UNIVERSITY OF CALIFORNIA, SAN DIEGO

A New Network Architecture for Future Optical Networks: Coarse Optical Circuit Switching By Default, Rerouting Over Circuits for Adaptation

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science and Engineering

by

Jerry Chou

Committee in charge:

Professor Bill Lin, Chair
Professor Geoffrey M. Voelker, Co-Chair
Professor George Papen
Professor Joseph Pasquale
Professor Stefan Savage

2009

The dissertation of Jerry Chou is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____
Co-Chair

_____
Chair

University of California, San Diego

2009

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

It has been a quite long and challenged journey in the past five years of my pursuit of PhD degree in UCSD. However, I cannot achieve it alone without the help from many others. To name a few, I would like to thank the following people.

First, I want to thank my advisor, Professor Bill Lin, who always gave me guidance, advise, and inspiration. Not only he is dedicated and responsible to research, he is also friendly and thoughtful to students. I consider myself be very lucky to have him as my advisor when I was confused and puzzled on the road of research. I particularly appreciate his patient, and I am impressed by his knowledge and smartness. Without his push, motivate and encourage, I really couldn't reach such proud achievement. I would also like to thank my mentor, Subhabrata Sen and Oliver Spatscheck, during my AT&T summer intern. They inspired several key ideas in the thesis and advised on my writing and experiments.

Thanks to my proposal and dissertation committee members, Professor Joseph Pasquale, Stefan Savage, George Papen and Geoffrey Voelker, who is also my co-advisor in CSE department, for their reviews and advises of this thesis. I also thank my former advisor Professor Andrew Chien's director in my first two difficult years. He gave me support both financially and mentally to adjust in a new environment. It was a pleasure to work with him and learned his unique vision and enthusiasm for research.

Thanks to all my colleagues and friends from my research labs, Chia-wei Chang, Hao Wang, Rohit Ramanujam, Shan Yan, Yang-Suk Kee, Nut Taesombut, Dionysios Logothetis and Ryo Sugihara. They have been great assets of my life and research.

To my parents, Hwai-Pwu Chou and Shi-Chie Fuh, I thank them for everything. Their support on my education and life never be less no matter where and how I am.

Finally, I would like to give my deepest appreciation to my dear wife, Ming-Yi Lee. There is nothing for comparison when you have someone who can listen to you, share feeling with you and care about you at any moment of your life. Because of her, I have the privilege to return a warm and lovely home after school.

Because of her, I can quickly recompose and refocus myself after all the pressure, frustration or worry from work. She is all the reasons that I consider my long and bumpy study journey is full of happiness and excitement. Needless to say that it has also been a particularly hard for her to live far away from her hometown, parents and friends and to find jobs in an unfamiliar foreign country. Thus, at the end, I really want to thanks for her consistent and endless support, encouragement, accompany, understanding and mostly love.

Chapter 1, in part, is a reprint of the material as it appears Journal of Optical Communications and Networking, 2009. Chou, Jerry; Lin, Bill, IEEE OSA Press, 2009. The dissertation author was the primary investigator and author of this paper.

Chapter 2, in full, is currently being prepared for submission for publication of the material. Chou, Jerry; Lin, Bill. The dissertation author was the primary investigator and author of this material.

Chapter 5, in full, is a reprint of the material as it appears in Transaction on Networking. Chou, Jerry; Lin, Bill; Sen, Subhabrata; Spatscheck, Oliver, IEEE/ACM Press, 2009. The dissertation author was the primary investigator and author of this paper.

VITA

2000-2003	B. S. in Computer Science , National Tsing Hua University, Taiwan
2003-2004	M. S. in Computer Science , National Tsing Hua University, Taiwan
2003-2004	Graduate Teaching Assistant, National Tsing Hua University, Taiwan
2006	Summer Intern, AT&T Labs Research, Florham Park, New Jersey
2007-2008	Graduate Teaching Assistant, University of California, San Diego
2004-2009	Ph. D. in Computer Science and Engineering, University of California, San Diego

PUBLICATIONS

Jerry Chou, Bill Lin, Subhabrata Sen and Oliver Spatscheck, "Proactive Surge Protection: A Defense Mechanism for Bandwidth-Based Attacks," in *IEEE/ACM Transaction on Networking (ToN)*, 2009.

Jerry Chou and Bill Lin, "Coarse Optical Circuit Switching By Default, Re-Routing Over Circuit For Adaptation," in *IEEE OSA Journal of Optical Communications and Networking*, Vol. 8, Iss. 1, pages 33-50 , Jan 2009.

Jerry Chou and Bill Lin, "Utility Max-Min Fair Bandwidth Allocation under Multi-path Routing," in *IEEE International Workshop on Quality of Service (IWQoS)*, 2009.

Jerry Chou, Bill Lin, Subhabrata Sen and Oliver Spatscheck, "Proactive Surge Protection: A Defense Mechanism for Bandwidth-Based Attacks," in *17th USENIX Security Symposium*, pages 123-138, Aug 2008.

Jerry Chou, Bill Lin, Subhabrata Sen and Oliver Spatscheck, "Minimizing Collateral Damage by Proactive Surge Protection," in *ACM SIGCOMM Workshop on Large Scale Attack Defense (LSAD)*, pages 97-104, Aug 2007.

ABSTRACT OF THE DISSERTATION

A New Network Architecture for Future Optical Networks:
Coarse Optical Circuit Switching By Default,
Rerouting Over Circuits for Adaptation

by

Jerry Chou
Doctor of Philosophy in Computer Science and Engineering

University of California San Diego, 2009

Professor Bill Lin, Chair

Professor Geoffrey M. Voelker, Co-Chair

As Internet traffic continues to grow unabated at an exponential rate, it is unclear whether or not the existing packet routing network architecture based on electronic routers will continue to scale at the necessary pace. On the other hand, optical fiber and switching elements have demonstrated an abundance of capacity that appears to be unmatched by electronic routers. In particular, the simplicity of circuit switching makes it well-suited for optical implementations. Therefore, given the rapidly increasing traffic and optical transport capabilities, and the growing disparity between optical capabilities and Moore's Law, we would like to bridge this gap for future optical networks.

In this thesis, we present a new approach to optical networking based on a paradigm of "coarse optical circuit switching by default" and "adaptive rerouting over circuits with spare capacity". We consider the provisioning of long-duration quasi-static optical circuits between edge routers at the boundary of the network to carry the traffic by default. When the provisioned circuit is inadequate, excess traffic demand is rerouted through circuits with spare capacity. In particular, by adaptively load-balancing across circuits with spare capacity, excess traffic is routed to their final destinations without the need to create circuits "on-the-fly". We

call this new network architecture COPLAR, which stands for "[C]oarse [OP]tica[L] circuit switching with [A]daptive [R]erouting".

Specifically, we first formulate the problem of circuit provisioning as a multipath utility max-min bandwidth allocation problem, which considers routing as an optimization parameter rather than input. The optimal and local algorithms proposed in the thesis are the first solutions to this problem. For traffic that cannot be handled by the default provisioned circuits, we apply an adaptive routing algorithm based on game theory to adaptively reroute the excess traffic over circuits with spare capacities. To evaluate COPLAR, we conducted extensive experiments using two separate real large-ISP PoP (point of presence)-level topologies, Abilene [2] and GEANT [44]. The results show that coplar has the ability to improve network throughput while significantly reducing electronic router overhead and the number of O/E/O conversions in the network.

Finally, we show that our utility max-min bandwidth allocation algorithm can be extended to other network problems. In particular, we present a network security application called PSP (Proactive Surge Protection) that provides a defense against bandwidth-based distributed denial-of-service attacks.

Chapter 1

Introduction

1.1 Motivation

The Internet has become the main conduit for virtually all wide-area data communications as it continues its phenomenal growth in traffic volumes and reach, extending into telephony and television broadcast services that were once only transported in the domain of dedicated networks. For the past decade, Internet traffic has been doubling nearly every year, and there is no indication that this rate of growth will decelerate in the near future. While the packet switching approach used in the Internet backbone networks has thus far been able to keep up, it is unclear whether electronic routers that have been used at the core of backbone networks will continue to scale to match future traffic growth or optical link rates [25].

On the other hand, optical fiber and switching elements have demonstrated an abundance of capacity that appears to be unmatched by electronic routers. The rate of increase in optical transport capacity has been keeping pace with traffic growth (with 100 Gb/s per wavelength in the next generation). Thus, one possible way of keeping pace with future traffic demands is to build an all-optical backbone network. However, packet switching requires the buffering and processing of packets, of which optical switches are not capable today, and it is unclear if these functions can be practically realized in optics. In contrast, circuit switching has much a simpler data transport, making it well-suited to optics and

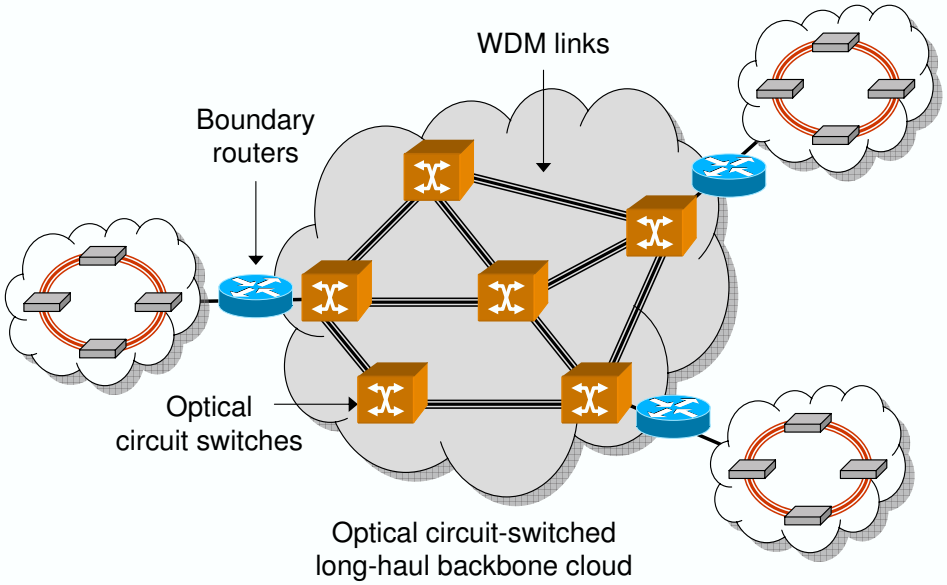


Figure 1.1: Optical circuit-switched cloud with boundary routers.

its vast capacity potential.

To harness the huge capacity of optical circuit switching in an evolutionary way that is compatible with packet switching at the edge of the network, transparent to the user, a number of candidate optical network data transport architectures have been proposed [70, 97, 65, 64, 101, 54, 53, 91, 17, 98, 21]. From a bird's-eye view, these architectures all share a similar conceptual starting point in which the core of the network is an all-optical circuit-switched cloud, as depicted in Figure 1.1. The optical circuit-switched cloud is comprised of long-haul DWDM links that are interconnected by optical cross-connects (OXC). Traffic traverses the circuit-switched cloud through pre-established circuits (lightpaths) at optical speeds. Boundary routers at the edge of the circuit-switched cloud provide a compatible packet switching interface to the rest of the Internet. The different proposed optical network data transport architectures differ in how they adapt to changing traffic conditions and the corresponding requirements on the granularity of circuits and the frequency of changes to the circuit configurations.

Several approaches are based on frequent changes to circuit configurations [70, 97, 65, 91, 64]. For example, in optical burst switching (OBS) [70, 97],

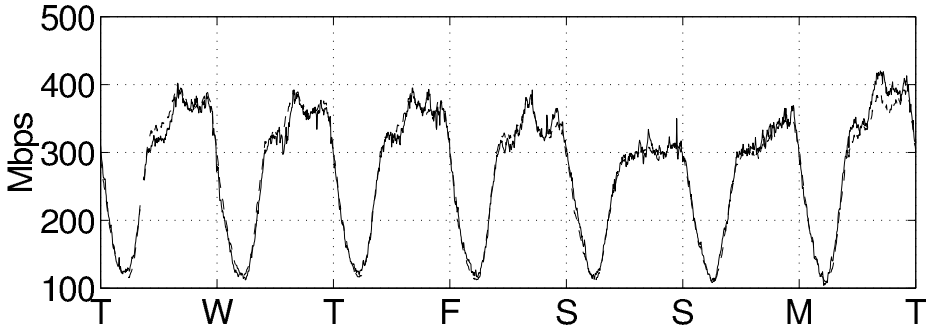


Figure 1.2: Aggregate traffic on a tier-1 US backbone link [77].

bursts of data are aggregated at the network edge by the boundary routers, and an out-of-band signaling process is used for establishing temporary circuits across the optical circuit-switched cloud for each burst. OBS adapts to changing traffic conditions by changing the circuit configurations on a frequent time-scale (i.e., at each data burst). In TCP switching [65], the detection of a new application (TCP) flow triggers the creation of its own new circuit. TCP switching adapts to changing traffic conditions by frequent creations of new fine-grained circuits. A dynamic coarse circuit switching scheme has also been proposed (Ch. 5 of [64]) in which a coarse circuit is established between each pair of boundary routers for carrying traffic with the same pair of ingress-egress (IE) nodes. This dynamic coarse circuit switching approach adapts to changing traffic conditions by frequently adjusting the circuit configurations on relatively short time-scales based on an online traffic estimation mechanism. Although the frequency of dynamic circuit reconfigurations imposed by the above approaches, on the order of tens of microseconds to a second, is well within the capabilities of available optical switching technologies, the coordination of such frequent network-wide reconfigurations is not easy. Moreover, new signaling mechanisms and (electronic) control planes are required to facilitate the coordination.

1.2 Approach Overview

In this thesis, we introduce COPLAR, a novel optical networking design based on a new paradigm of "coarse optical circuit switching by default", and "adaptive rerouting of excess traffic over circuits with spare capacity" when necessary. COPLAR stands for "[C]oarse [OP]tica[L] circuit switching with [A]daptive [R]erouting"[1]. COPLAR exploits in part the observation that future traffic conditions can be predicted offline using past observations. Previous studies [77, 61] have shown that the aggregate traffic at the core of the network tends to be very smooth and that it follows strong diurnal patterns that are easy to characterize. Figure 1.2 shows the total aggregate traffic on a backbone link in a tier-1 US ISP backbone network [77]. As can be observed in the figure, the aggregate traffic varies over time in a regular and predictable way. Such diurnal traffic observations over repeated data sets suggest that coarse circuits could be provisioned to handle the expected traffic. Indeed, we make use of past traffic measurements to pre-compute offline such coarse circuit configurations. As we shall see in our extensive evaluations in Chapter 4, careful offline selection of coarse circuit configurations that take into account the statistical daily traffic variations over different hours observed in past measurements can produce coarse circuits that can accommodate the actual traffic most of the time. Therefore, in our approach, traffic is sent "by default" directly over the coarse optical circuit provisioned between the corresponding pair of boundary routers.

However, our approach also provides a mechanism for adapting to scenarios in which the actual traffic differs from prediction or when there are sudden unexpected changes in traffic (e.g., due to external events). In particular, our solution uses an adaptive routing algorithm based on game theory for rerouting (hopefully small amounts of) excess traffic over circuits that have spare capacity when the provisioned circuit provides insufficient capacity. By adaptively routing across circuits with spare capacity, excess traffic is routed to their final destinations without the need to create new circuits "on-the-fly" – additional capacity needs

[1] COPLAR is pronounced the same as the word "copular", which is the adjective form of the noun "copula", meaning "something that connects or links together".

are met through rerouting rather than fast dynamic circuit establishment. This is in notable contrast to dynamic circuit provisioning approaches discussed above that rely on frequent circuit reconfigurations on relatively short time-scales for adapting to changing traffic conditions. Intuitively, the approach works because the provisioned circuits form a "logical" topology over the boundary routers with many "non-direct" paths that can carry the excess traffic from a source to its final destination. Our adaptive load-balancing approach exploits the significant amount of path diversity available for traffic rerouting.

