for each of these backup paths, and it is shown to be the best possible for the considered problem.

Other works concentrate on non-SCA criteria. In [40,71], several heuristics based on the UR scheme are proposed for minimizing restoration time and bandwidth consumption. In [107], two additional parameters are introduced into the online SCA problem. These parameters allow the length of the backup paths to be reduced and bandwidth consumption to be reduced further. In [10], the authors address the online SCA problem while guaranteeing an upper bound on the delay of both the primary and the backup paths. They use UR and show it to have better performance than GR.

A few papers compare some of the various recovery schemes presented in Section 4.1. Most consider SCA as the optimization criterion. In [16,31,36,49,86,95,106], the main focus is on the GR and RLR schemes. Most of these papers use an ILP for optimizing SCA, and then approximate it. Their consensus is that GR outperforms RLR, because global backup paths are more flexible and generally traverse fewer links than RLR backup paths. Hence the GR scheme is more effective in decreasing the extra bandwidth that has to be reserved to ensure restorability.

Only a few papers use throughput maximization as an optimization criterion for the selection of primary and backup paths. In [65], an FPTAS based on the primal-dual approach is developed. The proposed algorithm deals with the splittable version of the problem, where each demand can be split into several primary and backup paths. This algorithm uses the RLR scheme. In [27], simple heuristics that find backup lightpaths for the restoration of IP over WDM are presented. This paper also compares the throughput and the restoration time of each algorithm.
4.3 Problem Definition and Computational Complexity

Two types of event failures can be considered: a link failure and a node failure. In [76] it was observed that link failures account for about 70% of the total network failures. We focus our discussion here on link failures. In Section 4.6 we extend the discussion and explain how to apply our results to node failures as well.

Throughout the chapter we deal with offline optimization. Namely, we assume that the operator knows the characteristics of the LSPs in advance. The justification for the offline model is that most of the LSPs are reserved in advance. For instance, a typical VoIP provider will reserve an LSP (VoIP trunk) with a certain bandwidth between two VoIP gateways for certain hours every day. Hence, the network operator can execute the offline algorithms once in a while, in order to optimize its revenue for the next hour(s).

In this work we address two cases. In the first we assume that the primary LSPs are predetermined and only the backup LSPs need be established, while in the second case we need to establish both. Most of this chapter is devoted to the former case, for two reasons. First, from a practical viewpoint, it is difficult for the network operator to retain joint optimization of primary and backup LSPs because of the implication that existing primary LSPs will require constant rearrangement to meet changing demands. Rerouting backup LSPs when necessary has far less impact on network operation than rerouting primary LSPs. It is therefore more practical to set up the primary LSPs, and then optimize the establishment of the backup LSPs separately. The second reason is that by setting up the primary LSPs in advance, we can focus our attention on comparing the performance of the various recovery schemes.

We refer to the problem where both primary and backup LSPs should be established as the Restorable Flow Problem (RFP). The problem where primary LSPs are given and only backup LSPs must be established is referred to as the Primary-restricted Restorable Flow Problem (PRFP). For each of the two problems we study the splittable and the unsplittable variants. In the next subsections we formally define these problems and address their computational complexity. Table 4.3 summarizes our main results in this section. For every problem and every recovery scheme, the table indicates whether the problem is $\mathcal{NP}$-complete, or in $\mathcal{P}$, or its complexity is unknown (?). For some of the $\mathcal{NP}$-complete problems we also give a lower bound on their approximation ratio.
<table>
<thead>
<tr>
<th>Recovery schemes</th>
<th>GR</th>
<th>LR</th>
<th>RLR</th>
<th>FLR</th>
<th>EkFLR</th>
<th>UR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S-PRFP</strong>&lt;br&gt;(Sec. 4.3.1)</td>
<td>$\mathcal{P}$ (Theorem 7)</td>
<td>$\mathcal{P}$ (Theorem 7)</td>
<td>$\mathcal{P}$ (Theorem 7)</td>
<td>$\mathcal{P}$ (Theorem 7)</td>
<td>$\mathcal{P}$ (Theorem 7)</td>
<td>$\mathcal{P}$ (Theorem 7)</td>
</tr>
<tr>
<td><strong>U-PRFP</strong>&lt;br&gt;(Sec. 4.3.2)</td>
<td>$\mathcal{NP}$-$\mathcal{C}$ (Theorem 8)</td>
<td>$\mathcal{NP}$-$\mathcal{C}$ (Theorem 8)</td>
<td>$\mathcal{NP}$-$\mathcal{C}$ (Theorem 8)</td>
<td>$\mathcal{NP}$-$\mathcal{C}$ (Theorem 8)</td>
<td>$\mathcal{NP}$-$\mathcal{C}$ (Theorem 8)</td>
<td>$\mathcal{NP}$-$\mathcal{C}$ (Theorem 8)</td>
</tr>
<tr>
<td></td>
<td>no $</td>
<td>E</td>
<td>^{\frac{1}{2}-\epsilon}$-apx. (Theorem 9)</td>
<td>no $</td>
<td>E</td>
<td>^{\frac{1}{2}-\epsilon}$-apx. (Theorem 9)</td>
</tr>
<tr>
<td><strong>S-RFP</strong>&lt;br&gt;(Sec. 4.3.3)</td>
<td>$?$</td>
<td>$?$</td>
<td>$\mathcal{P}$ (Theorem 12)</td>
<td>$\mathcal{P}$ (Theorem 12)</td>
<td>$\mathcal{P}$ (Theorem 12)</td>
<td>$?$</td>
</tr>
<tr>
<td><strong>U-RFP</strong>&lt;br&gt;(Sec. 4.3.3)</td>
<td>$\mathcal{NP}$-$\mathcal{C}$ (Theorem 10)</td>
<td>$\mathcal{NP}$-$\mathcal{C}$ (Theorem 10)</td>
<td>$\mathcal{NP}$-$\mathcal{C}$ (Theorem 10)</td>
<td>$\mathcal{NP}$-$\mathcal{C}$ (Theorem 10)</td>
<td>$\mathcal{NP}$-$\mathcal{C}$ (Theorem 10)</td>
<td>$\mathcal{NP}$-$\mathcal{C}$ (Theorem 10)</td>
</tr>
<tr>
<td></td>
<td>no $</td>
<td>E</td>
<td>^{\frac{1}{2}-\epsilon}$-apx. (Theorem 11)</td>
<td>no $</td>
<td>E</td>
<td>^{\frac{1}{2}-\epsilon}$-apx. (Theorem 11)</td>
</tr>
</tbody>
</table>

Table 4.3: Summary of the computational complexities of the problems
4.3.1 The Splittable Primary-restricted Restorable Flow Problem (S-PRFP)

We now define the *Splittable Primary-restricted Restorable Flow Problem (S-PRFP)* with respect to each recovery scheme. For simplicity, we assume throughout the work that only one primary LSP is established for each flow. However, the results of the work can be easily extended for the case where every flow has several primary LSPs. Let $G = (V, E)$ be a directed graph. Let $u_e$ be the bandwidth capacity of edge $e \in E$. Let $F \subseteq V \times V$ be a set of source-destination pairs representing traffic flow demands. For every traffic flow $f = (s_f, t_f) \in F$, let $s_f$ be the source node, $t_f$ the target node, $d_f$ the bandwidth demand, $P_f$ the sequence of edges along the primary LSP, and $w_f$ the profit for $f$. A feasible solution is one that admits some of the traffic flows into the network while meeting the edge capacity constraints. Each admitted flow is routed on its primary LSP and must be fully restorable in the face of any single link failure. Hence, for every admitted flow $f$ and edge $e \in P_f$, there must exist a set of backup LSPs that satisfies the constraints of the considered recovery scheme and can accommodate the admitted traffic of $f$ when $e$ fails. The objective is to maximize the total profit of the admitted traffic flows.