While our proposed solution draws upon the pseudo-periodic behavior observed in real traffic to pre-compute offline circuit configurations, our approach differs from previous offline circuit configuration solutions in two important ways. First, previous offline circuit configuration approaches did not consider ways for adapting to unexpected traffic changes, which poses a legitimate concern for their deployment. For our adaptive rerouting mechanism, we apply a known adaptive routing algorithm based on game theory to utilize the available circuit capacities. The convergence and performance properties of the routing algorithm have been proved in [36]. But previous works on the adaptive routing are limited to the discussion of optimization techniques, and they generally assumed that the routing paths are given. In contrast, our work focuses on finding a set of loop-free paths with maximum path diversity and the setting of cost function for the optimization technique to achieve our expected network performance objectives. Section 1.5 describes our rerouting problem in more details.

Second, previous offline configuration approaches were designed to handle a specific traffic matrix or an entire set of traffic matrices [18, 73, 85, 74, 13]. Our work is different in that our formulation takes into consideration the statistical daily traffic variations observed in past measurements and the probability of traffic demands given their statistical distribution of occurrence in past measurements. In particular, we propose a new offline circuit configuration formulation that explicitly considers the statistical properties of past observations. In our formulation, the pre-computed coarse circuit configurations do not necessarily provide sufficient circuit capacities for supporting all the traffic matrices captured in the historical data

sets. Instead, our problem is formulated as a utility max-min fair bandwidth allocation problem that aims to maximize the acceptance probability of the expected traffic demand by using the cumulative distribution function over the historical data sets as the objective function. Our solution allocates all available network resources across multiple paths to provide as much "headroom" as possible. Since our solution does not rely on an online dynamic circuit creation mechanism, there is no need to leave behind network resources for establishing new circuits. To the best of our knowledge, the general multi-path utility max-min fair bandwidth allocation problem has not been solved previously. Previous utility max-min fair allocation formulations only considered the single-path case [22, 24, 79, 71], and previous multi-path max-min fair allocation formulations did not consider general (non-linear) utility functions [12]. Thus, an important contribution of this work is a first solution to this open general problem.

1.3 Approach Illustration

Here we use an example to illustrate our idea. As shown in Figure 1.3, the traditional packet routing approach relies on intermediate electronic routers to performance routing decisions. With Internet traffic rapidly increasing , it is unclear whether electronic routers that have been used at the core of backbone networks will continue to scale to match future traffic growth or optical link rates [25]. Furthermore, network traffic could suffer long latencies because of queueing and O/E/O (Optic-Electronic-Optic) conversion delays at each intermediate router.

The motivation for using circuit switching is to achieve an all-optical network where traffic is forwarded through a network without involving electronic routers. In contrast to previous approaches [97, 64, 91] that aim to dynamically create circuits upon traffic arrivals, COPLAR avoids the need for frequent switch reconfiguration and coordination by pre-configuring a circuit between each pair of boundary routers (Ingress and Egress routers) at a coarse time scale, such as on an hourly basis, based on historical traffic measurements. For example, as shown in Figure 1.4, if the average traffic from Seattle and Denver to New York are both

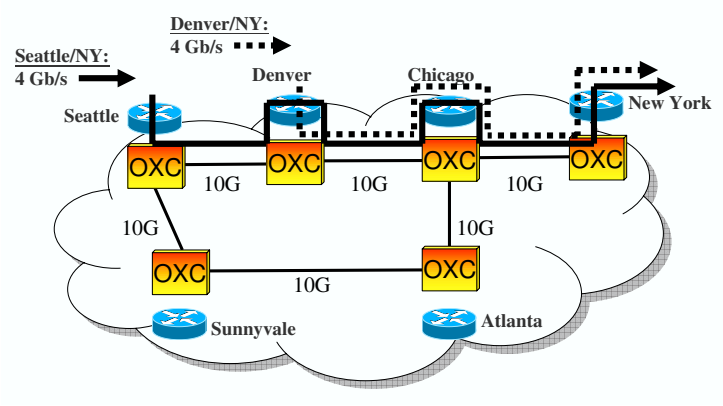


Figure 1.3: Network routing under traditional packet switching.

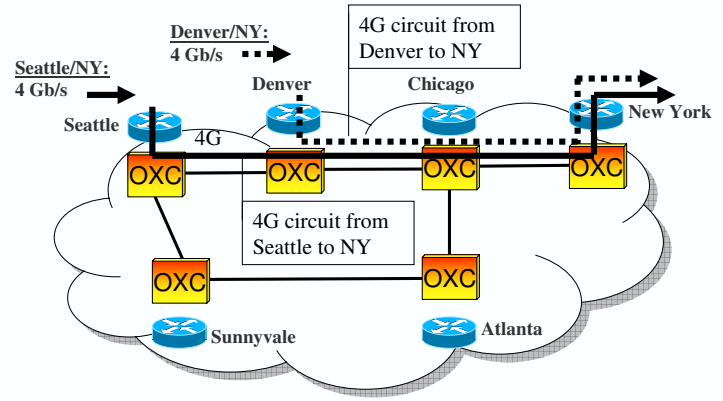


Figure 1.4: Network routing under circuit switching.

4 Gb/s between 1-2pm in the past month, we would configure a 4 Gb/s circuit for each of the two IE-pairs between 1-2pm, and the network traffic would be routed by default on these provisioned circuits during this period of time. To further handle traffic variation, we provide more bandwidth headroom by allowing a circuit to be setup across multiple physical circuits and by fully allocating network capacity to all IE-pairs. As shown in Figure 1.5, if the maximum traffic from Seattle to New York is 6 Gb/s, we could allocate a 6 Gb/s virtual circuit with 4 Gb/s going through Denver and the other 2 Gb/s going through Atlanta. Since there is limited bandwidth, and circuit capacity is not supposed to be shared among different IE-pairs, it is crucial to find a circuit provisioning algorithm that can maximize

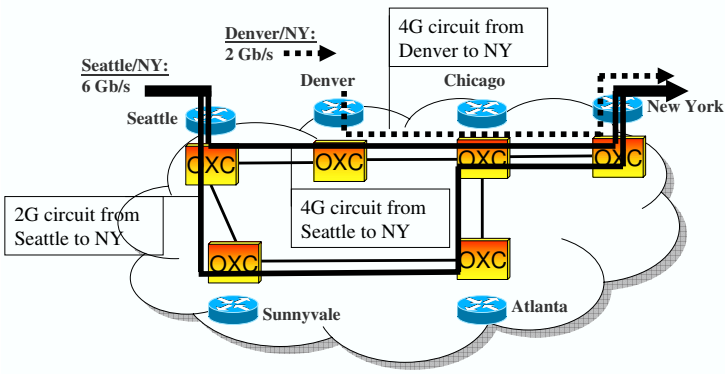


Figure 1.5: Achieve higher throughput by provisioning circuits over multiple paths.

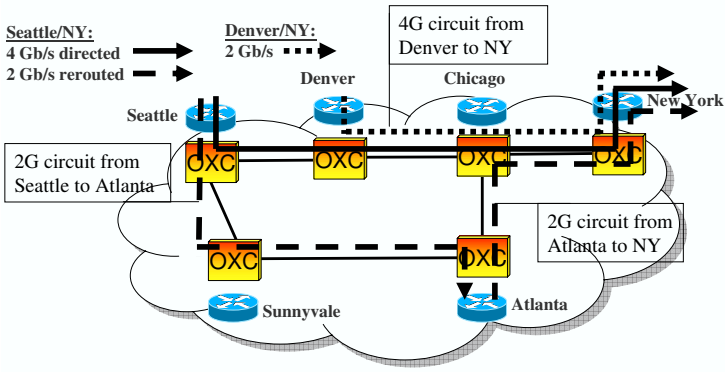
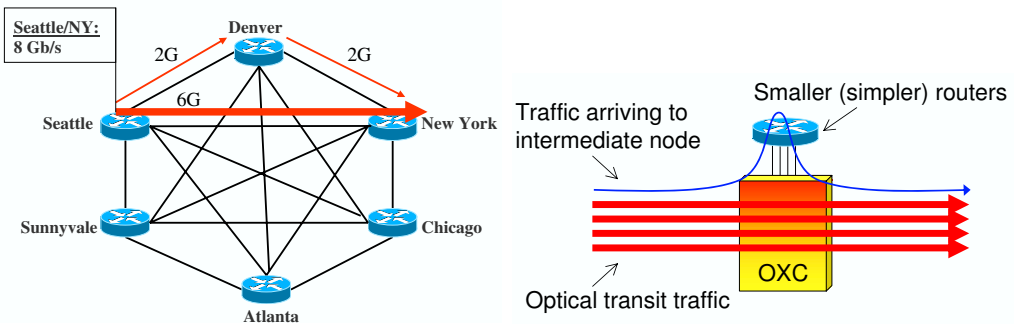


Figure 1.6: The 2 Gb/s exceeding traffic is rerouted on the residual capacity of other circuits.

the likelihood of having sufficient capacity for each of the circuits. Without going into the details of the algorithm, as we mentioned, it is formulated as a multi-path utility max-min bandwidth allocation where the utility function of each IE-pair is based on its historical traffic measurements.

Although our study results showed majority of the network traffic could be carried by our provisioned circuits, there still could be some excess demand caused by unexpected traffic changes. Thus the idea of our second concept, adaptive rerouting, is to utilize the residual capacity of circuits without having to dynamically reconfigure circuits on-the-fly. For example, assume the circuit configuration from the circuit provisioning algorithm is the one shown in Figure 1.6. The circuit capacities for IE-pairs (Seattle, New York) and (Denver, New York) are both



(a) Adaptive rerouting over virtual circuits. (b) Only limited workload is added on electronic routers for rerouting.

Figure 1.7: Abstract of rerouting over circuits in COPLAR

.

4 Gb/s, and the circuit capacities for IE-pair (Seattle, Atlanta) and (Atlanta, New York) are both 2 Gb/s. Suppose on rare occasions, the actual traffic from Seattle to New York becomes 6 Gb/s, which exceeds its circuit capacity of 4 Gb/s. Suppose there is no traffic for the IE-pairs (Seattle, Atlanta) and (Atlanta, New York). Therefore, the additional 2 Gb/s traffic from Seattle to New York could be rerouted through the residual capacity of the circuit from Seattle to Atlanta, then through the residual capacity of the circuit from Atlanta to New York. However, comparing to Figure 1.5, we do not need to allocate a full 6 Gb/s circuit for handling such rare occasions when rerouting is employed. Thus, the utilization of circuits and network throughput could be improved when rerouting is considered. Furthermore, since no circuit configuration needs to be changed during rerouting, no new signal protocol is required. Overall, Figure 1.7 (a) illustrates the our rerouting strategy at a high level. First, the circuit provisioning algorithm converts the network topology into a fully connected mesh by setting up a circuit between each IE-pair. Then an adaptive rerouting algorithm dynamically adjusts routing paths to handle excess traffic on top of the provisioned circuit network. Although the adaptive rerouting scheme would require the participation of electronic routers, majority of the traffic is expected to be carried by its direct default circuit. As a result, each electronic router only needs to forward a small amount of excess traffic during the rerouting process as shown in Figure 1.7 (b). Therefore, the overhead

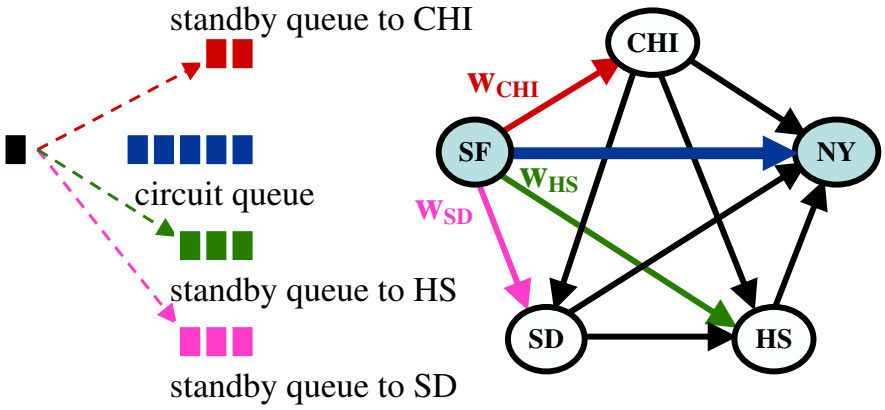


Figure 1.8: Rerouting mechanism design.

of electronic routers could be minimize while the network throughput could be maintained or even improved as shown in our evaluations in Chapter 4.

In terms of the implementation, as shown in Figure 1.8, after we provision a circuit between an IE-pair, it has a corresponding circuit queue and standby queue. Both queues act as buffer of a circuit at the network boundary. The circuit queue handles the direct traffic while the standby queue handles the reroute traffic. A circuit queue has a higher priority than the standby queue, so that rerouting traffic would only be handled by the spare circuit capacity. We always send traffic to its direct circuit if possible until the circuit queue is full. Regardless the queuing policy, when a packet cannot be inserted into its circuit queue, it is sent to a standby queue of the circuit to a next hops of a rerouting path. Then the packet becomes reroute traffic and is handled by standby queues at each intermediate nodes afterwards. More specifically, for each IE-pair, we maintain the separate set of split ratios to all possible next hops of rerouting paths. So, reroute traffic is sent to the standby queue of a next hop with a probability proportional to its corresponding split ratios. These split ratios are dynamically adjusted by an adaptive routing algorithm, so that the traffic can be load balanced among all rerouting paths.

1.4 Problem Formulation of Circuit Provisioning

Our circuit provisioning problem is to find a bandwidth allocation vector $B(t)$ and its corresponding circuits $P(t)$ for each time interval t, such that for any flow i, we can set up a circuit with capacity $B_i(t)$ by reserving bandwidth along a set of paths $P_i(t)$ at time t. Flow i refers to the aggregate traffic between the corresponding pair of ingress-egress (IE) router pair (s_i, d_i). Unless otherwise noted, we will use the terms a *flow* and an IE (ingress-egress) flow interchangeably. Also, for the remainder of the thesis, unless otherwise noted, we will simply refer to a "coarse circuit" as a circuit. The formal definition of a circuit configuration is given in Definition 1 where the index t is omitted for simplicity. Notice that in our problem definition, the paths are not given. Rather, path choices are *variables* in our problem. A *feasible* circuit configuration is one in which the circuit provisioning does not exceed any link capacity, as defined in Definition 2.

Definition 1 (Circuit configuration (B, P)). *B is a bandwidth allocation vector whose element B_i is the rate assigned to the flow i. Each rate B_i can be routed through a set of paths P_i. For each $P \in P_i$, $f(P)$ represents the portion of B_i that has been split to P.*

$$\sum_{\forall P \in P_i} f(P) = B_i, \forall \text{ flow } i. \tag{1.1}$$

Definition 2 (Feasible circuit configuration). *Given a (physical) network topology (V, E) with link capacities C, where $C(e)$ is the capacity of link e, a circuit configuration (B, P) is said to be feasible if and only the total bandwidth allocated to circuits passing each link e do not exceed the link's*

$$\sum_{\forall P \ni e} f(P) \leq C(e), \forall e \in E. \tag{1.2}$$

While there exists many possible feasible circuit configurations, we formulate the problem as a *multi-path utility max-min fair allocation* problem in which we derive our circuit configurations to capture the traffic variance observed in historical traffic measurements by using a Cumulative Distribution Function (CDF) model as the utility function. In particular, the use of CDFs [22] captures the

acceptance probability of a particular bandwidth allocation as follows. Let $X_i(t)$ be a random variable that represents the *actual* traffic for flow i at time t, and let $x_i(t)$ be the bandwidth allocation. Then the CDF of $X_i(t)$ or the acceptance probability of flow i is denoted as

$$Pr[X_i(t) \leq x_i(t)] = \Phi_{i,t}(x_i(t)).$$

Therefore, the intuition of maximizing acceptance probability is to have a circuit provisioning that can carry the traffic of each IE pair at the maximum probability by considering the traffic variance and distribution of individual IE pairs.

In general, the expected traffic can be modeled using different probability density functions with the corresponding CDFs. One probability density function is to use the empirical distribution that directly corresponds to the historical traffic measurements taken. Let $(r_{i,t}(1), r_{i,t}(2), \ldots, r_{i,t}(M))$ be M measurements taken for a flow i at a particular time of day t over some historical data set. Then for a flow i at time t, its acceptance probability or utility, $\Phi_{i,t}$, with a respect to a given bandwidth allocation $x_{i,t}$ is defined as

$$\Phi_{i,t}(x_i(t)) = \frac{\#\text{ measurements } \leq x_i(t)}{M} = \frac{1}{M}\left(\sum_{k=1}^{M} I(r_{i,k}(t) \leq x_i(t))\right),$$

where $I(r_{i,k}(t) \leq x_i(t))$ is the indicator that the measurement $r_{ij,k}(t)$ is less than or equal to $x_i(t)$.

Once our circuit provisioning is formulated as a multi-path utility max-min bandwidth allocation, we could solve a more general bandwidth allocation problem and apply to other network application as well.

1.5 Problem Description of Adaptive Rerouting

The main objective of our rerouting scheme is to increase network throughput by handling bursty or unexpected traffic changes through the residual capacity of circuits. Since the residual capacity of circuits is dynamically changing depend-

ing on the network traffic, an adaptive rerouting algorithm is needed to adjust routing paths on-the-fly. Several adaptive routing algorithms [33, 49, 35] have been proposed, and the majority of these works are focused on the optimization of network cost and the convergence time of the algorithm. These algorithms are well-developed. Therefore, it is not our intention to propose a fundamentally new adaptive routing algorithm. Instead, our goal is to apply an adaptive routing algorithm on COPLAR and evaluate the performance improvement on COPLAR. However, it is not trivial to apply an existing adaptive routing algorithm to our COPLAR framework because the challenges and objectives in our setting are different.

The first challenge is to explore routing path diversity without routing information explosion and routing loops. In previous adaptive routing studies, they focused on the selection of routing paths for traffic where routing paths are assumed to be known or given in the problem. In contrast, the selection of routing paths is considered as an optimization parameter in our problem setting. With more paths provided to our adaptive rerouting algorithm, potentially better network throughput could be achieved. However, considering all possible routing paths in our COPLAR routing framework is infeasible because the underlying topology is a fully connected mesh. The number of possible routing paths simply explodes. For each IE-pair, the number of possible paths is $N!$, where N is the network size. For example, if N is merely 50, the number of possible paths explodes to already more than 10^{64}. As a result, the paths used for adaptive rerouting in COPLAR must be implemented using a set of distributed routing tables, similar to IP routing tables. However, unlike traditional IP routing [66, 8], which only has one or few routing paths for each IE-pair, we would like to consider as many paths as possible to exploit greater path diversity for our rerouting scheme. Furthermore, routing loop has to be avoided because routing loops can cause unnecessary router overhead and capacity usage. In our evaluations, to be presented later in subsequent chapters, our evaluations show that the amount of resource wasted by routing loops can grow significantly as the network gets more congested. Moreover, routing loops can lead to live-locks where a packet may never arrive to its destination. There-

fore, in Chapter 3, we propose a loop-free routing table structure to ensure that the routing paths considered are without loops.

In addition to proposing a new loop-free routing table structure, we must compute the corresponding paths for our adaptive rerouting algorithm. We propose a propagation protocol that can compute path costs online by aggregating the costs along paths without the full-knowledge of routing information from every node.

Last but not least, we expect our adaptive rerouting algorithm for COPLAR can maximize network throughput while minimizing O/E/O conversions. Notice that the number of O/E/O conversion is the same as the number of rerouting hops over the provisioned circuits. Although maximizing network throughput is the primary objective of our rerouting scheme, reducing router workload or O/E/O conversion is also a critical requirement in our COPLAR routing paradigm. Otherwise, our routing strategy would be the same as packet routing. Inevitably, there is a tradeoff between network throughput and O/E/O conversion because adaptive routing algorithms often use longer paths to load balance traffic and achieve higher network throughput. Thus, it is not trivial as to how hop count should be considered in an adaptive routing algorithm. In particular, we don't want to simply limit the available path options to be within a certain length because that would only guarantee the worst case hop count.

1.6 Additional Related Work

A number of related work has already been introduced in Section 1.1. As such, we restrict discussion in this section to related work not yet described to minimize redundancy. To implement the actual circuit configurations, a number of signaling protocols have been proposed. For example, the Generalized Multi-Protocol Label Switching (GMPLS) [17, 98] has been proposed as a way to extend MPLS [76] to incorporate circuit switching in the domains of time, frequency and space. GMPLS defines the signaling mechanisms for various management aspects of constructing coarse label-switched paths that are circuit-switched. The scope of GMPLS is complementary to ours in that it does not specify a control algorithm to

decide when to create circuits and with what capacity. Rather, GMPLS provides us with one way to implement our pre-computed coarse circuit configurations.

Besides the dynamic circuit configuration approaches mentioned in Section 1.1, another interesting transport architecture proposal is optical flow switching (OFS) [91]. The approach is different than the high-level design depicted in Figure 1.1 in that the user-initiated data circuits are established between end-users rather than boundary routers. Nonetheless, it provides another representative transport architecture for creating circuits "on-the-fly", which can adapt to changing traffic conditions along with the necessary signaling mechanism and control plane to change circuit configurations on relatively short time-scales.

Approaches based on fast optical packet switching have also been proposed (e.g. [92, 95, 99]). For example, in the work of [99], the core of the optical comprises of high-performance optical packet routers with intelligent edge routers at the boundary of the optical network to perform intelligent traffic shaping.

Another approach based on long-duration coarse circuits is based on two-phase routing [101, 54, 53]. Rather than configuring circuits to support a specific set of traffic matrices, researchers have suggested the configuration of long-duration circuits that are optimized for the worst-case throughput under all possible traffic patterns permissible within the network's natural ingress-egress capacity constraints. These approaches work by setting up static coarse circuits and sending traffic across the optical circuit-switched cloud twice in a load-balanced manner in two phases. However, optimizing for the worst-case commonly leads to rather pessimistic performance.

Finally, motivated by the desire to adapt to changing traffic conditions, possibly corresponding to sudden traffic shifts, a number of online adaptive routing policies have been developed. TeXCP [49], MATE [33], and REPLEX [35] are representative examples. These approaches take advantage of alternative routing paths in the network. These adaptive routing approaches are complementary to our work in that we can incorporate them into our COPLAR architecture for routing over non-direct paths via traversal through multiple circuits.

1.7 Outline of the Thesis

Given the rapidly increasing traffic and optical transport capabilities, and growing disparity between optical capabilities and Moore's Law, we need a new approach to bridge these gaps in the design of future optical networks. This thesis proposes a novel optical network architecture design based on a new paradigm of "coarse optical circuit switching by default" and "adaptive rerouting of excess traffic over circuits with spare capacity" when necessary. The outline of this thesis is as follows:

Chapter 2 defines and solves the multi-path utility max-min bandwidth allocation problem for the circuit provisioning part of our approach. The majority of work on max-min fairness has been limited to the case where the routing of flows has already been defined and this routing is usually based on a single fixed routing path for each flow. But in this chapter, we consider the more general problem in which the routing of flows, possibly over multiple paths per flow, is an optimization parameter in the bandwidth allocation problem. Our goal is to determine a routing assignment for each flow so that the bandwidth allocation achieves optimal utility max-min fairness with respect to all feasible routings of flows. Both optimal and local algorithms for the problem are proposed for the first time to the problem.

Chapter 3 describes how we apply an adaptive routing algorithm based on game theory for our rerouting mechanism. Specifically, we propose a loop-free routing table structure to explore path diversity without forming routing loops and a propagation protocol for computing path costs. In addition, we design a cost function to achieve our network performance objectives.

Chapter 4 uses a well-known realistic network simulator NS2 to extensively evaluate our COPLAR approach on two real large PoP-level backbone networks, namely Abilene [2] and GEANT [44], with real traffic trace data over two months. Our evaluation results show a number of interesting results. First, the majority of the traffic was able to be carried by the circuits computed from our circuit provisioning algorithm even during peak traffic hours. In addition, when the actual traffic matrices were scaled up, COPLAR was surprisingly able to handle a higher traffic load than conventional packet routing [66, 8]. The effectiveness of COPLAR

can be attributed to the path diversity available for the adaptive rerouting of traffic. Finally, COPLAR achieved higher network throughput with significantly less overhead on electronic routers and O/E/O conversions for traffic.

To show our bandwidth allocation algorithm and framework can be extended on other common network applications and problems, we apply our bandwidth allocation framework on a network security problem and propose a novel defense mechanism called PSP (Proactive and Surge Protection) in Chapter 5. The objective of PSP is to avoid network congestion collapse at a core network caused by bandwidth based DDoS (Distributed Denial-of-Services) attacks. PSP uses a similar bandwidth allocation framework as COPLAR to deploy soft admission control at the ingress routers of a core network and to prioritize the dropping of potential attack traffic at congested routers. The PSP application was evaluated on two large tier-1 commercial backbone networks, and the results show that we can significantly reduce the packet drop rate caused by DDoS attacks.

Finally, Chapter 6 concludes this thesis by addressing some limitations and future works.

Chapter 1, in part, is a reprint of the material as it appears Journal of Optical Communications and Networking, 2009. Chou, Jerry; Lin, Bill, IEEE OSA Press, 2009. The dissertation author was the primary investigator and author of this paper.

Chapter 2

Multi-path Utility Max-Min Bandwidth Allocation

2.1 Introduction

Bandwidth allocation is a fundamental problem in various areas of networking. In this chapter, we consider a general allocation problem in which a network consisting of links with fixed capacity is given along with a set of flows between source and destination pairs. The problem is to allocate a rate or bandwidth to each flow without exceeding link capacity. On one hand, we would like to improve the overall network utilization by maximizing the total throughput from all flows. On the other hand, fairness among flows must be maintained to guarantee the performance of individual flows. Therefore, an important goal of bandwidth allocation is to maximize the utilization of network resources while sharing the resources in a fair manner among network flows.

To strike a balance between fairness and throughput, a widely studied criterion in the network community is the notion of *max-min fairness* [22]. An allocation of bandwidths or rates is said to be max-min fair if it is not possible to increase the bandwidth of a flow without decreasing another already smaller flow. While max-min fair allocation treats all flows evenly and tends to allocate them with similar bandwidths, many variants [51, 24, 71] of max-min fair allocation

have been proposed to differentiate the bandwidth requirements among flows. One such max-min fair variant is *weighted max-min fair allocation* [22], which assigns a weight to each flow. According to the weights, a flow would receive a bandwidth allocation proportional to its weight to gain the same fairness as others. Therefore, by giving varied weights to flows, the bandwidth requirement of flows can be differentiated. However, as shown in [24], those traditional max-min bandwidth allocations will often result in significant disparity in the actual throughput or performance of a flow, despite a "fair" allocation of bandwidth. Therefore, to further capture the general and possibly non-linear relationship between bandwidth allocation and the throughput of a flow, utility functions [81] were introduced as a general performance measure, and *utility max-min fair allocation* [24] was formulated to optimize for the max-min fairness of flows in terms of their utilities.

Nevertheless, the majority of work on max-min fairness has been limited to the case where the routing of flows has already been defined and this routing is usually based on a single fixed routing path for each flow. Although this setup simplifies the problem by decoupling the complicated flow routing problem from bandwidth allocation, the utilities that can be achieved by flows are unnecessarily hamstrung by routing decisions that have been fixed, ignoring potentially better allocations that could be achieved if optimal routing and bandwidth allocation were solved together simultaneously, and if the path diversity can be exploited by splitting traffic over multiple paths (e.g. by using MPLS tunnels). Therefore, in this chapter, we consider the more general problem in which the routing of flows, possibly over multiple paths per flow, is an optimization parameter in the bandwidth allocation problem. Our goal is to determine a routing assignment for each flow so that the bandwidth allocation achieves optimal utility max-min fairness with respect to all feasible routings of flows. We call the resulting bandwidth allocation and optimal routing as a *multi-path utility max-min fair allocation*.

As we know, most max-min fair allocation formulations are based on some iterative water-filling algorithm [22]. In each iteration, the algorithm aims to maximize the allocation of all flows. The flows whose bandwidth cannot be further increased are then identified as saturated and are fixed for the remaining itera-

tions. However, we face several new challenges when we consider the multi-path routing of flows as a part of the optimization problem. In particular, max-min fair allocation algorithms for the fixed single-path case generally rely on some notion of bottleneck link that determines the maximum common utility. However, under simultaneous multi-path routing optimization, the bandwidth allocation of a flow is not necessarily throttled by a certain link because it may be possible to reroute flows over different combinations of multiple paths to achieve higher utilities. To achieve global optimality, flows may need to be rerouted along different paths at each iteration. Such additional degrees of freedom make our more general problem significantly harder.

To the best of our knowledge, our combined optimal multi-path routing and bandwidth allocation problem under utility max-min fairness has not been solved previously. Specifically, the main contributions of our work are as follows: First, we present a global optimization algorithm that is guaranteed to find an optimal routing and bandwidth allocation that can achieve optimal utility max-min fairness with respect to all feasible routings of flows, including multi-path routings. Second, we propose a fast fully polynomial iterative ϵ-approximation algorithm that can be efficiently implemented using a linear program solver. Third, we propose two local algorithms to achieve a local utility max-min allocation with less computational cost and complexity. Finally, evaluate these algorithms by using historical traffic distributions as utility functions to model expected future traffic demands. Our evaluations show that significantly higher minimum utility and lower excess demand can be achieved when multi-path routing is considered simultaneously with bandwidth allocation. Also, our local solution is able to achieve the similar results as optimal solution with much faster computational time.

The remainder of this chapter is organized as follows. First, Section 2.2 reviews related work. We next briefly provide in Section 2.3 background material on max-min allocation, utility functions, and multi-path routing. After we give the formal definition of our multi-path utility max-min fair allocation problem in Section 2.4, we present the optimal and local algorithms for solving this problem in Section 2.5 and Section 2.6, respectively. Then Section 2.7 evaluates our multi-

path utility max-min allocation algorithms with results demonstrating significant improvements in utilities that can be achieved. Finally, Section 2.8 concludes the chapter.

2.2 Related Work

Max-min fairness has been a widely-studied measure of fairness in the network community. However, the vast majority of work on max-min fairness has focused on the problem where routing decisions have already been fixed, often based on a single fixed routing path per flow. Several centralized solutions based on global knowledge of the network have been developed [10, 22, 46]. Distributed algorithms [11, 15, 19, 42] have also been proposed to achieve max-min fairness by adjusting flow rates based on limited link states and local flow information. In addition, several max-min fair variants have been studied, such as proportional max-min fairness [51] and utility max-min fairness [24]. In particular, utility max-min has been applied to several application-oriented allocation problems, such as flow control [58], link resource [63], etc.