Note, the traffic demand of each admitted flow need not be fully satisfied. Moreover, following a failure, the admitted traffic of a flow may be split among several backup LSPs. Therefore, this splittable version of the problem is more applicable to the case where the network can technically split each flow into smaller sub-flows. A good example for this is a VoIP trunk between two media gateways. Such a trunk should carry thousands of low bandwidth VoIP calls at any given time. Hence, when necessary, the flow carried by this trunk can be divided into smaller sub-flows, each carrying only a portion of the traffic. Allowing multiple LSPs to back up a single primary LSP is especially attractive when no single backup LSP can accommodate the entire flow demand. However, multiple backup LSPs incur more signalling and state overhead.

**Theorem 7** S-PRFP is in $\mathcal{P}$ for all recovery schemes discussed in Section 4.1.

**Proof**
To show this, we formulate the problem as a linear program. We first present the constraints of the problem that are common to all recovery schemes. Then, we present additional constraints for each individual scheme. We define the following variables:

- $y_{f}^{e}$ – the fraction of $d_{f}$ routed over edge $e$ when edge $\bar{e}$ fails; when no edge fails, $\bar{e} = \phi$.
- $x_{f}$ – the total routed fraction of $d_{f}$.
The target function is to maximize the total gained profit:

\[
\text{Maximize } \sum_f w_f \cdot x_f
\]

subject to the following constraints:

\[
\begin{align*}
(C-1) & \quad \sum_{e=(u,v)} y_{fe}^e - \sum_{e=(v,u)} y_{fe}^e = \begin{cases} x_f & v = t_f \\ -x_f & v = s_f \\ 0 & \text{else} \end{cases} \\
\forall v \in V, \forall f \in F, \forall e \in \{E, \phi\} \\
(C-2) & \quad \sum_f d_f \cdot y_{fe}^e \leq u_e \quad \forall e \in E, \forall e \in \{E, \phi\} \\
(C-3) & \quad y_{fe}^e = 0 \quad \forall f \in F, \forall e \in P_f \\
(C-4) & \quad y_{fe}^e = 0 \quad \forall f \in F, \forall e \in E \\
(C-5) & \quad 0 \leq x_f \leq 1, \quad 0 \leq y_{fe}^e \leq 1 \quad \forall e \in E, \forall e \in \{E, \phi\}, \forall f \in F.
\end{align*}
\]

The set (C-1) of constraints ensures flow conservation. The set (C-2) ensures that no edge carries more than its capacity. The set (C-3) ensures that when no failure occurs, each flow is routed only along its primary LSP. The set (C-4) ensures that no flow is routed over a failed link. Finally, the set (C-5) of constraints ensures that the total routed bandwidth of each flow and the routed bandwidth on each backup LSP do not exceed flow demand.

The above constraints do not restrict the backup LSPs to be built according to a specific recovery scheme. The program selects, for every routed flow and for every failed link, a set of backup LSPs that can carry the flow's demand following a failure. Technically, according to these constraints, a flow may be diverted to a set of backup LSPs even if it is not routed through the failed edge. In fact, the above linear program solves \(|E|\) different instances of the splittable flow problem, one for each failed link. The only connection between these \(|E|\) instances is the requirement that the routed demand of each flow in every instance must be the same (i.e., \(x_f\)). It is clear that such a flexible “recovery scheme,” which has almost no restriction, would yield the best performance.

However, each of the recovery schemes presented in Section 4.1 imposes a set of additional constraints on the backup LSPs. The more constraints a recovery scheme imposes, the less flexible and efficient it is. We now present the specific set of constraints for each recovery scheme.

The specific set of constraints for the LR scheme is:

\[
\begin{align*}
(LR-1) & \quad y_{fe}^e \geq y_{fe}^\phi \quad \forall f \in F, \forall e \in E, \{e | e \in E, e \neq \bar{e} \text{ and } \bar{e} \text{ is not a downstream edge of } \bar{e} \text{ along } P_f\}.
\end{align*}
\]

The above set of constraints ensures that the backup LSP of \(f\) for \(\bar{e} = (u, v)\), assuming \(\bar{e} \in P_f\), will follow the primary LSP all the way from the source to \(u\). From node \(u\) to the destination node, the backup LSP is not constrained.
The specific set of constraints for the RLR scheme is:

\[
\text{(RLR-1) } y_{fe}^\phi \geq y_{fe}^\delta \quad \forall f \in F, \forall \bar{e} \in E, \{e | e \in E, e \neq \bar{e}\}.
\]

RLR-1 is similar to LR-1, except that it also ensures that if a backup LSP protects against a failure of edge \( \bar{e} = (u, v) \), it will follow the primary LSP not only from the source to \( u \) but also from \( v \) to the destination.

Since S-PRFP allows the traffic of the failed primary LSP to be split between several backup LSPs, RLR may use an unbounded number of backup LSP for each link failure. Hence, it is easy to see that an optimal solution for the FLR scheme and for the EkFLR scheme can be produced from an optimal solution for the RLR scheme. Hence, there is no need to specify special constraints for these two recovery schemes.

The specific set of constraints for the GR scheme is:

\[
\begin{align*}
\text{(GR-1) } y_{fe}^\phi & \begin{cases} 
= 0 & \forall \bar{e} \in E, \{f | f \in F, \bar{e} \in P_f\} \\
\geq y_{fe}^\delta & \text{otherwise}
\end{cases} \\
\text{(GR-2) } y_{fe}^\phi - y_{fe}^\delta & = \Delta y_{fe}^\phi & \forall f \in F, \forall \bar{e}, e \in E \\
\text{(GR-3) } \Delta y_{fe}^\phi & = \Delta y_{fe}^\phi & \forall f \in F, \forall \bar{e}, \bar{e}_1, \bar{e}_2 \in P_f \\
& & \text{where } \bar{e}_2 \text{ immediately follows } \bar{e}_1 \text{ on } P_f, \forall e \in E.
\end{align*}
\]

The set (GR-1) of constraints ensures that the backup LSPs of every flow whose primary LSP crosses the failed edge must be edge disjoint with the primary LSP. The set (GR-2) introduces auxiliary variables \( \Delta y_{fe}^\phi \). Each of these variables represents the difference between the bandwidth of \( f \) routed on the primary LSP, and the bandwidth of \( f \) to be routed on the backup LSPs that protect the flow against the failure of edge \( \bar{e} \). This difference yields a circular traffic flow that traverses the backup LSPs from the source to the destination and the primary LSP in the reverse direction. The set (GR-3) of constraints ensures that for each flow the same set of backup LSPs is used to protect all the edges along the primary LSP.

Finally, the set of specific constraints for the UR scheme is:

\[
\text{(UR-1) } y_{fe}^\phi \geq y_{fe}^\delta \quad \forall \bar{e}, e \in E, \forall f \in F, \bar{e} \notin P_f.
\]

This set ensures that if the failed link is not included in the primary LSP of a flow, then the backup LSP is identical to the primary LSP. Otherwise, the set of backup LSPs has no constraint.

From the above discussion it is obvious that RLR imposes more constraints than LR. Hence, we expect LR to perform better. In addition, it is clear the UR outperforms all the other recovery schemes. However, it is not clear from the above
discussion whether GR outperforms LR, or vice versa. On the one hand, LR is more restricted in that its backup LSPs must follow the route of the primary LSP all the way to the failed link, whereas the backup LSPs of GR must only avoid using the links of the primary LSP. On the other hand, GR requires the same set of backup LSPs to protect against all possible failures on the primary LSP, whereas LR imposes no such restriction.

4.3.2 The Unsplittable Primary-restricted Restorable Flow Problem (U-PRFP)

We now address the Unsplittable Primary-restricted Restorable Flow Problem (U-PRFP). There are two differences between U-PRFP and S-PRFP. First, in U-PRFP, a profit can be obtained for a flow only when its entire demand is satisfied. Second, in U-PRFP, the traffic of each flow can be restored using only a single backup LSP. We now address the computational complexity of U-PRFP. In the decision variant of this problem, we are given a number \( K \) and ask whether there exists a feasible solution that yields a profit equal to or larger than \( K \).