As discussed in Section 2.1, the simultaneous optimal multi-path routing and bandwidth allocation problem under the general setting of utility max-min fairness has not been solved before. Even the routing problem in the context of traditional (weighted) max-min fair bandwidth allocation is rarely discussed in the previous literature. The fair bandwidth allocation for single source flows was first studied by [62]. [52] and [30] proposed approximation algorithms to find unsplittable flow routings. The fair bandwidth allocation problem has also been studied in the online setting where a route is assigned to each flow when it arrives. [26] proposed a heuristic routing algorithm which selects the best single route for a new flow based on link congestions, such that the max-min bandwidth allocation is maximized after the flow is added. [37] developed an approximation algorithm that could achieve a max-min fair allocation with $O(\log^2 n \log^{1+\epsilon} U/\epsilon)$-competitive ratio. This bound was further improved in [23]. Finally, multi-path routing under fair bandwidth allocation has been studied [12, 41, 48, 60] as well. But majority

of works [48, 60] consider routing as input rather than an optimization parameter. While we consider [12] and [41] have the closest problem formulation to us, both works only considered the weighted max-min case instead of the more general utility max-min problem with arbitrary utility functions. As shown in [24] and Figure 2.1, utility functions simply cannot be capture as a linear line, and it is non-trivial to extend the solution from weights to nonlinear functions.

2.3 Background

We consider a network consists of N nodes connected by M links $\ell = (\ell_0, \ell_1, \ldots, \ell_{M-1})$ with link capacity $c(\ell_i)$ for any link ℓ_i. Given n commodities $\Gamma = (C_0, C_1, \ldots, C_{n-1})$ where C_i represents the flow from node s_i to t_i, our objective is to decide an allocation vector r whose component r_i is the rate for commodity C_i. Notice, the terms of flow and commodity are used interchangeably.

2.3.1 Max-Min Fair Bandwidth Allocation

Max-min fair is one of the most widely-used fairness criteria in bandwidth allocation. Its general definition is as follows:

Definition 3 (Max-min fair bandwidth allocation). *An allocation vector $r = (r_0, r_1, \ldots, r_{n-1})$ is max-min fair when any component r_i of r cannot be increased without decreasing some already smaller or equal component r_k ($r_k \leq r_i$).*

In previous works, commodities were restricted to use a given routing path. Thus, the set of feasible bandwidth allocations is defined as follows:

Definition 4 (Feasible bandwidth allocation). *A feasible bandwidth allocation $r = (r_0, r_1, \ldots, r_{n-1})$ assigns rate r_i to commodity C_i such that no link in the network is congested:*

$$\sum_{\forall C_i \ uses \ \ell_j} r_i \leq c(\ell_j), \forall \ell_j \in \ell \tag{2.1}$$

Section 2.3.3 provides the definition of feasible allocation under the more general setting where the routing of flows is not known and a flow can be routed over multiple paths.

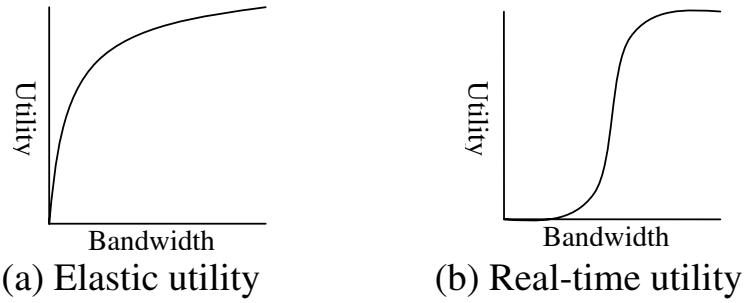


Figure 2.1: Utility functions for different application classes.

2.3.2 Utility Functions

Utility functions were first introduced into the bandwidth allocation problem by [81] to capture the performance of application flows. For example, elastic applications, including traditional data applications like emails and file transfers, are best described by a convex utility function. In contrast, a real-time application could be modeled by a nearly single-step function because of its sensitivity to bandwidth requirement. As shown in Figure 2.1, from those application flows, their performance simply cannot be capture by a weight or linear function.

Therefore, utility max-min fairness based on a set of utility functions $\phi = \{\phi_0, \phi_1, \ldots, \phi_{n-1}\}$ was introduced as an alternative performance measure. Each function $\phi_i \in \phi$ computes the utility for commodity C_i delivered under its allocated bandwidth r_i as

$$\mu_i = \phi_i(r_i), \forall i \in n \tag{2.2}$$

We assume utility functions are strictly increasing function over the domain range $[0, 1]$; that is, $\phi_i(k) < \phi_i(k'), \forall k' > k$ and $0 < \mu_i < 1, \forall i$. Thus, the inverse of a utility function is also well-defined as

$$r_i = \phi_i^{-1}(\mu_i), \forall i \in n \tag{2.3}$$

Accordingly, a utility max-min allocation corresponding to a given set of utility functions is defined as follows:

Definition 5 (Utility max-min bandwidth allocation). *A utility max-min allocation is a feasible bandwidth allocation vector* $r = (r_0, r_1, \ldots, r_{n-1})$ *where any component* r_i *of* r *cannot be increased without decreasing some component* r_k *with equal or smaller utility* ($\phi_k(r_k) \leq \phi_i(r_i)$).

2.3.3 Multi-Path Routing

We consider the bandwidth allocation problem under multi-path routing where a commodity can use an arbitrary routing, and its traffic can be split over multiple paths. A general formulation for an arbitrary routing assignment is $\mathbf{R}$ where $\mathbf{R}_{ij}$ is the fraction of traffic from commodity C_i routed on link ℓ_j, and we say a routing assignment $\mathbf{R}_{ij}$ is feasible if and only if it satisfies the following constraint.

Definition 6 (Feasible multi-path allocation). *A feasible multi-path bandwidth allocation vector* $r = (r_0, r_1, \ldots, r_{n-1})$ *assigns rate* r_i *to commodity* C_i *where* r *can be realized by a feasible routing assignment* $\mathbf{R}$ *without violating the flow conservation and/or overloading the network, such that*

$$\sum_{\ell_j \in E^+(k)} r_i R_{ij} - \sum_{\ell_j \in E^-(k)} r_i R_{ij} = \begin{cases} r_i & \text{if } k = s_i \\ -r_i & \text{if } k = t_i \\ 0 & \text{otherwise} \end{cases}$$

$$\forall i \in n \text{ and } k \in N \tag{2.4}$$

$$\sum_{\forall C_i} R_{ij} \cdot r_i \leq c(\ell_j) \qquad \forall j \in M \tag{2.5}$$

$$R_{ij} \geq 0 \qquad \forall i \in n \text{ and } j \in M \tag{2.6}$$

where $E^+(k)$ *and* $E^-(k)$ *represent the set of incoming and outgoing links at node* k.

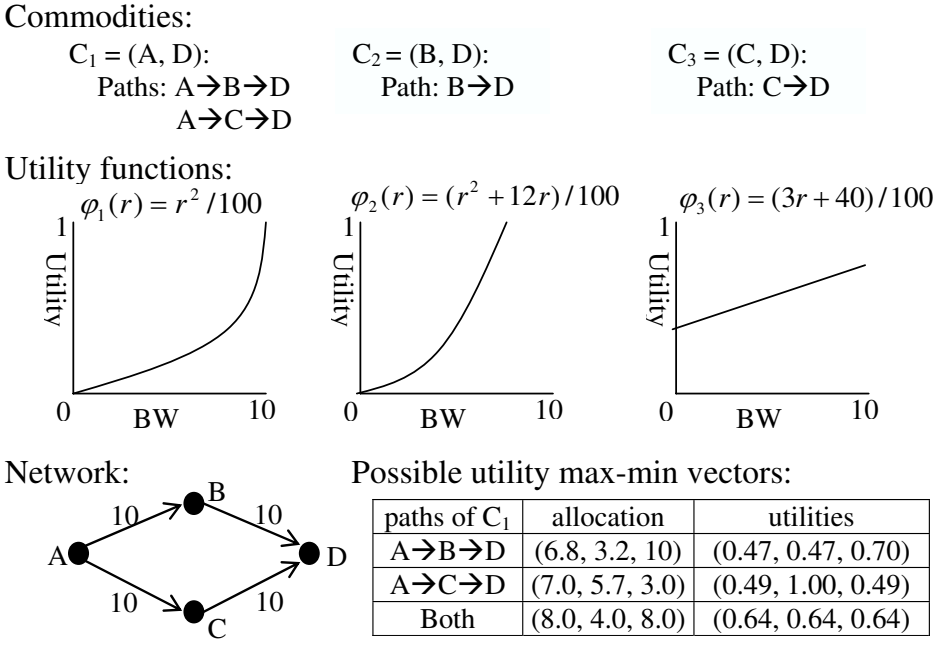
2.4 Problem Definition

2.4.1 Motivation

We start with an example to illustrate the difference in bandwidth allocation when considering utility functions and multi-path routing. Figure 2.2 shows a network with four nodes interconnected by 10-units bandwidth links. The network has three commodities, (A, D), (B, D) and (C, D), and their utility functions corresponding to a given bandwidth allocation r are $\phi_1(r) = r^2/100$, $\phi_2(r) = (r^2 + 12r)/100$ and $\phi_3(r) = (3r + 40)/100$, respectively. Notice that some of the utility functions given are non-linear. In this example, both commodities (B, D) and (C, D) have only one possible routing path each. However, commodity (A, D) has two possible routing paths, $A \rightarrow B \rightarrow D$ and $A \rightarrow C \rightarrow D$.

First, we consider the traditional max-min allocation problem where commodities are routed over a single path, and we assume the path specified for commodity (A, D) is $A \rightarrow B \rightarrow D$. Under the single-path max-min allocation defined by Definition 3 and Definition 4, the max-min fair allocation vector is $(5, 5, 10)$. This arises by assigning a common 5-units of bandwidth to all three commodities in the first iteration, which would saturate both commodities, (A, D) and (B, D), respectively. The third commodity (C, D) is increased to a full 10-units of bandwidth in the second iteration. Corresponding to this max-min allocation, the resulting utilities for commodities (A, D), (B, D), and (C, D) are $\phi_1(5) = 0.25$, $\phi_2(5) = 0.84$, and $\phi_3(10) = 0.70$, respectively.

On the other hand, under the single-path *utility* max-min allocation defined by Definition 5, the utility max-min fair allocation vector is $(6.8, 3.2, 10)$, and the corresponding utilities achieved are $\phi_1(6.8) = 0.47$, $\phi_2(3.2) = 0.47$, and $\phi_3(10) = 0.70$, respectively. This arises by allocating bandwidth to achieve the maximum common utility for all three commodities in the first iteration, which in this example is 0.47. However, to achieve this maximum common utility, more bandwidth for example needs to be allocated to the first commodity (A, D) than to the second commodity (B, D). Again, the bandwidth allocation can be further increased for the third commodity (C, D) to achieve a higher utility. Comparing with



paths of C_1	allocation	utilities
A→B→D	(6.8, 3.2, 10)	(0.47, 0.47, 0.70)
A→C→D	(7.0, 5.7, 3.0)	(0.49, 1.00, 0.49)
Both	(8.0, 4.0, 8.0)	(0.64, 0.64, 0.64)

Figure 2.2: Multi-path utility max-min example.

the traditional max-min allocation, clearly the utility max-min allocation achieves better fairness with respect to the given utility functions since utility max-min allocation specifically aims to do so. In particular, the minimum utility is increased from $\phi_1(5) = 0.25$ to $\phi_1(6.8) = 0.47$.

When we further consider utility max-min allocation under *multi-path routing*, there could be three possible allocation results, depending on the choice of routing paths available to commodity (A, D). If we choose the path $A \rightarrow B \rightarrow D$ to route commodity (A, D), then the allocation would be the same as the previous utility max-min allocation. However, if we choose the path $A \rightarrow C \rightarrow D$ to route commodity (A, D), then the utility max-min fair allocation would be $(7, 5.7, 3)$, and the corresponding utilities would be $(0.49, 1.00, 0.48)$. Finally, if we choose to use both paths to route commodity (A, D), then the utility max-min allocation would be $(8, 4, 8)$, where commodity (A, D) would be allocated 6 units of bandwidth along the path $A \rightarrow B \rightarrow D$ and 2 units of bandwidth along the path $A \rightarrow C \rightarrow D$. As a result, the corresponding utilities to the allocation would be $(0.64, 0.64, 0.64)$.

Comparing the three utility allocation results, not only does the last allocation fully utilize available capacities along all links, it also achieves better fairness than the allocation results using a single routing path because no one commodity can increase its utility by decreasing another one. In fact, by considering multi-path routing, it guarantees to achieve equal or better fairness than the traditional single-path bandwidth allocation because the solution space with single-path routing is only a subset of the solution space with multi-path routing. Therefore, utility max-min allocation under multi-path routing provides a more powerful and general fair allocation framework, but it is also a much more complicated problem because it couples flow routing and bandwidth allocation together.

2.4.2 Definitions

As shown by the example in Figure 2.2, a utility max-min allocation exists for each given routing assignment. We define a local multi-path utility max-min allocation as the utility max-min allocation with respect to a given routing assignment $\mathbf{R}$ in Definition 7. Then according to the utility order defined in Definition 9, we define the optimal multi-path utility max-min allocation as the largest one among all local allocations in Definition 10. Finally, we say an allocation is an ϵ-approximation to the optimal solution if it satisfies the requirement defined in Definition 11.

Definition 7 (Local multi-path utility max-min allocation). *A local multi-path utility max-min allocation is a feasible multi-path allocation vector* $r = (r_0, r_1, \ldots, r_{n-1})$ *where any component* r_i *of* r *cannot be increased without decreasing some component* r_k *with equal or smaller utility* ($\phi_k(r_k) \leq \phi_i(r_i)$) *under some routing assignment* $\mathbf{R}$.

Definition 8 (Utility-ordered allocation vector). *Given a allocation* $r = (r_0, r_1, \ldots r_{n-1})$, *we define a corresponding utility-ordered allocation vector* $\overline{r} = (r_{i_0}, r_{i_1}, \ldots, r_{i_{n-1}})$, *such that* $\phi_{i_k}(r_{i_k}) \leq \phi_{i_{k+1}}(r_{i_{k+1}})$ *for* $k = 0 \ldots n-2$, *where* f_i *is the utility function of commodity* C_i.

Definition 9 (Utility order ($>_u$)). *Given allocation vectors a, b and their utility-ordered allocation* $\overline{a} = (a_{i_0}, a_{i_1}, \ldots, a_{i_{n-1}})$ *and* $\overline{b} = (b_{j_0}, b_{j_1}, \ldots, b_{j_{n-1}})$*, we say* $a >_u b$ *if only if there is some m such that* $\phi_{i_k}(a_{i_k}) = \phi_{j_k}(b_{j_k})$ *for* $0 \leq k < m$ *and* $\phi_{i_m}(a_{i_m}) > \phi_{j_m}(b_{j_m})$.

Definition 10 (Optimal multi-path utility max-min allocation). *The optimal multi-path utility max-min fair allocation is a feasible multi-path allocation vector that is the largest among all local multi-path utility max-min allocations under the ordering defined by* $>_u$*. In other words,* $r = (r_0, r_1, \ldots, r_{n-1})$ *is an optimal multi-path utility max-min vector if any commodity* C_i *cannot be increased without decreasing the rate of another* C_j*, such that* $\mu_j \leq \mu_i$ *where* $\mu_j = \phi_j(r_j)$ *and* $\mu_i = \phi_i(r_i)$ *under any routing assignment.*

Definition 11 (ϵ-approximation to the optimal allocation). *Let* $r = (r_0, r_1, \ldots, r_{n-1})$ *be the optimal multi-path allocation vector, and* $\overline{r} = (r_{i_0}, r_{i_1}, \ldots, r_{i_{n-1}})$ *is its utility-ordered vector. We say another utility max-min allocation vector* $r' = (r'_0, r'_1, \ldots, r'_{n-1})$ *with its utility-ordered vector* $\overline{r'} = (r'_{j_0}, r'_{j_1}, \ldots, r'_{j_{n-1}})$ *is* ϵ*-approximation to the optimal multi-path utility max-min allocation r if there is some m such that* $\phi_{j_k}(r'_{j_k})(1+\epsilon) \geq \phi_{i_k}(r_{i_k})$ *for* $0 \leq k < m$ *and* $\phi_{j_m}(r'_{j_m}) > \phi_{i_m}(r_{i_m})$.

2.5 Optimal Algorithms

In this section, we first provide a general problem formulation to achieve the optimal multi-path utility max-min allocation. We then provide a linear programming (LP) formulation solution to achieve an approximate optimal solution. Table 2.5 summarizes all the variables used in these algorithms.

2.5.1 OPT_MP_UMMF

The basic idea to solving a utility max-min allocation problem is to iteratively increase the utility of all commodities and determine the maximum common utility that can be achieved in each iteration. Commodities that reached their maximum utilities are tagged as saturated and removed from the water-filling process by fixing their corresponding utility.

Table 2.1: Variables used in multi-path utility max-min algorithms.

ℓ	a set of links $\ell_0, \ell_1, \ldots, \ell_{M-1}$ with capacity $c(\ell)$
Γ	set of commodities $C_0, C_1, \ldots, C_{n-1}$ where C_i from s_i to t_i
ϕ_i	utility function for commodity C_i
r_i	bandwidth allocated to commodity C_i
$\mathbf{R}$	a feasible routing assignment where $\mathbf{R}_{ij}$ is the fraction of traffic from commodity C_i routes on link ℓ_j
μ_i	utility achieved by commodity C_i
d_i	a temporary rate assignment for C_i
π	a iteration counter starts from 1
Γ^{π}_{SAT}	set of commodities identified as saturated at iteration π
Γ^{π}_{UNSAT}	$\Gamma^{\pi}_{UNSAT} = \Gamma \backslash \bigcup_{k=0}^{\pi-1} \Gamma^{k}_{SAT}$
μ^{π}_{max}	the maximum common utility achieved at iteration π

To formulate our problem into an iterative form, we have an iterative optimization algorithm, OPT_MP_UMMF, which consists of the following steps. The algorithm starts with initializing $\pi = 1$, $\Gamma^0_{SAT} = \emptyset$, and stops when all commodities are identified as saturated (i.e., $\Gamma^{\pi}_{UNSAT} = \emptyset$).

Step 1: Find the maximum common utility μ^{π}_{max} that can be achieved by all unsaturated commodities.

$$\text{maximize } \mu^{\pi}_{max} \tag{2.7}$$

$$\text{subject to} \tag{2.8}$$

$$d_i = \phi_i^{-1}(\mu_i), \forall C_i \in \bigcup_{k=0}^{\pi-1} \Gamma^k_{SAT} \tag{2.9}$$

$$d_i = \phi_i^{-1}(\mu^{\pi}_{max}), \forall C_i \in \Gamma^k_{UNSAT} \tag{2.10}$$

$$\sum_{\forall C_i} \mathbf{R}_{ij} \cdot d_i \leq c(\ell_j), \forall \ell_j \in \ell \tag{2.11}$$

In the formulation, the first two constraints give the bandwidth requirement of each commodity. In particular, Equation 2.9 sets the bandwidth for the saturated commodities as their previously assigned utility, while Equation 2.10 sets the bandwidth of the unsaturated commodities to the current maximum common utility. Finally, there must be a feasible routing assignment $\mathbf{R}$ that can carry the bandwidth requirement d by satisfying the constraint in Equation 2.11.

Step 2: Identify newly saturated commodities, Γ^{π}_{SAT}, by testing each unsaturated commodity $C_i \in \Gamma^{\pi}_{UNSAT}$ with the following optimization problem such that commodity C_i is saturated if its utility cannot be increased by any routing.

$$\text{maximize } \tau \tag{2.12}$$

$$\text{subject to} \tag{2.13}$$

$$d_j = \phi_j^{-1}(\mu_j), \forall C_j \in \bigcup_{k=0}^{\pi-1} \Gamma^{k}_{SAT} \tag{2.14}$$

$$d_j = \phi_j^{-1}(\mu^{\pi}_{max}), \forall C_j \in \Gamma^{\pi}_{UNSAT} \backslash C_i \tag{2.15}$$

$$d_i = \phi_i^{-1}(\mu^{\pi}_{max} + \tau) \tag{2.16}$$

$$\sum_{\forall C_i} \mathbf{R}_{ij} \cdot d_i \leq c(\ell_j), \forall \ell_j \in \ell \tag{2.17}$$

The above optimization problem fixes the rates of all commodities in Equations 2.14 and 2.15, except the commodity C_i being tested. It then finds the maximum utility that can be increased for commodity C_i in Equation 2.16 while there still exists some feasible routing $\mathbf{R}$ to carry all commodities. Therefore, if $\tau < 0$, we cannot increase the utility of commodity C_i by any routing and $\Gamma^{\pi}_{SAT} = \Gamma^{\pi}_{SAT} \bigcup C_i$.

Step 3: Assign the utility and bandwidth for each newly saturated commodity $C_i \in \Gamma^{\pi}_{SAT}$.

$$\mu_i = \mu^{\pi}_{max}, r_i = \phi_i^{-1}(\mu^{\pi}_{max})$$

The last step is to assign the bandwidth allocation for the newly saturated commodities and move them into the saturated set Γ^{π}_{SAT}. Once commodities are in the saturated set, their bandwidth allocations and utilities will not be changed, but their routing paths still could be altered to better utilize residual capacities and achieve higher utilities for the remaining unsaturated commodities in later iterations.

Next, we prove the correctness of the above optimal algorithm.

Theorem 1. *The allocation vector r returned by OPT_MP_UMMF is optimal multi-path utility max-min.*

Proof. According to Definition 10, we have to prove that the rate of any commodity C_i cannot be increased without decreasing the rate of another C_j, such that $\mu_j \leq \mu_i$ where $\mu_j = \phi_j(r_j)$ and $\mu_i = \phi_i(r_i)$ under any routing $\mathbf{R}$.

To prove by contradiction, we assume commodity $C_i \in \Gamma^{\pi}_{SAT}$ is identified as saturated at iteration π and its bandwidth can be increased without decreasing the bandwidth of any commodity C_j which has less or equal utility than C_i.

First of all, according to the OPT_MP_UMMF algorithm, if the utility of some commodity $C_j \in \Gamma^{\pi'}_{SAT}$ is less than or equal to the utility of commodity C_i, then $C_j \in \bigcup_{k=0}^{\pi} \Gamma^{k}_{SAT}$. This is because their utilities are assigned to be the maximum common utility of the iteration when they are identified as a saturated commodity, and μ_{max} is a non-decreasing vector. Thus, if $\mu_j \leq \mu_i$, then $\pi' \leq \pi$ and $C_j \in \bigcup_{k=0}^{\pi} \Gamma^{k}_{SAT}$.

Then in Step 2 of the OPT_MP_UMMF algorithm, at iteration π, when we test for commodity C_i, we set the rate of any commodity $C_j \in \bigcup_{k=0}^{\pi} \Gamma^{k}_{SAT}$ to be their final allocation r_j and the rate of any commodity $C_j \notin \bigcup_{k=0}^{\pi} \Gamma^{k}_{SAT}$ to be the maximum common utility at iteration π, μ^{π}_{max}, which is also the same as r_i because $r_i = \mu^{\pi}_{max}$. However, we still cannot increase the bandwidth of commodity C_i by any feasible routing. Therefore, we cannot increase the utility of commodity C_i by decreasing the bandwidth of any commodity C_j with greater utility C_i. In other words, we have to increase the utility of commodity C_i by decreasing the bandwidth of some commodity C_j with less or equal utility than C_i, and that is in contradiction to our assumption. □

2.5.2 ϵ-OPT_MP_UMMF

The OPT_MP_UMMF algorithm is a non-linear optimization problem because the utility functions used in Step 1 can be non-linear. Therefore, we propose a fast fully polynomial approximation algorithm, ϵ-OPT_MP_UMMF, which uses a binary search formulated as linear programming to find the maximum common utility. In addition, we re-formulate the optimization problems in ϵ-OPT_MP_UMMF

such that they can all be solved as a well-defined Maximum Concurrent Flow (MCF) [50] routing problem. Given set of commodity C with its demand function, $d(C)$, and a set of links ℓ with capacities, $c(\ell)$, a MCF solver finds a routing $\mathbf{R}$ to maximize λ, which is the common fraction of demand for each commodity that can be routed with the given link capacities.

The modified Step 1 of the ϵ-OPT_MP_UMMF algorithm is as follows.

Step 1: Binary search the utility domain to achieve the maximum common utility μ^{π}_{max} for unsaturated commodities. The initial values of variables are $\mu_{high} = 1, \mu_{low} = 0, \lambda = 0$.

while $\mu_{high} - \mu_{low} \geq \delta$ and $\lambda < 1$
  $\mu^{\pi}_{max} = (\mu_{high} + \mu_{low})/2$
  $d_i = \phi_i^{-1}(\mu_i), \forall C_i \in \bigcup_{k=0}^{\pi-1} \Gamma^k_{SAT}$
  $d_i = \phi_i^{-1}(\mu^{\pi}_{max}), \forall C_i \in \Gamma^{\pi}_{UNSAT}$
  $(\lambda, \mathbf{R}) = MCF(C, d, \ell, c)$
  if $\lambda < 1$
    $\mu_{high} = \mu^{\pi}_{max}$
  else
    $\mu_{low} = \mu^{\pi}_{max}$
  endif
endwhile

The binary search procedure starts with guessing the maximum common utility as a value in the utility domain range 0 to 1. Then with a given common utility μ, we verify if μ can be achieved by finding a feasible routing to carry the corresponding bandwidth allocation; that is, the required bandwidth to reach common utility μ for unsaturated commodities and to satisfy previously assigned utilities for saturated commodities. According to the Maximum Concurrent Flow (MCF) problem, by assigning the demand of commodities as the required bandwidth for the common utility μ, there exists a feasible routing to achieve the common utility μ if the λ returned by the MCF solver is ≥ 1.0. Because utility functions are strictly increasing, we find μ as the maximum common utility when there is no

feasible routing to achieve an even greater common utility $\mu + \delta$, where δ can be an arbitrarily small value depending on the ϵ selected by the algorithm.

Since the maximum common utility in found Step 1 is approximated, the utility of a saturated commodity could be further increased. Thus, we must change Step 2 accordingly to guarantee the utility of a saturated commodity can only be increased by at most a fraction of ϵ to its current utility.

Step 2: Identify newly saturated commodities Γ^{π}_{SAT} by verifying if each unsaturated commodity C_i meets the following saturation test where $C_i \in \Gamma^{\pi}_{SAT}$ if and only if the utility of C_i cannot be further increased by ϵ of its current utility.

$d_j = \phi_j^{-1}(\mu_j), \forall C_j \in \bigcup_{k=0}^{\pi-1} \Gamma^k_{SAT}$
$d_j = \phi_j^{-1}(\mu^{\pi}_{max}), \forall C_j \in \Gamma^{\pi}_{SAT} \backslash C_i$
$d_i = \phi_i^{-1}(\mu^{\pi}_{max} \cdot (1+\epsilon))$
$(\lambda, \mathbf{R}) = MCF(C, d, \ell, c)$
if $\lambda < 1$
  $\Gamma^{\pi}_{SAT} = \Gamma^{\pi}_{SAT} \bigcup C_i$
endif

In the above Step 2, we determine a commodity as saturated if its utility cannot be increased by a fraction of ϵ to the current utility under any feasible routing assignment. Again, we use a MCF solver to verify if there exists a feasible routing to increase the utility of a commodity by assigning its demand to the required bandwidth for achieving an additional ϵ fraction of its current utility. If the λ returned from a MCF solver is less than 1, then we know that the utility of the commodity cannot be increased by more than ϵ with any feasible routing, and therefore the commodity should be identified as saturated.

Notice, the difference between δ and ϵ is that δ means increasing limit for common utility of unsaturated commodities, while ϵ represents the increasing limit of the utility of individual unsaturated commodities. Thus, if we can increase the common utility by δ, it does not mean the utility of each unsaturated commodity cannot be increased by δ. Therefore, δ has to be chosen carefully with respect to ϵ to guarantee that there is at least one commodity whose utility cannot be

increased by ϵ when the common utility cannot be increased by δ. We construct such δ based on ϵ in Lemma 1.

Lemma 1. *For a given ϵ, there exists some δ such that there is at least one commodity identified as saturated at every iteration in the ϵ-OPT_MP_UMMF algorithm.*

Proof. The basic idea is to construct δ based on ϵ. If we can find a routing to increase the utility for each previous unsaturated commodity by ϵ, there must exist a routing which can increase the utility of all previous unsaturated commodities by δ. Thus, we prove the lemma by contradiction because if all previous unsaturated commodities in Step 2 can increase its utility by more than ϵ and remain as unsaturated, then there must exist a routing to increase the utility of all previous unsaturated commodities by more then δ, which clearly contradicts to the termination condition of the binary search in Step 1.

Now we construct such a δ as the following. At any iteration π, we know the current maximum common utility is μ^{π}_{max}. For each commodity $C_k \in \Gamma^{\pi}_{UNSAT}$, if it is identified as unsaturated, there must exist a routing $\mathbf{R}^k$ to carry the addition bandwidth $\triangle d_i$ for commodity C_i where

$$\triangle d_k = \phi_k^{-1}(\mu^{\pi}_{max}(1+\epsilon)) - \phi_i^{-1}(\mu^{\pi}_{max})$$

Therefore, if all $C_k \in \Gamma^{\pi}_{UNSAT}$ are identified as unsaturated commodities, we can construct a feasible routing $\mathbf{R}'$ to carry an additional bandwidth $\triangle d'_k$ for each commodity $C_k \in \Gamma^{\pi}_{UNSAT}$ by combining all routings $\mathbf{R}^k$ where

$$\triangle d'_k = \frac{\triangle d_k}{\mid \Gamma^{\pi}_{UNSAT} \mid} \text{ and } R'_{ij} = \sum_{\forall k \in \Gamma^{\pi}_{UNSAT}} \frac{R^k_{ij}}{\mid \Gamma^{\pi}_{UNSAT} \mid}$$

Accordingly, let $\triangle \delta_k$ be the utility can be increased corresponding to the additional bandwidth $\triangle d'_k$ $\forall C_k \in \Gamma^{\pi}_{UNSAT}$.

$$\triangle \delta_k = \phi_k(\phi_k^{-1}(\mu^{\pi}_{max}) + \triangle d'_k) - \mu^{\pi}_{max}.$$

Then we choose δ as the minimum value of all $\triangle\delta_k$, so

$$\delta = \min(\triangle\delta_k, \forall C_k \in \Gamma^{\pi}_{UNSAT}).$$

As a result, if we can increase the utility of each $C_i \in \Gamma^{\pi}_{UNSAT}$ by ϵ, there must exist a feasible routing to increase the utility of all $C_i \in \Gamma^{\pi}_{UNSAT}$ by δ. Therefore, at least one commodity is identified as saturated by the δ we found. □

Finally, we show that the ϵ-OPT_MP_UMMF algorithm achieves an ϵ-approximation allocation to the optimal solution, and the algorithm eventually terminates as following.

Theorem 2. *ϵ-OPT_MP_UMMF achieves ϵ-approximation to the optimal multi-path utility max-min allocation.*

Proof. Let $\mu^{*\pi}_{max}$ and μ^{π}_{max} be the maximum common utility achieved by the optimal and ϵ-OPT_MP_UMMF algorithm,respectively. We say ϵ-OPT_MP_UMMF algorithm achieves ϵ approximation to the optimal if there exist some m such that $\mu^{\pi}_{max} \cdot (1+\epsilon) \geq \mu^{*\pi}_{max}, \forall \pi < m$ and $\mu^{m}_{max} > \mu^{*m}_{max}$.