**Theorem 8** U-PRFP is \( \mathcal{NP} \)-complete for all recovery schemes.

This can be shown using a simple reduction from the knapsack problem [38].

**Theorem 9** U-PRFP for GR, LR, or UR schemes cannot be approximated within \( |E|^{1/2-\epsilon} \) unless \( \mathcal{P} = \mathcal{NP} \).

**Proof**

We start by showing that U-PRFP for GR, LR, or UR schemes can be reduced from the Unsplittable Flow Problem (U-FP) [41, 100]. An instance of U-FP contains \( G = (V, E) \), \( u_e \), \( F \), \( d_f \), \( w_f \), and \( K \), which are all the same as in U-PRFP. The target is to route a subset \( F' \subseteq F \) of the flows such that their demands are fully satisfied and the edge capacity constraints are met. Each routed flow must use a single path. The question is whether there exists such a subset \( F' \) whose total profit is greater than \( K \).

We consider an instance of U-FP and show a reduction to an U-PRFP instance for the GR scheme. In the following we add a superscript \( r \) to all the parameters constructed for U-PRFP. Let \( G^r = (V^r, E^r) \) be a directed graph with \( V^r = V \cup \{u_1, u_2\} \) and \( E^r = E \cup \{(v, u_1), (u_2, v) \mid v \in V\} \cup \{(u_1, u_2)\} \). For every \( e \in E \), let \( u^r_e = u_e \); otherwise, let \( u^r_e = \infty \). Figure 4.4 depicts \( G^r \).

Let \( F^r = F \). For each flow \( f \in F^r \), let \( d^r_f = d_f \) and \( w^r_f = w_f \). For every \( f = (s_f, t_f) \in F^r \), let \( P^r_f = (s_f, u_1, u_2, t_f) \). Namely, for every flow we construct a primary LSP that passes through the new vertices. Note that these paths do not pass
through the original edges of $G$. Finally, we define $K^r = K$. It is easy to see that this U-PRFP instance can be constructed in polynomial time.

In the constructed U-PRFP instance, the failed edge $(u_1, u_2)$ constrains the maximum profit for any feasible solution. It is easy to see that any feasible solution for U-FP is equivalent to a feasible backup routing for U-PRFP under such failure. Hence, the existence of a solution for U-FP with a profit of more than $K$ implies the existence of a backup routing for the U-PRFP instance with a profit of more than $K^r$, and vice versa.

A similar reduction can be used to show that U-PRFP is $\mathcal{NP}$-complete for the UR or LR schemes.

The above reduction is an L-reduction with the constants $\alpha = \beta = 1$ [84], because any feasible solution for a U-FP instance has the same value as for the constructed U-PRFP instance. In [84] it is shown that an L-reduction is an approximability preserving reduction. Hence, using the above reduction, every approximation algorithm for U-PRFP can be translated to an approximation algorithm for U-FP with the same approximation ratio. This implies that the best approximation ratio that can be achieved for U-PRFP is not worse than the best approximation ratio that can be achieved for U-FP. Since U-FP is $\mathcal{NP}$-hard to approximate within $|E|^{1/2-\epsilon}$ (see [41]), U-PRFP with GR, LR, or UR is also $\mathcal{NP}$-hard to approximate within $|E|^{1/2-\epsilon}$. □

### 4.3.3 The Unsplittable and Splittable Restorable Flow Problems (U-RFP and S-RFP)

We now address the Unsplittable Restorable Flow Problem (U-RFP) and the Splittable Restorable Flow Problem (S-RFP). Recall that the goal of these problems is to establish not only the backup, but also the original (primary) LSPs. U-RFP establishes one primary LSP for every flow, and one backup LSP for every failure event along
the selected primary LSP. A profit is obtained for an admitted flow only if its entire demand is satisfied. In contrast, S-RFP can split the traffic over several primary LSPs. Every edge along these LSPs can be protected by several backup LSPs. The demand of every flow can be partially satisfied, in which case only part of the profit is obtained.

**Theorem 10** U-RFP is $\mathcal{NP}$-complete for all recovery schemes discussed in Section 4.1.

This is a trivial consequence of Theorem 8.

**Theorem 11** U-RFP for GR, LR, or UR cannot be approximated within $|E|^{1/2-\epsilon}$ unless $\mathcal{P} = \mathcal{NP}$.

**Proof**
A similar reduction to the one presented in Theorem 9 for U-PRFP using GR, LR and UR can be used to reduce U-FP to U-RFP. The only difference is that the reduction does not set the primary LSPs for the flows. This is a suitable reduction, because if there exists a solution for U-FP with a profit greater than $K$, then there must exist a solution for U-RFP with a profit greater than $K^{r}$ that routes all primary LSPs over $(u_1, u_2)$ and all backup LSPs over the routes used in the U-FP solution. If there exists a solution for U-RFP that routes a set of flows whose profit is greater than $K^{r}$, there must exist a routing that does not go through $(u_1, u_2)$ for the same flows. Hence, there is a solution for U-FP with a profit greater than $K$.

The above reduction is also approximability preserving. Therefore, as for U-PRFP, this implies that U-RFP is $\mathcal{NP}$-hard to approximate within $|E|^{1/2-\epsilon}$.

**Theorem 12** S-RFP is in $\mathcal{P}$ for RLR, FLR, and EkFLR schemes.

**Proof sketch**
The linear program constraints for S-PRFP with RLR do not depend on the primary LSPs. Thus, we can use this linear program for S-RFP without the set of constraints (C-3) that restricts the primary LSPs. Hence, we get that S-RFP with RLR can be solved in polynomial time. As noted in Section 4.3.1, it can be easily shown that the optimal solutions for S-RFP with RLR, FLR and EkFLR are the same. This is because, in S-RFP, there is no bound on the number of backup LSPs.

Note that Theorem 12 improves the results presented in [13], where only an FPTAS was shown for S-RFP with RLR.

To formulate the S-RFP using the other recovery schemes (GR, LR, and UR) we need to use path-indexed variables, namely variables that indicate for each flow the routed bandwidth on every possible path in the graph. However, since the number of such paths is exponential in the size of the graph, it is not easy to solve this formulation in polynomial time. We therefore leave the complexity of S-RFP with GR, LR and UR open for future research.
4.4 Algorithms for U-PRFP and U-RFP

4.4.1 Approximation Algorithms

We now present approximation algorithms for U-PRFP and U-RFP for the case where edge capacities are all equal and the bandwidth demand of each individual flow does not exceed half of the edge capacities. While these assumptions are not always realistic, these algorithms have theoretical value due to their worst case performance guarantee. The presented algorithms guarantee an approximation ratio of $|E|^{1/2}$. The algorithms are similar to those presented in [100] for U-FP, but have a different analysis. The algorithms can be used with every recovery scheme, and are based on the following simple observation.

Observation 1 If the load on each edge does not exceed half of the edge capacity, following a single failure every flow can be fully restored using arbitrary backup LSPs.

We first present the algorithm for U-PRFP. The algorithm begins by solving a similar problem, called the Splittable Primary-restricted Flow Problem (S-PFP). In this problem each flow can only be routed along a primary LSP given in advance, and restorability is not considered. Moreover, in this problem a flow can be only partially satisfied. A linear program for the problem follows. The edge capacities are denoted by $u$. We use the same variables as defined in Section 4.3:

Maximize $\sum_f w_f \cdot x_f$
subject to the following constraints:

\begin{align}
(S\text{-PFP}-1) \quad & \forall v \in V, \forall f \in F \\
& \sum_{e=(u,v)} y_{fe}^\phi - \sum_{e=(v,u)} y_{fe}^\phi = \begin{cases} 
 x_f & v = t_f \\
 -x_f & v = s_f \\
 0 & \text{else}
\end{cases} \\
(S\text{-PFP}-2) \quad & \sum_f d_f \cdot y_{fe}^\phi \leq u \quad \forall e \in E \\
(S\text{-PFP}-3) \quad & y_{fe}^\phi = 0 \quad \forall f \in F, \forall e \notin P_f \\
(S\text{-PFP}-4) \quad & 0 \leq x_f \leq 1, \quad 0 \leq y_{fe}^\phi \leq 1 \quad \forall e \in E, \forall f \in F.
\end{align}

The approximation algorithm solves this linear program and applies a randomized rounding procedure that yields a solution for U-PRFP. Let $x_f^*$ be the value given by the optimal solution of the linear program to the variable $x_f$. The algorithm then routes each flow $f$ over the primary LSP with probability $x_f^*/\gamma$, where $\gamma$ is a constant larger than 1.