We prove by induction. After first iteration, clearly $\mu^{1}_{max} \cdot (1+\epsilon) \geq \mu^{*1}_{max}$, because otherwise it would contradict the condition of saturation test. After second iteration, let μ'^{2}_{max} be the optimal maximum common utility can be achieved when the utility of commodity $C_i \in \Gamma^{1}_{SAT}$ is assigned to be μ^{1}_{max}. Then $\mu'^{2}_{max} > \mu^{*2}_{max}$ because $\mu^{1}_{max} < \mu^{*1}_{max}$. Since $\mu^{2}_{max} \cdot (1+\epsilon) > \mu'^{2}_{max}$ and $\mu'^{2}_{max} > \mu^{*2}_{max}$, $\mu^{2}_{max} \cdot (1+\epsilon) > \mu^{*2}_{max}$.

After π iterations, if $\mu^{\pi-1}_{max} > \mu^{*\pi-1}_{max}$, we find $m = \pi$. Otherwise, we know $\mu^{k}_{max} < \mu^{*k}_{max}, \forall k < \pi$. Let μ'^{π}_{max} be the optimal maximum common utility can be achieved when the utility of commodity $C_i \in \Gamma^{k}_{SAT}$ is assigned to be $\mu^{k}_{max}, \forall k < \pi$. Then $\mu'^{\pi}_{max} > \mu^{*\pi}_{max}$, because the utility assignment for any saturated commodity $C_i \in \bigcup^{\pi}_{k=1} \Gamma^{k}_{SAT}$ is less than its optimal utility μ^{*k}_{max}. Again, since the saturation test guarantees $\mu^{\pi}_{max} \cdot (1+\epsilon) > \mu'^{\pi}_{max}$ and we already know $\mu'^{\pi}_{max} > \mu^{*\pi}_{max}$, $\mu^{\pi}_{max} \cdot (1+\epsilon) > \mu^{*\pi}_{max}$ holds for at any iteration π as long as $\mu^{k}_{max} < \mu^{*k}_{max}, \forall k < \pi$. □

Theorem 3. *ϵ-OPT_MP_UMMF algorithm terminates after at most n iterations, where n is the number of commodities.*

Proof. According to Lemma 1, at least one commodity is identified as saturated after each iteration. Since the algorithm terminates after all commodities are saturated, it has at most n iterations, where n is the number of commodities. □

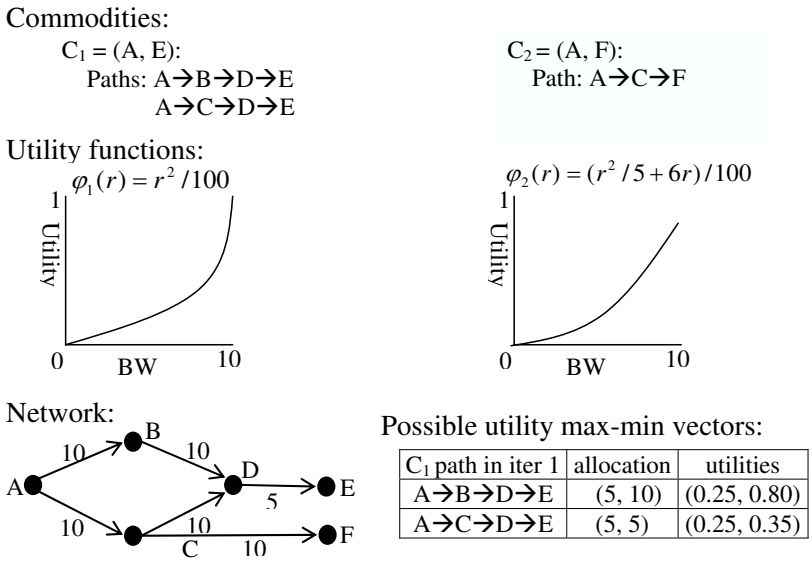
Conclude all, the overall ϵ-OPT_MP_UMMF algorithm is also polynomial time solvable with respect to the size of network because (1) there can be at most n iterations, (2) the number of binary searchers at each iteration is a constant with respect to ϵ, and (3) each binary search step involves a MCF problem that can be solved in polynomial time.

2.6 Local Algorithms

To further reduce computational cost and algorithm complexity, in this section we consider local utility max-min solutions which fix routing after each allocation iteration. However, local solutions could result smaller utility max-min allocation under the utility order as we will explain it in Section 2.6.1 followed by the description of our two local algorithms ϵ-LOCAL_MP_UMMF and PWL-LOCAL_MP_UMMF.

2.6.1 Example

We use Figure 2.3 to illustrate why fixing routing after each iteration could cause small utility max-min bandwidth allocation. In the example, we have two commodities, (A, E) and (A, F), and their utility functions are $\phi_1(r) = r^2/100$, $\phi_2(r) = (r^2/5 + 6r)/100$, respectively. For commodity (A, E), there are two paths $A \to B \to D \to E$ and $A \to C \to D \to E$, while commodity (A, F) has only one path $A \to C \to F$. All links in the network has 10 unit bandwidth, except the link $(D \to F)$ is 5. Therefore, in the first iteration, the maximum common utility can be achieved for both commodities is only 25% and commodity (A, E) is saturated. Since in the first iteration we only need to route 5 unit for commodity (A, E), it can use either of its two paths. If commodity (A, E) choose path $A \to C \to D \to E$ in the first iteration and the routing is fixed after the iteration, there is only 5 unit bandwidth left for commodity (A, F). As a result, the final utility allocation is



C_1 path in iter 1	allocation	utilities
A→B→D→E	(5, 10)	(0.25, 0.80)
A→C→D→E	(5, 5)	(0.25, 0.35)

Figure 2.3: Local multi-path utility max-min example.

only (0.25, 0.35). On the other hand, if we could reroute traffic after each iteration, we will reroute commodity (A, E) on path $A \rightarrow B \rightarrow D \rightarrow E$ instead, and the utility of commodity (A, F) can be greatly increased to 80%.

As we can see from the example, the reason fixing routing could produce smaller utility is because we saturate certain link in the early iteration and limit the path options for other unsaturated commodities in the future iteration. Therefore, it is important to use a load balance routing algorithm like MCF to allocate bandwidth, so that a link would not be saturated unless necessary. As we shown by our experimental results in Section 2.7.3, even a local algorithm could achieve similar results to the optimal solution. In the following, we will describe our local algorithm in more detailed.

2.6.2 ϵ-LOCAL_MP_UMMF

Our first local algorithm, ϵ-LOCAL_MP_UMMF, is the same as ϵ-OPT_MP_UMMF except we fix the routing and disallow traffic rerouting by removing allocated link capacity after each iteration. Accordingly, we simplify the steps of our ϵ-OPT_MP_UMMF algorithm as the following.

Step 1 Find maximum common utility by a binary search with the following rate assignment in each search iteration instead.

$$d_i = 0, \forall C_i \in \bigcup_{k=0}^{\pi-1} \Gamma_{SAT}^k$$
$$d_i = \phi_i^{-1}(\mu_{max}^{\pi}) - \phi_i^{-1}(\mu_{max}^{\pi-1}), \forall C_i \in \Gamma_{UNSAT}^{\pi}$$

Step 2 Identify saturated commodities by residual capacity.

$c(\ell) = c(\ell) - \sum_{\forall C_i \in \Gamma_{SAT}^{\pi}, P_j \ni \ell} \mathbf{R}_{ij} \cdot d_i$
for each $C_i \in \Gamma_{UNSAT}^{\pi}$
  $\triangle d_i = \phi_i^{-1}(\mu_{max}^{\pi} \cdot (1+\epsilon)) - \phi_i^{-1}(\mu_{max}^{\pi})$
  if $\sum_{\forall P_j \ni C_i} c(P_j) < \triangle d_i$
    $\Gamma_{SAT}^{\pi} = \Gamma_{SAT}^{\pi} \bigcup C_i$
  endif
endfor

Step 3: Assign the utility and bandwidth for each newly saturated commodity $C_i \in \Gamma_{SAT}^{\pi}$ and remove link capacity from later iterations.

$$\mu_i = \mu_{max}^{\pi}, r_i = \phi_i^{-1}(\mu_{max}^{\pi})$$
$$c(\ell) = c(\ell) - \sum_{\forall C_i \in \Gamma_{SAT}^{\pi}, P_j \ni \ell} \mathbf{R}_{ij} \cdot d_i$$

Comparing to the optimal algorithm, because the local algorithm is able to remove link and commodities from the network after each iteration, the problem space keep decreasing. More important, the expensive saturation test is replaced with a simple capacity check. Therefore, the above ϵ-LOCAL_MP_UMMF has much lower algorithm complexity than the previous optimal algorithm ϵ-OPT_MP_UMMF.

2.6.3 PWL-LOCAL_MP_UMMF

Next, considering the utility functions could be formulated or approximated by piecewise linear functions, we propose another local algorithm PWL-LOCAL_MP_UMMF that replaces the binary search in Step 1 of ϵ-LOCAL_MP_UM

MF by a single step scaling problem. Under the definition of piecewise linear function, a utility function for commodity C_i is defined by s+1 points $(x_i[0], y_i[0])$, $(x_i[1], y_i[1]), \cdots, (x_i[s], y_i[s])$. For example, Figure 2.4(a) are three piecewise utility functions, and the utility function in solid line is defined by points (10, 0.2), (15, 0.5), (20, 0.7) and (100, 1.0).

The main idea of PWL-LOCAL_MP_UMMF algorithm is to divide the utility space into s number of horizontal segments, then iteratively increase utility by segments. Since, in each segment, all commodities have linear utility functions, the algorithm is able to find the maximum common utility by a simple scaling equation. For simplicity, let $\Delta x_i[\kappa] = x_i[\kappa] - x_i[\kappa-1]$ and $\Delta y_i[\kappa] = y_i[\kappa] - y_i[\kappa-1]$. The PWL-LOCAL_MP_UMMF algorithm finds the maximum common utility at segment κ and iteration π by using the following modified Step 1 with initial values, $\lambda = 1, \pi = 0$, and $\kappa = 1$.

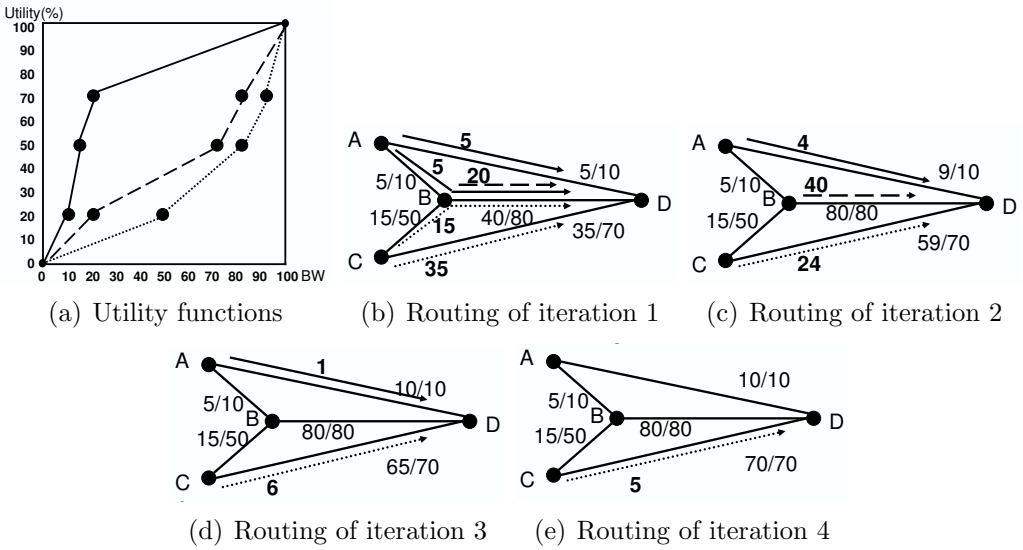
Step 1 Find the maximum common utility in segment κ

$d_i = 0, \forall C_i \in \bigcup_{k=0}^{\pi-1} \Gamma_{SAT}^k$
if $\lambda \geq 1$
  $d_i = \Delta x_i[\kappa], \forall C_i \in \Gamma_{UNSAT}^{\pi}$
else
  $d_i = d_i \cdot (1 - \lambda), \forall C_i \in \Gamma_{UNSAT}^{\pi}$
endif
$\lambda = MCF(d, c)$
if $\lambda < 1$
  $\mu_{max}^{\pi} = \mu_{max}^{\pi-1} + d_i \cdot \lambda \cdot \frac{\Delta y_i[\kappa]}{\Delta x_i[\kappa]}$, any $C_i \in \Gamma_{UNSAT}^{\pi}$
else
  $\mu_{max}^{\pi} = \mu_{max}^{\pi-1} + d_i \cdot \frac{\Delta y_i[\kappa]}{\Delta x_i[\kappa]}$, any $C_i \in \Gamma_{UNSAT}^{\pi}$
  $\kappa++$
endif

Figure 2.4 and Table 2.2 illustrate the PWL-LOCAL_MP_UMMF algorithm in a network with four nodes and three commodities. For simplicity, all links in the network is single direction from left to right, and their link capacities are varied

Table 2.2: Illustration of the PWL-LOCAL_MP_UMMF algorithm.

Iter.	Segment (utility)	Routed demand (A,D)	(B,D)	(C,D)	λ	Saturated commodities
1	1(20)	10/10	20/20	50/50	2.0	
2	2(50)	4/5	40/50	24/30	0.8	(B,D)
3	2(50)	1/1	0	6/6	1.0	(A,D)
4	3(70)	0	0	5/10	0.5	(C,D)
Allocated BW		15	60	85		
Allocated Utility		50	44	60		



(a) Utility functions (b) Routing of iteration 1 (c) Routing of iteration 2

(d) Routing of iteration 3 (e) Routing of iteration 4

Figure 2.4: Illustration of PWL-LOCAL_MP_UMMF algorithm.

from 10 to 80 as indicated in the figure. The utility functions are formulated by piecewise linear functions and divided into 4 segments between 0%, 20%, 50%, 70% and 100% as shown in Figure 2.4 (a). We illustrate the algorithm by iterations and list out the allocated bandwidth and utility resulting from the end of each iteration in Table 2.4 with the corresponding flow routing and link usage shown in Figure 2.4 (b)-(e).

The algorithm starts with the first segment and determines until all commodities are saturated or 100% utility is reached. In the first iteration, the algorithm uses the bandwidth difference, $\Delta x[1]$, in the first segment (0%-20%) as

the routed demand for each commodity, that is 10, 20 and 50 for commodities (A, D), (B, D) and (C, D), respectively. Given the requested demand, a MCF solver routes demand as shown in Figure 2.4 (b) to achieve the maximum λ, 2. Since λ is larger then 1, the second iteration proceeds to the next segment (20%-50%) and repeat the MCF flow routing on the residual link capacity with demand 5, 50, 30. As shown in Figure 2.4(c), in the second iteration, the maximum λ can be achieved is only 0.8 because link (B, D) is saturated. As the result, we have to scale down the routed demand by λ, 0.8, and identify commodity (B, D) as saturated. Since λ is less than 1 after the second iteration, the third iteration stays at the same segment (20%-50%) and route the reminding demand 1 and 6 for unsaturated commodities (A, D) and (C, D), respectively. Its routing shown in Figure 2.4(d) indicates there is just enough capacity to route the demand, so the algorithm can proceed to the next segment and commodity (A, D) is identified as saturated. Then the fourth iteration proceeds to the third segment (50%-70%) with the demand of the only left commodity (C, D). Finally, the algorithm is terminated because link (C, D) is fully utilized and commodity (C, D) is also saturated as shown in Figure 2.4(d). The final bandwidth allocation of commodities is the summation of all iterations. The last two rows of Table 2.2 summarize the allocated bandwidth and achieved utility of each commodity. As we can see, the algorithm terminates within 4 iterations and only calls the MCF solver once in each iteration. Therefore, the algorithm complexity can be greatly reduced.

2.7 Evaluation

In the following, we first briefly introduce our evaluation setup. We then compare our multi-path utility max-min allocation results with existing max-min allocation solutions to demonstrate the improvements of our approach. Finally, we analysis the effectiveness and complexity of our local algorithms.

Figure 2.5: Abilene network topology.

2.7.1 Setup

We evaluated our bandwidth allocation solutions on the Abilene network using actual network topology and traffic trace data. As shown in Figure 2.5, the Abilene network has 11 nodes interconnected by 10 Gb/s links. The traffic trace data can be found in [100] as a set of traffic matrices. Each traffic matrix contains the demand rate between an OD pair at a 5-minute time interval, and it is computed based on the actual packet sampling information collected from network routers. In Section 2.7.2 and 2.7.2, we compare our multi-path utility max-min allocation with the traditional weighted max-min and utility max-min allocations under single-path problem formulation. For simplicity, we use MP_UMMF to denote the results from our optimal multi-path utility max-min allocation algorithm, while UMMF and WMMF represent the single-path allocation of utility max-min and weighted max-min, respectively. Specifically, for the two single-path max-min allocations scenario, we fixed the routing path of each commodity to its shortest path between the source and destination node. We determine the weight and utility function of each commodity based on the historical traffic measurements over 2 months period from 3/1/04 to 4/21/04. In the weighted max-min allocation scenario, the weight of a commodity is the average demand over its historical traffic measurements. On the other hand, in the utility

max-min allocation scenario, we determine the utility of a commodity as the empirical cumulative distribution function of its historical demands, so that the utility value is directly corresponding to the *acceptance probability* (i.e., the probability of having sufficient bandwidth allocation for a commodity). The allocation of both single-path max-min allocations can be solver by the traditional water-filling algorithm [22], while our multi-path utility max-min allocation is computed by solving the ϵ-OPT_MP_UMMF algorithm with the linear programming optimization tool CPLEX [43]. Finally, Section 2.7.3 compares the allocation results computed from our optimal algorithm and the two local algorithms, ϵ-LOCAL_MP_UMMF and PWL-LOCAL_MP_UMMF. All experimental results were computed on a Intel 2 GHz Duo CPU with 3 GB memory.

2.7.2 Max-Min Fair Allocations Comparison

Here, we compare the allocation results of MP_UMMF, UMMF and WMMF. As we know, current backbone networks are over-built to accommodate fluctuations in traffic. To emphasize the importance of bandwidth allocation when link capacity is relatively scarce, we adjusted the degree of resource contention in our evaluations by scaling down the link capacity (10 Gb/s) by a factor of 10 to 20. In the following, we first show the utilities of individual commodities achieved by each of the allocations when link capacity is 1 Gb/s. Then we compare the minimum utility and excess demand under varied degrees of resource contention.

Utility of Individual Commodities

We take the allocation results when link capacity is 1 Gb/s as an example to show the utility achieved for each commodity. The results under other degrees of resource contention are discussed later. Since the Abilene network has 11 nodes, there are 110 OD pairs in total, with the corresponding commodities defined. We plot the utility of each commodity in Figure 2.9 from the commodity with the lowest utility to the highest utility, and we have the following observations.

First, the results of utility max-min allocations appear as step functions because all commodities identified as saturated in the same iteration have the

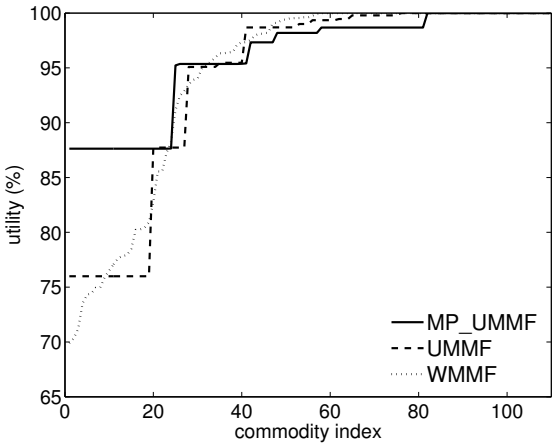


Figure 2.6: The utility of individual commodities under different max-min allocations when link capacity is 1 Gb/s.

same utility/acceptance probability. But, in weighted max-min allocation, the commodities saturated at the same iteration could have different utilities because its linear utility function is only an approximation to the actual non-linear utility function of acceptance probability.

Second, we found MP_UMMF achieves much greater utility than UMMF and WMMF for most of the commodities, especially for the ones with smaller utility. For example, the minimum utility of MP_UMMF, UMMF and WMMF are 87.63%, 75.99% and 69.89%, respectively. In other words, MP_UMMF increases the minimum utility of UMMF and WMMF by a factor of 1.15x (87.63/75.99) and 1.25x (87.63/69.89), respectively. In addition, the result of UMMF is also higher than WMMF by a factor of 1.10x (87.63/69.89).

As defined by the utility order in Definition 9, the allocation results with higher minimum utility are fairer. Therefore, our multi-path utility multi-path allocation appears to improve both the fairness and total utility of bandwidth allocation. Accordingly, we further investigate the minimum utility and actual excess demand under varied resource contention in Section 2.7.2 and 2.7.2, respectively.

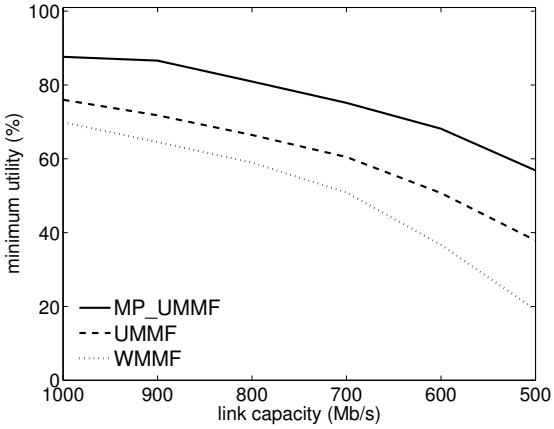


Figure 2.7: The minimum utility of max-min allocations when link capacity (10 Gb/s) is scaled down by a factor of 10 to 20 with every 100 Mb/s apart.

Minimum Utility

Figure 2.7 plots the minimum utility under varied degrees of resource contention when we scale down the link capacity by a factor of 10 to 20 with every 100 Mb/s apart. For all allocations, the minimum utility decreases as the degree of resource contention increased. However, MP_UMMF consistently achieves the highest utility among all allocations, especially when the degree of resource contention is higher. This is because the allocation becomes crucial as the link capacity is limited. For example, when link capacity is 500 Mb/s, the minimum utilities of MP_UMMF, UMMF and WMMF are 56.84%, 37.87% and 19.12%, respectively. In other words, the MP_UMMF is able to further increase the minimum utility of UMMF and WMMF allocations by a factor of 1.50x (56.84/37.87) and 2.97x (56.84/19.12), respectively. As shown from the figure, although UMMF optimizes for the same utility functions as MP_UMMF, it is not able to efficiently utilize bandwidth by adjusting routing. As a result, it clearly achieves much less utility than the multi-path allocation. Furthermore, the minimum utility of WMMF is ever smaller than UMMF because it is difficult to use a single weight value to capture or approximate the actual non-linear utility function. Therefore, only MP_UMMF can consistently achieve the best results by considering both multi-path routing and nonlinear utility functions.

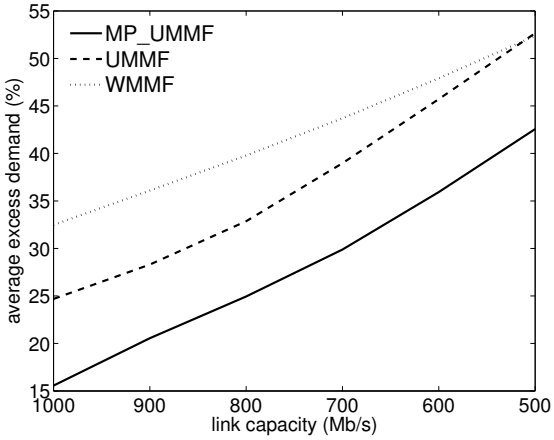


Figure 2.8: The excess demand of max-min allocations when link capacity (10 Gb/s) is scaled down by a factor of 10 to 20 with every 100 Mb/s apart.

Excess demand

Next, we compare the excess demand under varied link capacity. Under a given link capacity, we determine the excess demand of the traffic offering during 5-days period from 4/22/04 to 4/26/04 which is different from the traffic datasets for computing our bandwidth allocations. Figure 2.8 plots the average excess demand over the results at all time intervals within the 5-days period. As shown in the figure, we have lower excess demand with greater link capacity because each commodity can be allocated more bandwidth and less traffic demand would excess the bandwidth allocation. But, again MP_UMMF is still able to achieve the lowest excess demand among the three. For example, when link capacity is 1,000 Mb/s, the excess demand of MP_UMMF, UMMF and WMMF is 15.56%, 24.69% and 32.48%, respectively. In other words, MP_UMMF substantially reduces the excess demand of UMMF and WMMF by 36.98% and 52.09%, respectively. Therefore, our improvement on the utilities shown previously also transfers to less excess demand for our COPLAR circuit provisioning.

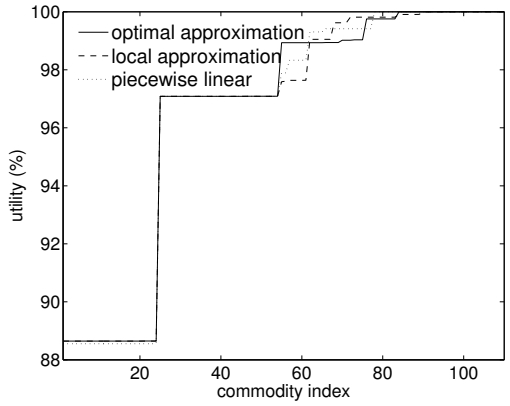


Figure 2.9: The utility of individual commodities under optimal and local multi-path utility max-min allocations when link capacity is 1 Gb/s.

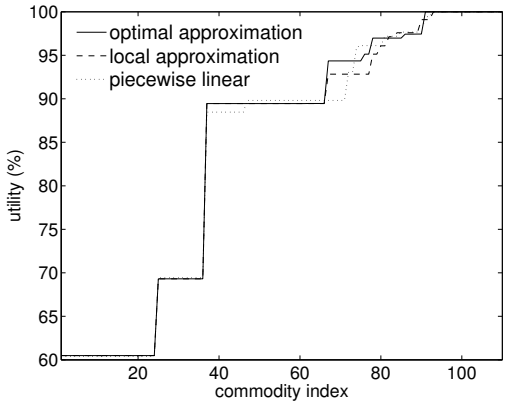


Figure 2.10: The utility of individual commodities under optimal and local multi-path utility max-min allocations when link capacity is 500 Mb/s.

2.7.3 Local Algorithm Analysis

We analyze our two local algorithms, ϵ-LOCAL_MP_UMMF and PWL-_LOCAL_MP_UMMF, by comparing them with the optical algorithm ϵ-OPT_MP_UMMF. First of all, we show that our local algorithms are able to produce smaller but similar allocation results to the approximated optical solution. Figure 2.9 and 2.10 plot the allocated utility results of commodities with link capacity 1 Gb/s and 500 Mb/s. As we have shown in Figure 2.7, less network link capacity results lower bandwidth allocation and utility. But also here we observed the results of both local algorithms were very close to the optimal solution. Specifically, the average utility difference between ϵ-LOCAL_MP_UMMF and ϵ-OPT_MP_UMMF in three figures were 0.14%, 0.53% and 2.32%, respectively. Similarly, the average utility difference between PWL-LOCAL_MP_UMMF and ϵ-OPT_MP_UMMF in three figures were 0.15%, 0.27% and 3.12%, respectively. Even the 90 percentile number of the utility differences between ϵ-LOCAL_MP_UMMF and ϵ-OPT_MP_UMMF were only 0.44%, 0.99% and 8.74%, while the 90 percentile number between ϵ-LOCAL_MP_UMMF and ϵ-OPT_MP_UMMF were 0.73%, 1.52% and 10.96%, respectively. There are greater differences under less link capacity because early link saturation is more likely to occur in such network and causes lower utility

allocation for local algorithms. However, as shown by the average number and figures, these larger difference points suddenly occur. Also notice from the figures, optimal algorithm still achieves the greatest utility max-min allocation according to the utility order because it has larger utility value for commodities with less utility. But, optimal does not provide strictly higher utility for all commodities because of the limited network capacity. Thus, we could still find some commodities have higher utility values under local algorithm. But those commodities are mostly located at the end of the curves, so the fairness among commodities can be maintained.

Next, we demonstrate how much computational time and algorithm complexity can be reduced by using local algorithms. Table 2.3 summarizes the results under several network settings. For simplicity, we use OPT, LOC and PWL to denote algorithms ϵ-OPT_MP_UMMF, ϵ-LOCAL_MP_UMMF and PWL-LOCAL_MP_UMMF, respectively. We represent the complexity of algorithms by the number of function calls to a MCF solver because it is most expensive operation in all three algorithms. Generally, complexity increases when network has less link capacity because fewer commodities are able to achieve 100% utility and more computation iterations are required. But it may also depend on the number of commodities identified as saturated at each iteration. As we observed, ϵ-LOCAL_MP_UMMF actually had less MCF calls at 500 Mb/s than 1 Gb/s. Nevertheless, the optimal algorithm clearly required a lot more number of MCF function calls than the other two local algorithms in all test scenarios. In particular, among the two local algorithms, PWL-LOCAL_MP_UMMF algorithm had even much fewer MCF calls than ϵ-LOCAL_MP_UMMF.

Furthermore, the complexity reduction of local algorithms over the optimal algorithm was greatly increased as the problem getting more complex. For example, with link capacity 250 Mb/s, the number of MCF calls of OPT, LOC and PWL were 917, 324 and 195, respectively. As we explained in Section 2.6, the ϵ-OPT_MP_UMMF algorithm has much more MCF calls because it uses MCF solver to test the saturation of each commodity. While the ϵ-LOCAL_MP_UMMF local algorithm does not have this expensive saturation test, it still makes multi-

Table 2.3: Running time and complexity comparison of optical and local multi-path utility max-min algorithms.

Network	Link Capacity	# of MCF function calls			Running Time (seconds)		
		OPT	LOC	PWL	OPT	LOC	PWL
Abilene 11 nodes 28 links	1 Gb/s	639	249	177	62.21	14.71	7.18
	500 Mb/s	696	218	192	67.52	11.73	7.64
	250 Mb/s	917	324	195	94.53	15.83	7.69
GEANT 23 nodes 74 links	38 Mb/s ≀ 2.5Gb/s	2032	882	478	60h31m	3h57m	1h2m

ple MCF calls from the binary search step in each iteration. In contrast, PWL-LOCAL_MP_UMMF algorithm only makes exactly one MCF calls in each iteration and the number of iterations is bounded by the number of segments and commodities. Therefore, PWL-LOCAL_MP_UMMF algorithm has the lowest complexity, and its complexity grows much slower than the others. As expected, the algorithm complexity directly affects the running time of algorithms. Thus, PWL-LOCAL_MP_UMMF algorithm also had much less running time. For example, under link capacity 250 Mb/s, PWL-LOCAL_MP_UMMF significantly reduces the running time of the approximated optimal and local algorithms by 91.87% and 51.42%, respectively.