Theorem 13 By choosing a proper value for $\gamma$, there exists a deterministic algorithm, based on the above randomized one, which produces a solution in which the load imposed on every link does not exceed half of the link capacity, and the total profit is lower bounded by $\Omega(w^*_S,\text{PFP}/|E|^{1/2})$, where $w^*_S,\text{PFP}$ is the value of the optimal solution for S-PFP.
Proof
The proof is similar to the proof of the approximation ratio of $|E|^{1/2}$ for U-FP presented in [100] except that our aim is to bound the load on every edge to $1/2$ rather than $1$. We analyze the randomized rounding process in the approximation algorithm for U-PRFP. The purpose of this analysis is to show that by choosing an appropriate $\gamma$, the load imposed on every edge does exceed half the edge capacity, and the total profit is not less than $\Omega(w^*_E / |E|^{1/2})$.

Let $\hat{x}_f$ be a Bernoulli random variable for choosing flow $f$. The success probability of $\hat{x}_f$ is $x^* / \gamma$. The expected total load imposed on every link $e \in E$ by the end of this process is

$$E[L_e] = E[\sum_{i \in P_f} d_f \hat{x}_f] = \sum_{i \in P_f} d_f x^*_f / \gamma \leq u / \gamma.$$  \hspace{1cm} (4.1)

The inequality follows from constraints (S-FP-2) in the linear program. For the sake of simplicity, we scale the demands and capacities such that $u = 1$.

For every $e \in E$, let $\mathcal{E}_f$ denote the “bad event” that $L_e > 1/2$. We now construct well-behaved estimators for these events. Consider first the case, where $d_f \leq 1/4$ holds for every $f$. Define $d'_f = 4d_f$ and $L'_e = \sum_{i \in P_f} d'_f \hat{x}_f$ for all $e \in E$. From (4.1) it follows that $\mu \leq E[L'_e] \leq 4/\gamma$. From [100] we know that a possible well-behaved estimator for $\mathcal{E}_f$ is $\chi^1_e$:

$$\chi^1_e = \prod_{i \in P_f} (1 + \delta)^{d'_f \hat{x}_f} \over (1 + \delta)^{\mu (1 + \delta)},$$

where $\delta = 2/\mu - 1$. From [100] we also know that

$$E[\chi^1_e] \leq G(4/\gamma, \gamma/2 - 1),$$

where $G(\mu, \delta) \equiv \left( \frac{e^\delta}{(1 + \delta)^{1 + \delta}} \right)^\mu$.

Next, consider the case where $d_f \geq 1/4$ for every $f$. In this case, the event $\mathcal{E}_f$ holds, if and only if more than 2 flows whose primary LSP traverses $e$ are chosen. From [100] it can be shown that a well-behaved estimator for $\mathcal{E}_f$ is $\chi^2_e$:

$$\chi^2_e = \psi_2(\{\hat{x}_f | e \in P_f\}),$$

$$\Psi_2(z) = \sum_{1 \leq i < j \leq n} z_i z_j, \text{ for } (z_1, z_2, \ldots, z_n) \in \mathbb{R}^n$$

Since in this case Eq. (4.1) implies that $\sum_{i \in P_f} E[\hat{x}_f] \leq 4/\gamma$, $E[\chi^2_e] = 8/\gamma^2$, must hold.

Next, we construct a proper estimator for the bad event “$w(T) < w^*(T)/(2\gamma)$”, where $w(T) = \sum_{i \in T} w_f \hat{x}_f$, and $w^*(T) = \sum_{i \in T} w_f x^*_f$. We know that $\mu_1 \leq E[w(T)] = w^*(T)/\gamma$. From [100] follows that a proper estimator for the above bad event is

$$\chi_w(T) = \prod_{i \in T} (1 - \delta_1)^{w_f \hat{x}_f} \over (1 - \delta_1)^{\mu_1 (1 - \delta_1)}.$$
where $\mu_1(1 - \delta_1) = w^*(T)/(2\gamma)$. From [100], it is also known that

$$E[X_w] \leq H(w^*(T)/\gamma, 1/2)$$

where $H(\mu, \delta) = e^{-\mu\delta^2/2}$.

Before proceeding to the approximation ratio of the algorithm, we present the following theorem from [100]:

**Theorem 14** Let $E_1, E_2, \ldots, E_t$ be events and $r, s$ be non-negative integers with $r + s \leq t$ such that:

- $E_1, E_2, \ldots, E_r$ are all increasing, with respective well-behaved estimators $g_1, g_2, \ldots, g_r$;
- $E_{r+1}, \ldots, E_{r+s}$ are all decreasing, with respective well-behaved estimators $g_{r+1}, \ldots, g_{r+s}$;
- $E_{r+s+1}, \ldots, E_t$ are arbitrary events, with respective proper estimators $g_{r+s+1}, g_2, \ldots, g_t$;
- For every $i$, $E_i$ and $g_i$ are completely determined by $\bar{X}$.

Then, if

$$1 - \left(\prod_{i=1}^{r} (1 - E'[g_i])\right) + 1 - \left(\prod_{i=r+1}^{r+s} (1 - E'[g_i])\right) + \sum_{s=r+s+1}^{t} E'[g_i] < 1$$

holds, where $E'(-) = \min\{E(-), 1\}$, we can efficiently construct a deterministic assignment for $\bar{X}$ under which none of $E_1, E_2, \ldots, E_t$ holds. Note that: (a) empty products are taken to be 1; (b) If there is $g_i$ such that $E[g_i] > 1$, then the entire product is equal to 0.

We continue by showing the $\Omega((w^*)^2/|E|)$ bound. Let $I_0$ be the subset of flows for which traffic demand is at most $1/4$ and $I_1 = I/I_0$. We first assume that $w_{i_1}^1 \leq w_{i_0}^1$, i.e., $w_{i_0}^1 \geq w^*/2$. Since the events $E_{e}$ are increasing, avoiding the bad events would require

$$1 - \left(\prod_{e} (1 - E'[\chi^1_e])\right) + E'[\chi_w(I_0)] \leq 1.$$  \hspace{1cm} (4.2)

For a suitably large constant $c$, $\gamma = c|E|/w^*$ satisfies the inequality of Eq. (4.2), since:

$$E[\chi_w(I_0)] \leq H(w^*(I_0)/\gamma, 1/2) \leq H(w^*/(2\gamma), 1/2).$$

Thus, from Theorem 14 follows that if $w^*(I_1) \leq w^*(I_0)$, we can efficiently select feasible paths for $I_0$ with objective function value $\Omega((w^*(I_0))^2/|E|) = \Omega((w^*)^2/|E|)$.

Next, we consider the case where $w^*(I_1) \geq w^*(I_0)$. As in the previous case, we must have

$$1 - \left(\prod_{e} (1 - E'[\chi^2_e])\right) + E'[\chi_w(I_1)] \leq 1.$$  \hspace{1cm} (4.3)
if we are to avoid all the bad events. Again, for a suitably large constant \( c \), and from the same considerations given above, \( \gamma = c\|E\|/w^* \) satisfies the inequality of Eq. 4.3. Thus, from Theorem 14 follows that if \( w^*(I_1) \geq w^*(I_0) \), we can efficiently select feasible paths for \( I_1 \) with objective function value \( \Omega((w^*(I_1))^2/|E|) = \Omega((w^*)^2/|E|) \).

To conclude the proof we need to show the \( \Omega(w^*/\sqrt{|E|}) \) bound. If \( w^* \geq \sqrt{|E|} \), this bound immediately follows from the \( \Omega((w^*)^2/|E|) \) bound. If \( w^* < \sqrt{|E|} \), we can simply choose to admit a flow \( f \) for which \( w_f = 1 \).

By Observation 1, the flows routed by the algorithm can be fully restored in the face of a single link failure using any recovery scheme. It is obvious that \( w_{\text{S-PFP}}^* \) is greater than the optimal solution for S-PRFP with any recovery scheme. Hence, \( w_{\text{S-PFP}}^* \) is also greater than the optimal solutions for U-PRFP with any recovery scheme. We can therefore conclude that the algorithm can be used as an approximation algorithm for U-PRFP with an approximation ratio of \( O(|E|^{1/2}) \).

Next, we present an algorithm for U-RFP with an approximation ratio of \( O(|E|^{1/2}) \). The algorithm begins by solving the Splittable Flow Problem (S-FP). In this problem primary LSPs are not given in advance, and restorability is not considered. The linear program for this problem is the same as the one presented above for S-PFP, but without the set (S-PFP-3) of constraints. After finding an optimal solution for this linear program, every flow \( f \) is routed over a possibly empty set of \( \Gamma_f \) LSPs. Each LSP in \( \Gamma_f \) carries a fraction \( z_{fk}^* \) of the demand, where \( 1 \leq k \leq |\Gamma_f| \). The algorithm chooses to route the entire demand of flow \( f \) on the \( k \)-th LSP with probability \( z_{fk}^*/\gamma \), independently of the other LSPs in \( \Gamma_f \). If more than one LSP is chosen, only one is arbitrarily selected.

**Theorem 15** By choosing a proper value for \( \gamma \), there exists a deterministic algorithm, based on the above randomized one, which produces a solution in which the load imposed on every link does not exceed half of the link capacity, and the total profit is lower bounded by \( \Omega(w_{\text{S-PFP}}^*/|E|^{1/2}) \), where \( w_{\text{S-PFP}}^* \) is the value of the optimal solution of S-FP.

The proof is similar to the one presented for Theorem 13. For brevity reasons we will not repeat it.

As for U-PRFP, by Observation 1, the flows admitted by the algorithm can be fully restored in the face of a single link failure using any recovery scheme. It can also be shown that \( w_{\text{S-FP}}^* \) must be greater than the optimal value for U-RFP with any recovery scheme. Therefore, the algorithm guarantees an approximation ratio of \( O(|E|^{1/2}) \) for U-RFP.

### 4.4.2 Heuristics

The algorithms presented earlier for U-PRFP and U-RFP have theoretical value due to their worst case performance guarantee. However, we expect that their average
performance will not be satisfactory, since they only utilize up to half of the edge capacities. Moreover, since the algorithms do not depend on a specific recovery scheme, performance of the various schemes cannot be compared. Therefore, in this section we present heuristics that yield better average performance and allow us to compare the performance of the various recovery schemes. We only present here the heuristics for U-PRFP. However, they can be extended to U-RFP in a straightforward manner.

The first heuristic begins by solving the linear program presented in Section 4.4 for S-PFP. Let $x_f^*$ be the value given to the variable $x_f$ by the optimal solution of the linear program. We then sort the flows in a non-increasing order of $w_f/d_f$. Then, for each flow, we apply the randomized rounding procedure presented above for the U-PRFP approximation algorithm. If the flow is selected by the randomized rounding procedure, we verify that (a) the flow can be routed on its primary LSP without violating the capacity constraints, and (b) for the chosen recovery scheme, feasible backup LSPs that do not violate the capacity constraints also exist. If both conditions hold, the flow is admitted along with its backup LSPs. If there are several feasible backup LSPs, the shortest one is selected.

The second heuristic is based on a well-known algorithm for U-FP [8]. In the following discussion we refer to a backup LSP for edge $\bar{e}$ as the entire route from the source to the destination taken by the flow when $\bar{e}$ fails. For a flow $f$ and an LSP $P$ from $s_f$ to $t_f$, define

$$H(f, P) = \frac{w_f}{d_f \sum_{e \in P} 1/u(e)}$$

to indicate the ratio between the flow profit and cost for path $P$. In general, we would like to accept only flows that can be routed over a path whose $H$ value is greater than a given threshold. Given a set of flows, we determine a lower and an upper bound for $H$. The lower bound, denoted $H_{\text{min}}$, is $w_{\text{min}}/|V|$, where $w_{\text{min}}$ is the minimum profit for a flow. The upper bound, denoted $H_{\text{max}}$, is $w_{\text{max}} \cdot u_{\text{max}}/d_{\text{min}}$, where $w_{\text{max}}, u_{\text{max}}$ and $d_{\text{min}}$ are the maximum profit, maximum edge capacity, and minimum bandwidth demand, respectively. Let $L^f(e)$ be the current load on edge $e$ when $\bar{e}$ fails ($\bar{e} = \phi$ when no edge fails). We say that LSP $P$ is feasible for flow $f$ when edge $\bar{e}$ fails, if $\forall e \in P L^f(e) + \frac{f_e}{u(e)} \leq 1$ holds. We say that LSP $P$ is acceptable for flow $f$ with respect to $\alpha$, if $H(f, P) > \alpha$.

In general, the heuristic considers the flows sequentially. A flow is admitted only if

1. The primary LSP is feasible and acceptable with respect to a given threshold.
2. There are backup LSPs that protect the primary LSP, and these are also feasible and acceptable with respect to the given threshold.

The heuristic considers several values for the threshold, and chooses the best one. Denote by $RS$ the recovery scheme used. The heuristic works as follows:
• For each $k$ from $\lfloor \log H_{\min} \rfloor$ to $\lceil \log H_{\max} \rceil$,
  
  - Sort the flows in $F$ according to a non-increasing order of $w_f/d_f$.
  - For each $f \in F$, run $\text{Route}_{U-PRFP}(2^k, f, RS)$.

• Choose the best solution from those produced for every value of $k$.

The procedure $\text{Route}_{U-PRFP}(\alpha, f, RS)$ admits and routes the flow $f$ only if

1. $P_f$ is feasible and acceptable with respect to $\alpha$.

2. and for every $\bar{e} \in P_f$ there exists a backup LSP $P^\bar{e}$ that

  (La) protects against a failure of $\bar{e}$ according to the rules imposed by $RS$,
  (Lb) is acceptable with respect to $\alpha$,
  (Lc) is feasible when $\bar{e}$ fails.

In the specific case where $RS$ is FLR, the backup LSP for $\bar{e} = (u, v)$ must use the same route from $u$ to $v$ as a previously chosen backup LSPs for $\bar{e}$. The same rule is applicable also to EkFLR if no fewer than $k$ backup LSPs for $\bar{e}$ were already chosen.

4.5 Simulation Study

In this section we evaluate the performance of the algorithms presented in Sections 4.3 and 4.4 for the various recovery schemes. We use the BRITE simulator [77] to simulate MPLS domain topologies according to the Barabasi-Albert model [9]. This model captures two important characteristics of network topologies: incremental growth and preferential connectivity of a new label switch router (LSR) to well-connected existing LSRs. These characteristics yield a power-law degree distribution of the LSRs.

To validate our results, we also use actual ISP topologies, as inferred by the RocketFuel project [99]. The bandwidth for each link is based on the results reported in [74]. For each synthetic or real topology, we generate a set of flows according to a power-law distribution. A network topology and a set of flows form together one simulation instance. In the case of PRFP, the simulation instance also contains the primary LSP for each flow. For the primary LSP of each flow, we select the shortest path. For each simulation instance, we determine the backup LSPs using the following algorithms:

1. An optimal algorithm for S-PRFP that solves the linear program presented in Section 4.3.1. In the following, this algorithm is referred to as OPT-S-PRFP.
(a) Num. LSRS = 20, average degree = 3  
(b) Num. LSRS = 20, average degree = 5  
(c) Num. LSRS = 40, average degree = 3

Figure 4.5: Relative performance for OPT-S-PRFP for various MPLS domains
2. An optimal algorithm for S-PFP that solves the linear program presented in Section 4.4. In the following, this algorithm is referred to as OPT-S-PFP.

3. The first heuristic for U-PRFP presented in Section 4.4.2, based on the approximation algorithm for U-PRFP. In the following, this algorithm is referred to as U-PRFP-1.

4. The second heuristic for U-PRFP presented in Section 4.4.2, based on the approximation algorithm for U-PRFP. In the following, this algorithm is referred to as U-PRFP-2.

5. An optimal algorithm for S-RFP that solves the linear program presented in Section 4.3.3 (using RLR). In the following, this algorithm is referred to as OPT-S-RFP.

To solve the various linear programs, we use the Lp-Solve software [12].

We start by evaluating the various recovery schemes using the OPT-S-PRFP algorithm. Figure 4.5 depicts the results for three types of MPLS domains: (a) with 20 LSRs whose average node degree is 3 links; (b) with 20 LSRs whose average node degree is 5; and (c) with 40 LSRs whose average node degree is 3. To compare the performance of the various recovery schemes, we use a relative performance metric: the ratio between the profit of flows admitted by OPT-S-PRFP and the profit of flows that can be admitted when no backup LSPs have to be established (OPT-S-PFP). This relative performance metric indicates the “penalty” incurred by the restoration requirement. This metric is represented by the y-axis of all the graphs in Figure 4.5, while the offered load is represented by the x-axis. The value of the offered load is the average number of flows originated by each router.

As expected, it is evident from all three graphs that UR yields the best performance while RLR yields the worst. In addition, Figure 4.5 shows that GR yields higher profit than LR. However, the performance of UR, GR and LR differs only marginally (5% on the average), whereas RLR lags behind by about 15%. It is also evident that the performance differences for the domain with 20 LSRs and average degree of 5 are small (Figure 4.5(b)). This is attributed to the very short length (usually one or two links) of the primary LSPs, when established over shortest paths, in such MPLS domains. Consequently, the differences between the various recovery schemes cannot really be expressed. Since the domain with 40 LSRs and average degree of 3 (Figure 4.5(c)) has longer primary LSPs, the differences between the recovery schemes are much more visible. For instance, there is a difference of up to 20% between UR and RLR. In addition, in larger domains there are more backup LSPs to choose from. This further widens the gap between the more flexible and the less flexible schemes.

It is also evident from all the graphs in Figure 4.5 that the penalty associated with building backup LSPs increases as the offered load increases. This can be intuitively
explained by the fact that a more heavily loaded network requires that more flows be rejected in order for backup LSPs to be established. Moreover, the penalty of the backup LSPs is much higher for smaller domains, or for those with lower average degree. For example, the relative performance of UR in the domain with 20 LSRs and degree of 3 ranges between 0.67 and 0.85, where in the other domains it ranges between 0.73 and 0.95. This can be attributed to the fact that in the larger or denser domains the number of possible backup LSPs is higher, and the recovery scheme may be able to choose backup LSPs that allow more flows to be admitted.

Next, we evaluate the performance of the two heuristics for U-PRFP. In this case, we use a different relative performance metric: the ratio between the profit of flows admitted by the U-PRFP heuristic and the profit of flows admitted by OPT-S-PRFP using the same recovery scheme. This relative performance metric indicates the penalty incurred due to the inability to split the traffic following a failure. The metric is represented by the $y$-axis of the graph in Figure 4.6, while the $x$-axis indicates the recovery scheme used. Figure 4.6 depicts the results of the two heuristics for the various recovery schemes. Each column in the figure indicates the average performance over a range of offered loads. The results are for an MPLS domain of 20 LSRs whose average degree is 3. For the EkFLR scheme, we use $k = 2$.

Figure 4.6 suggests that the average performance of U-PRFP-1 is slightly better than that of U-PRFP-2. For almost all recovery schemes the heuristics exhibit excellent performance and yield roughly 80% of the optimal profit. This implies that the cost of using a recovery scheme with a single backup LSP rather than several is only about 20% of the profit. In particular, the relative performance of EkFLR with $k = 2$ is almost the same as the relative performance of RLR. Since, as explained in Section 4.3.1, both schemes have the same performance for OPT-S-PRFP, there is no real need to split the backup traffic into more than two LSPS. In contrast, the relative performance of FLR is only about 0.7. The higher penalty implies that one backup LSP per link may be too strict.

Next, we evaluate the performance of the two heuristics for different levels of offered load. Figure 4.7 depicts the average performance of U-PRFP-1 and U-PRFP-2 over all recovery schemes as a function of the offered load. The $y$-axis is the same relative performance as in Figure 4.6. The $x$-axis is the offered load. First, note that as the offered load increases, so does the performance difference between the two algorithms. Note also that algorithm U-PRFP-1 outperforms U-PRFP-2 by up to 7%. This can be attributed to the fact that U-PRFP-1 bases the decision whether to admit a flow on the optimal solution for S-PRFP, which takes into account all of the flows. In contrast, U-PRFP-2 bases the same decision on the current network condition only. It is also evident from Figure 4.7 that as the offered load increases, the penalty from the requirement not to split the flow decreases, and the performance of the heuristics approaches the performance of OPT-S-PRFP. This result is surprising, because we expect that in a highly loaded network, the ability to split the traffic of
each flow across several paths would contribute both flexibility and performance. Yet, this result can be explained by the fact that for all flows we use the shortest path as a primary LSP. Due to the power-law distribution of the flows in the network, the primary LSPs create bandwidth bottlenecks when the network becomes congested. This means that even though the optimal algorithm for S-PRFP might be able to back up more flows, these flows cannot be admitted because their primary LSP is saturated.

We now evaluate the penalty of using a single primary LSP for each flow. To this end we use the following relative performance metric: the ratio between the profit of flows that are admitted by OPT-S-PRFP and the profit of flows that are admitted by OPT-S-RFP using the RLR scheme. This relative performance metric indicates the penalty incurred when using a single primary LSP set in advance. This metric is represented by the $y$-axis of the graph in Figure 4.8, which depicts the results for two types of 20-LSR MPLS domains: one with average degree of 3 and another with average degree of 5. It is evident that as the offered load increases, so does the penalty for using a single primary LSP set in advance. This relation is not surprising since a highly loaded network requires the traffic to be split into several paths in order to maximize the admitted traffic. It is also evident that the penalty increases for a network with a higher average degree. This is because a higher network degree gives more options for splitting the traffic between two end nodes.

To validate the results from the synthetic graphs, we also used real AS topologies, inferred by the RocketFuel project [99]. The results for these simulations are shown in
Figure 4.7: Relative performance of the heuristics for U-PRFP as a function of the offered load

Figure 4.9. The topology studied in Figure 4.9 is of the Exodus ISP, which consists of 80 routers and 147 links (average degree is 1.8). In Figure 4.9 we compare the recovery schemes using the following relative performance metric: the ratio between the profit of flows admitted by U-PRFP-1 and the profit of flows that can be admitted when no backup LSPs have to be established. This metric, represented by the y-axis of the graph in Figure 4.9, reveals the penalty incurred by both the restoration requirement and the use of a single backup LSP. It is evident that the comparative performance of the various recovery schemes is similar to what we found in the synthetic graphs (see Figures 4.5 and 4.6). Namely, GR outperforms LR, and EkFLR (with k = 2) is close to RLR.

To summarize, the main conclusions we draw from the simulation study are:

- The performance differences between UR, GR, and LR are only marginal (5% on the average) while RLR is considerably worse. Hence, LR should be the recovery scheme of choice due to its short restoration time.

- If low administrative overhead is the main goal, EkFLR with k=2 should be preferred over RLR.

- Heuristics U-PRFP-1 and U-PRFP-2 achieve close to optimal profit.

- In congested networks, U-PRFP-1 outperforms U-PRFP-2.
• When the primary LSPs are set in advance in congested networks, splitting the backup LSPs yields only a small added profit (less than 10%).

• In non-congested networks, the added profit is small for joint optimization of primary and backup LSPs (less than 10%).

4.6 Node Failure

In this section we extend the results of the work to the case of node failures. A node failure causes the failure of all edges incident to it. In particular, a failure of a single node may take down two links along an LSP.

4.6.1 The Splittable Problems (S-PRFP and S-RFP)

S-PRFP for node failures is in $\mathcal{P}$. To show this we change the linear program for link failures presented in Section 4.3.1 in the following ways. First, the variable $y_{fe}^0$ indicates the fraction of demand of flow $f$ routed over edge $e$ after a failure of node $\bar{v}$. When no node fails, $\bar{v} = \emptyset$. The constraints presented in Section 4.3.1 are changed
as follows:

(C-1) \[ \sum_{e=(u,v)} y_{fe}^v - \sum_{e=(v,u)} y_{fe}^u = \begin{cases} x_f & \text{if } v = t_f \\ -x_f & \text{if } v = s_f \\ 0 & \text{otherwise} \end{cases} \]

\( \forall v \in V, \forall f \in F, \forall \bar{v} \in \{V, \phi\} \)

(C-2) \[ \sum_f d_f \cdot y_{fe}^v \leq u_e \]

\( \forall e \in E, \forall \bar{v} \in \{V, \phi\} \)

(C-3) \[ y_{fe}^\phi = 0 \]

\( \forall f \in F, \forall \bar{v} \notin P_f \)

(C-4) \[ y_{fe}^\bar{v} = 0 \]

\( \forall f \in F, \forall \bar{v} \in V, \bar{v} \text{ incident to } \bar{v} \)

(C-5) \[ 0 \leq x_f \leq 1, \ 0 \leq y_{fe}^\bar{v} \leq 1 \]

\( \forall e \in E, \forall \bar{v} \in \{V, \phi\}, \forall f \in F \)

(LR-1) \[ y_{fe}^\bar{v} \geq y_{fe}^\phi \]

\( \forall f \in F, \forall \bar{v} \in V, \{e|e \in E, e \text{ is not downstream} \}

upstream neighbour of \( \bar{v} \) along \( P_f \)\}

(RLR-1) \[ y_{fe}^\bar{v} \geq y_{fe}^\phi \]

\( \forall f \in F, \forall \bar{v} \in V, \{e|e \in E, e \text{ is not incident to } \bar{v} \} \)
As for link failures, S-RFP for the RLR scheme can be solved using the constraints (C-1), (C-2), (C-4), (C-5), and (RLR-1).

4.6.2 The Unsplittable Problems (U-PRFP and U-RFP)

U-PRFP and U-RFP are $\mathcal{NP}$-complete. This can be shown using the same reduction presented in Section 4.3.2 from U-FP. The maximal profit in the constructed instance of U-PRFP is upper bounded by the maximal profit when one of the nodes added by the reduction fails. This profit is bounded by the maximal profit for the U-FP instance, and vice versa. The reduction is also approximability preserving, implying that, in the case of node failures, U-PRFP and U-RFP cannot be approximated within $|E|^{1/2−\epsilon}$.

Assume that all edge capacities are equal to $u$. Let us define $r_{\text{max}} = \max\{r_{\text{in}}^{\text{max}}, r_{\text{out}}^{\text{max}}\}$, where $r_{\text{in}}^{\text{max}}$ and $r_{\text{out}}^{\text{max}}$ are the maximum in-degree and out-degree in $G$, respectively.

Observation 2 If the load on each edge does not exceed $u/(r_{\text{max}} + 1)$, then each flow can be fully restored using arbitrary backup LSPs.

This observation follows from the fact that if a node fails, then the total amount of bandwidth that must be restored does not exceed $u \cdot r_{\text{max}}/(r_{\text{max}} + 1)$. This amount can be fully routed on any edge. Hence, any arbitrary backup LSP can be used to restore the traffic.

An approximation algorithm very similar to the one presented in Section 4.4 can be adapted for node failures. However, here we assume that the bandwidth demand of each flow does not exceed $u/(r_{\text{max}} + 1)$. Using analysis similar to the one presented in the proof of Theorem 13, it can be shown that the load imposed by the algorithm does not exceed $u/(r_{\text{max}} + 1)$ and its maximum throughput is not lower than $\Omega(w^*/|E|^{1/2})$. 

95
4.7 Extension to MPLS over WDM lightpaths

We show here only how to extend the linear program of Section 4.3.1. The extension of the other linear programs is similar. When a flow enters a lightpath, it can leave it only at the lightpath’s egress node. Let $l_i$ denote the set of (physical) links composing lightpath $i$, and $y^e_{fe}$ denote the part of $y^f_e$ traffic that traverses lightpath $i$. The following constraints should be added to the linear program:

(O-1) $\sum_i y^e_{fe} = y^f_e$ \quad $\forall f \in F, \forall e \in E$

(O-2) $y^e_{fe_1} = y^e_{fe_2}$ \quad $\forall f \in F, \forall e \in E, \forall i, \forall e_1, e_2 \in l_i$

(O-3) $y^e_{fe} = 0$ \quad $\forall f \in F, \forall e \in E, \forall i, \forall e \notin l_i$

Constraints (O-1) ensure that the sum of the traffic carried over all the lightpaths on a certain link is equal to the total traffic traversing this link. Constraints (O-2) ensure that the amount of traffic carried by a lightpath is equal on all the links composing this lightpath. Finally, constraints (O-3) ensure that no traffic traversing a lightpath is carried over links that are not included in that lightpath.

4.8 Conclusions

We presented the first comprehensive study of maximizing restorable throughput in MPLS networks. We focused on the establishment of backup LSPs when the primary LSPs are already set. We showed that the splittable version of the problem is in $\mathcal{P}$ while the unsplittable version is $\mathcal{NP}$-complete and cannot be approximated within $|E|^{1/2 - \epsilon}$. We gave an algorithm with an approximation ratio of $|E|^{1/2}$ for the case where the bandwidth demand of an individual flow does not exceed half of the edge capacities. We developed two practical and efficient heuristics that were shown to achieve excellent performance. Using simulation, we compared the performance of the various MPLS recovery schemes. We showed that LR should be the scheme of choice since it has the fastest restoration time and almost the same performance as the best (UR) scheme. In addition, we showed that if reducing the administrative cost is the main concern, EkFLR with $k = 2$ should be the recovery scheme of choice.
Chapter 5

Conclusions

In this thesis we have investigated the traffic engineering in IP and MPLS networks. In its first part we studied the effectiveness of the N-hub Shortest-Path Routing concept in IP networks. We have demonstrated that this concept offers an excellent compromise between the simplicity of shortest-path routing and the efficiency of virtual circuit routing. We applied this concept to the problem of minimizing the maximum load in the network. We defined the corresponding optimization problem, and proved that it is NP-Complete even for $N = 1$. We also showed that it cannot be approximated within $2 - \epsilon$ for $\epsilon > 0$. We presented approximation algorithms for both the online and the offline versions of N-hub. Using simulations we showed that the performance of N-hub Shortest-Path Routing is very close to the performance of a hypothetical optimal algorithm that splits the traffic of the same flow among multiple routes. In addition, the simulations showed the effect of $N$ on the performance of N-hub is very small. Hence, even the performance of 1-hub is very close to optimal. We therefore conclude that N-hub Shortest-Path Routing, and in particular the $N = 1$ version, should be considered as a powerful mechanism for future datagram routing in the Internet.

In the second part of the thesis we have applied the N-hub Shortest-Path Routing concept to the traffic of network services. We proposed a novel approach for addressing the problems of placement and selection of service gateways. Rather than considering the need to route traffic to its destination through the gateways as a burden that has to be minimized, we take advantage of this need for the sake of traffic engineering. We translated the problems of gateway placement and selection to optimization traffic engineering problems whose objective is to maximize the admitted throughput. In this context, both problems are NP-complete. We have presented for both problems algorithms with very good performance. Our main conclusion in this part is that placement and selection of network services can be employed as an effective tool for traffic engineering.
In the third part of the thesis we have addressed the problem of throughput maximization in restorable MPLS networks. We focused on the establishment of backup LSPs when the primary LSPs are already set. We showed that the splittable version of the problem is in $\mathcal{P}$ while the unsplittable version is $\mathcal{NP}$-complete and cannot be approximated within $|E|^{1/2-\epsilon}$. We gave an algorithm with an approximation ratio of $|E|^{1/2}$ for the case where the bandwidth demand of an individual flow does not exceed half of the edge capacities. We developed two practical and efficient heuristics that were shown to achieve excellent performance. Using simulation, we compared the performance of the various MPLS recovery schemes. We showed that LR should be the scheme of choice since it has the fastest restoration time and almost the same performance as the best (UR) scheme. In addition, we showed that if reducing the administrative cost is the main concern, EkFLR with $k = 2$ should be the recovery scheme of choice.
References


105


שלנו היא בן שני מראים כיvim של התיאור לשחרים צומת לשכלו ברוחך י会展ון כליל.

יעיל הפוסט בחודש העברת.

החלק השלישית של העובדה ועסקת בניוון התיאורもらう לשחרים צומת הפרוטוקולים מיתוג מובסס תוריות (MPLS). הפרוטוקולים של בניוון אלה עוגן בחומpatibilityuncan המגנו של עוגני התיאור בדואות גנבים. התייחסוגנים שחלים חוסני הנגבי שלחמי מكلف מארז, לפני שהשלוחות ברוח מתרחש. רבי המתחדת עם הח DISCLAIM של נייקב ראייניים ושלי נייקב גרב מצומק בדוע ההשכלה (SPARE CAPACITY ALLOCATION) (Spare Capacity Allocation) המתחדתו העניקה עלי נייקב הגיבוי וייתרו מהفئה לפני שהכישוף של נתיבי הגיבוי על רוב המחקרים מתרחש. ראשיי התמקדות במזער ממקומיו מדיום. זה חפץ מועיל לבלתי רשת, אשו ברוחות פורת בצלת קיובית יבלוע ובחרונות למקסס את מכונותיו. מובזדה זה אפי מארג ממקס ממקס במקס ממקס התעבורה להינוגנת הנגד Büyük小さיתון. אפי מארג ואת האלגוריתמים הרשוייםᴛווןת של סיבוכיות ומקס תפונחות עובר הגורש השיבירת של העבויות. ברכז גול.slug העבירות יבלוע מהספנועתה מקומיו לשלי נייקב גרב. עובר הגורש הלא-שיבי, בכת כל נייקב יקבל hypoth בודק והיתו ממקס עלי נייקב גרב, אשו מתנוגס נתיוות להיקל והח 页面 שיתוף עם הניסיון בבלוע מתנוגס למקס.
המתקפלת היא בעיה נP - קשה לקירוב. אנו מציעים אלגוריתמים
עלילים לפתרון של ה問題ה, עם במקרא המקרה של למקרה
המקסימום, כאשר בפיתוח הנקרא-Benziner, מתוך
סימולציה מדגם, אשר ביפוריה באמצעות כificant
לנות בו, ויכולה תואם לטייס או עילג החשש, עם תוכו שלימוד של ויתרה
ורס תעבורה. על כן, אנו מציעים את הפיתוש ויתרה ומכונה עתידי
ל poner את עיקולון.
ברית השתי.

בחלקה השני של ענודו העוסק בتوفر של פיתוש הנקרא-Benziner ויתרה
רכוזת 열עבורה המקבלת שירתו לא. שירתו לשפסוקassy כדי שיתור לשיפור
שדרוך העבורה המפגוע. תוצאות לשיפורים הלא ה: הפרת מוניות, החסום
otent מינו, (caching), סינון עבורה פוגעים יכדם. עבורה קימון השיפוט במקים
התשיט הנקרא-Benziner, ביעורו או הפיכים עובריי דרכ שיתור
עבורה יכדו את השישום החשוב במשבבירה, על לשפתו
לשמימה של חינוך אלי על פיתויו החשש באופני כל. כי אם בינו
יף מתמטיקה בחר בידיעת לשת מחלק אקר, כך שחקל מהתעבורה
יירדה דרמה הנקרא-Benziner. לא מציעים בשיש החישיב לשיפוט השיתור, צור לקחת
בוחן של שיקל התערובת. במוקס בחשיבות או החשיבות המוביעות ייעי
העבורה, היא ומצעים מחשבים של יבדיל לשפר את הביצוע של בראש בשכל. השיפור
מושג עיון פיזור עמש העבורה הבור באיתו, המחו והבשרת
השתנייה היא בהישג המוקס בחר בוחר בלך שירת, והשנייה היא בחרת הששת
העבורה לבל וס תעבורה. העבורה בצאירו טיסות עם מקימם
משPers סגנוז של שירתו לא. אנו מציעים אלגוריתמים של ויתרה ויתרה
לעיבת התערובת לא זוגי והעבירה. סימולציה מדגם, ביפוריה thừaיה
ועובד מזא ומוחה שיפר ממעוי אלגוריתמים קימם. התאמה עיקורית
תקציר

הנדסת תעבורה היא אכף פעולות, אשר עוסקת בחישוב והערכה וה’utilизация של יוצגי רישה. יוצגי רישה הם רשתות מבנה מרכז משושה, אבדן ביצועים ויציע, על ידי תיאור לתעבורה, מרכז מתן הצעות, חומוט מהשה, אנצף פיקוד וה.’ סיווג של רשת ובו תכנית הדוגמת ב’autres פעולות השהיית השתייה, דרכ רובוט ויוצגי יוצגי רישה מכוניות אוניות וה’ רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip這樣的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip這樣的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip這樣的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip這樣的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip這樣的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip這樣的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip這樣的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip這樣的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip這樣的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip這樣的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip這樣的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip這樣的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. לדוגמה Ра isip有这样的ו רשתות מבנה מרכז משושה הם רשתות מבנה מרכז משושה. ל례
המfortunate הענה בהנחיית פروف' ראובן כה בפקולטה למדעי המחשב.

ברצוני להודות Lumpur של, ראובן, על ההכון והע른ה התוכנות שנועדו ל북ה
העבורה על המfortunate ראובן, וג'㉢itta זמצזת לעבד אתך.
אני מודה למשפחת שלל על העורר והחומכות ביлярץ' כל הש進め, לכל האמזונה ב.
תודה מוחצת לשרי של, אשת, אשר הענה עלידי העדדאת אתבי בברגעי מיויאשם,
ואשר הענה מנכה לש yalף עצי ארוכות בין במקהל תקינה ושכוניות רבי של
המאמרים.

אני מודה לטעני על התמיכה התכנית וה닫ה במחולום.
MPLS והנדסת תעבורה ברשתות ובכטסטוט ו- IP

ת밖에 על מחקר

לשם מילוי חלקי של הדרישות לקבלת תואר דוקטור לפילוסופיה

גדע נקבל

הונג לסנט הטכניון - מכון טכנולוגי לישראל
כיסל ה〒 lc  הפתה
דצמבר 2007
הנדסת תעבורה ברשתות מבוססות MPLS ו- IP

גבי נכסלי