Finally, while it is still feasible to use optimal algorithm for a small network like Abilene, we next show that the necessities of local algorithms for larger networks, such as GEANT [44]. GEANT is a European public network with 23 nodes connected by 74 links varied from 155 Mb/s to 10 Gb/s. Similar to the Abilene experiments, we build the utility of commodities from the historical traffic measurements [86] and compute the corresponding utility max-min bandwidth allocation. With the network size double, we not only have 4 times more number of commodities, the number of possible routing paths is also greatly increased. As a result, the running time significantly increases to around 60 hours, 4 hours and 1 hour for algorithm OPT, LOC and PWL, respectively. Therefore, as we can imagine, a huge commercial core network with hundreds of nodes and links would require a local algorithm like PWL-LOCAL_MP_UMMF algorithm to compute a relatively effective multi-path allocation within reasonable time.

2.8 Conclusion

We believe finding a multi-path utility max-min bandwidth allocation is fundamental and crucial for several network problems and applications, such as traffic engineering. As showing from our statistical traffic engineering application example, multi-path utility max-min algorithms are able to achieve higher utility, and this higher utility translates to significant performance improvements. Furthermore, with the rapidly increasing traffic and disproportional expensive of faster processor, multi-path transmissions may become a fundamental part of the Future Internet.

This chapter, for the first time, defines the utility max-min problem under multi-path routing and presents both optimal and local algorithms that can be efficiently implemented using a linear program solver. However, there are still many interesting questions on multi-path utility max-min remain unanswered. We consider this work as the first step toward this important problem and fully expect to have more efficient and accurate approximated algorithms for the optimal solution in the future. Last but not least, the scope of this chapter is limited in theoretical results. Thus, for practical implementation, we might want a distributed or online local algorithm. Also, we have to consider the packet re-ordering issue under multi-path routing.

Chapter 2, in full, is currently being prepared for submission for publication of the material. Chou, Jerry; Lin, Bill. The dissertation author was the primary investigator and author of this material.

Chapter 3

Adaptive Rerouting Mechanism

3.1 Introduction

This chapter describes our adaptive rerouting mechanism of COPLAR to dynamically handle burst and unexpected traffic exceeding from the provisioned circuits. In general, adaptive routing algorithms are interested in the following problem. Given a set routing paths between each source and destination pair and a load depending link cost function, the question is how to split traffic among available paths, so that the network cost is minimized. Several adaptive routing algorithms [33, 49, 35] has been proposed, and majority of their works were focus on the optimization of network cost and converge time of the algorithm. Therefore, it is not our interest to propose a new adaptive routing algorithm. Instead, our goal is to apply an adaptive routing algorithm on COPLAR and evaluate the performance improvement of COPLAR.

However, as mentioned in Chapter 1, it is not trivial to apply an adaptive routing algorithm on COPLAR, and this chapter would address three main challenges. First is to explore routing path diversity without routing information explosion and routing loop. Second is to compute the cost of path under our own routing infrastructure. Third is to design an appropriate cost function for the adaptive routing algorithm, so that when the algorithm reaches its optimal value, our network performance objective could also be achieved. Therefore, while the goal of an adaptive routing algorithm is to optimize the path cost of any giving

routing paths and cost function, our goal is to find the appropriate routing paths and cost function to achieve the objective of our network problem. Therefore, our work on adaptive routing is complement to those previous works on adaptive routing algorithm [33, 49, 35].

Specifically, in this chapter we consider a game theory adaptive routing algorithm based on Wardrop equilibrium. We choose this game theory based adaptive routing algorithm for several reasons. First, the Wardrop equilibrium is defined under selfish routing model [90], thus no synchronization or coordination is required among routers. Second, it is proven in [36] that Wardrop equilibria could be reached by a fast converge time algorithm. Third, Wardrop equilibrium routing algorithm does not require a path-centric routing infrastructure. It is a crucial factor to our problem because we would like to explore as much path diversity as possible in our COPLAR rerouting scheme. As a result, it could be infeasible to setup all possible routing paths for a large network using existing routing technology, such as MPLS (Multi Protocol Label Switching).

In the rest of the chapter, we first review previous work on adaptive routing in Section 3.2 and background of Wardrop equilibrium in Section 3.3. Then we address the three challenges of adaptive rerouting on COPLAR in Section 3.4 to Section 3.6. Section 3.4 proposes loop-free routing table structures, Section 3.5 proposes propagation protocol for computing path cost. Section 3.6 design the cost function for adaptive routing algorithm. Finally, the chapter concludes in Section 3.7

3.2 Related Work

In this section, we review some existing adaptive routing algorithms and theories. Majority of the work on adaptive routing [83] were proposed in theory in the sense that the entire network architecture has to be replaced and results were limited to off-line algorithm analysis. Until recently, approaches that allow automatic online traffic engineering on top of an existing routing infrastructure were proposed, such as MATE [33], TeXCP [49] and Replex [35]. All these routing

mechanisms aim to optimize network cost by adaptively selecting routing paths for traffic from source (ingress router) to destination (egress router), but all of them are based on a different theoretical analysis to guarantee proven performance and converge time.

MATE (MPLS Adaptive Traffic Engineering) is an end-to-end control protocol that does not require intermediate nodes to perform any action besides normal packet forwarding. But, it does rely on sending probe packets through each path to monitor the current cost, such as packet delay or packet loss. MATE is implemented as an on-line optimization algorithm based on the Kuhn-Tucker theorem [22]. According to the theorem, if we adjust traffic based on the derivative length of paths which can be computed from the path cost, the network system would eventually reach its system optimal where the total network cost is minimized. Different from the end-to-end approach of MATE, TeXCP uses a lightweight prob message to receive explicit feedback from intermediate routers/nodes and discover path utilization. Traffic is moved adaptively from over-utilized path to under-utilized path, and the maximum link utilization can be minimized. The traffic adjustment algorithm and strategy is similar to the XCP protocol for congestion control. Through the feedback message, intermediate routers/nodes are able to coordinate the traffic from different ingress routers and prevent traffic oscillation among these paths. Since feedback message is per path basis, routing path setting like MPLS is required. In contrast to MATE and TeXCP, Replex is not restricted to path-centric scenarios, such as MPLS (Multi Protocol Label Switching). The foundation theory of Replex is selfish routing. Thus, there is no coordination among routers besides aggregating cost along routing paths. Although the adaptive routing algorithm discussed in this chapter is similar to Replex, our focus is to apply adaptive routing rather than propose the routing algorithm as mentioned in Section 3.1.

Selfish routing is an attractive routing model to be considered for adaptive routing because it assumes no coordination and regulation among senders. Such model is well known to be formulated as a game theory problem [68] and people are interested in the existence of an equilibrium state, Nash equilibria, and the

bound between the cost of selfish routing and optimal routing. The problem has also been extensively studied in different area of networking with different cost models. For example, [55, 82] discussed the existence of equilibrium state in a flow control model at a router or switch. [56] proved Nash equilibrium is optimal for a two-node multiple links network. In particular, considering selfish routing on a general network routing problem has been studied since [90, 20]. The network reaches the equilibrium state when all paths have equal cost, and it is known as the Wardrop Equilibria [90]. [78] proved that if the cost of each edge is linear to its congestion, then the total network cost under the routes chosen by selfish routing is at more 4/3 times the optimal value. Furthermore, if the cost function is continuous and non-decreasing to the edge congestion, the results of selfish routing is no more than the cost incurred by optimally routing twice as much traffic.

3.3 Background

Here reviews the definitions and routing policy of Wardrop equilibrium.

3.3.1 Wardrop Equilibrium of Selfish Routing

Considering a network $G = (V, E)$ with a set of commodities $[k] = 1, \cdots, k$ specified by a source s_i, a destination t_i and a total flow demand d_i. Each commodity i is given a set of all possible routing paths P_i. Wardrop traffic model [90] assumes an infinite number of selfish agent, and each of them wants to send an infinitesimal amount of traffic through a network. Each agent belongs to a single commodity i and sends its traffic from s_i to d_i by using one of the paths $p \in P_i$. Therefore, the strategy space of the game is a flow vector f_p which represents the amount of traffic/agent routed on each path $p \in P_i, \forall i$. A flow vector is only considered as feasible or valid when $\sum_{\forall p \in P_i} f_p = d_i$. This flow assignment in turn induces cost on edges as the following. Each link $e \in E$ is associated with a non-decreasing cost function $\ell_e(f_e)$ where f_e denotes the amount of traffic on edge e. The cost of a path p is $\ell_p = agg_{\forall e \in p} \ell_e(f_e)$ where agg is some aggregation function. Since all agents selfishly route their traffic on path with lower cost, the network eventually

Algorithm 1 The $(\alpha - \beta)$-exploration-replication policy

Every time interval T, each commodity i performs:
1. Path sampling: Perform step (a) with probability β, and perform (b) with probability $1 - \beta$.
   (a) Uniform sampling:
   Pick a path $Q \in P_i$ with probability $1/|P_i|$.
   (b) Proportional sampling:
   Pick a path $Q \in P_i$ with probability f_Q/d_i.
2. Path migration: If $\ell_Q < \ell_P$, migrate from P to Q with probability $\frac{\ell_P - \ell_Q}{\ell_P + \alpha}$.

reaches a stead state called Wardrop equilibrium when all paths $p \in P_i \forall i$ have the same cost. Thus there is no incentive for any traffic/agent to change its routing assignment. The formal definition of Wardrop equilibrium is given in Definition 12

Definition 12 (Wardrop equilibrium [90]). *A feasible flow vector f is at Wardrop equilibrium if for each commodity $i \in [k]$ and each path $p \in P_i$ with $f_p > 0$ it holds that $\ell_p(f) \leq \ell'_p(f)$ for all $p' \in P_i$.*

3.3.2 $(\alpha - \beta)$-exploration-replication policy

The Wardrop equilibrium state can be reached quickly by a proven strategy called $(\alpha - \beta)$-exploration-replication policy [36]. The strategy is shown in Algorithm 1. Every interval T, agents activate with probability λ and simultaneously change their flow assignments based on an adaptive sampling method consisted of uniform and proportional sampling. The uniform sampling guarantees that every path has strictly positive sampling probably, and all paths in strategy space can be explored. On the other hand, the proportional sampling selects path proportional to traffic load, so that a better path would be preferred in the sampling process and converge time could be shorten. A path with higher load normally indicates a path with lower cost because traffic is routed selfishly. Finally, to avoid oscillation, traffic migrates from a higher cost path P to a lower cost Q with probability $\frac{\ell_P - \ell_Q}{\ell_P + \alpha}$ where α is some parameter to prevent the migration probability to become infinite. Therefore, given a set of routing paths P_i for a commodity i. If we know the

current split ratio of path w_p (i.e. the percentage of traffic of commodity i routed on path p) as well as the cost of path ℓ_p, Wardrop equilibrium can be reached by periodically adjusting the split ratio between each pair of paths P and $Q \in P_i$, such that when $\ell_Q < \ell_P$:

$$\triangle = \lambda[(\frac{\beta}{|P_i|} + (1-\beta)w_Q) \cdot \frac{\ell_P - \ell_Q}{\ell_P + \alpha}]$$

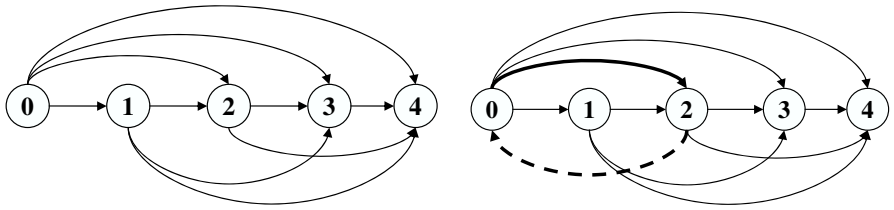
$$w_P = w_P - \triangle, w_Q = w_Q + \triangle.$$

3.4 Loop-free Routing Table Construction

As mentioned in Section 3.1, our first challenge is to explore path diversity for adaptive rerouting without forming routing loop by constructing loop-free routing tables. The basic idea of loop-free routing tables is to limit the path selections during rerouting process, so that a packet cannot select a node that has been visited before as its next intermediate node. At the same time, these loop-free routing tables should still preserve enough path diversities to allow the adaptive load-balance rerouting algorithm to have sufficient dynamic routing ability to explore the available residual circuit capacities in a network. In this section, we propose two methods to construct loop-free routing tables. The first one is based on maximum directed acyclic graph and the second one is based on ring topology.

3.4.1 Acyclic graph loop-free routing table

Our goal is to build a routing table for each IE pair (s,t) on every node, each routing entry on node i indicates a possible next forwarding node from i and t for the traffic of IE pair (s,t). Thus, assigning a routing entry is exactly the same as giving a direction to the corresponding outgoing circuit/edge on the network graph. As a result, constructing loop-free routing tables is the same as finding a directed acyclic subgraph on our provisioned circuits, so that any routing path that follows these directed circuits are loop-free. More specifically, to explore the maximum path diversity, we want to find the maximum directed acyclic graph [40], which is the directed acyclic subgraph with the most edges.



(a) A example of directed acyclic graph of a network with 5 nodes.

(b) A routing loop occurs, if a directed edge is added from node 2 to 0.

Figure 3.1: Illustration of directed acyclic graph for loop-free routing.

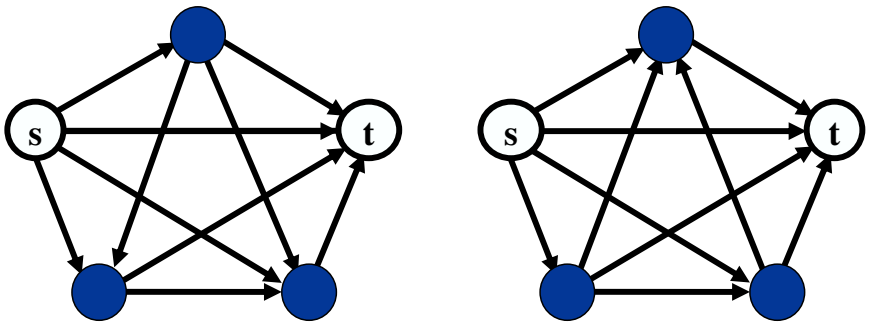


Figure 3.2: Multiple possible acyclic graph solutions.

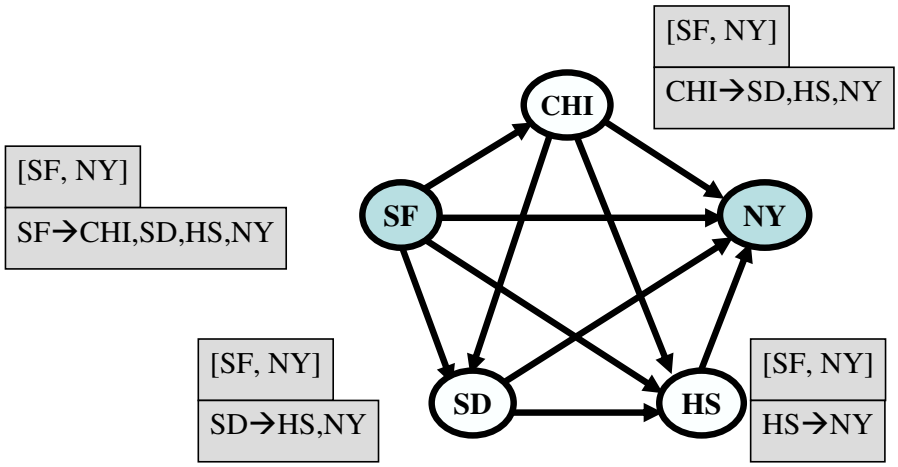


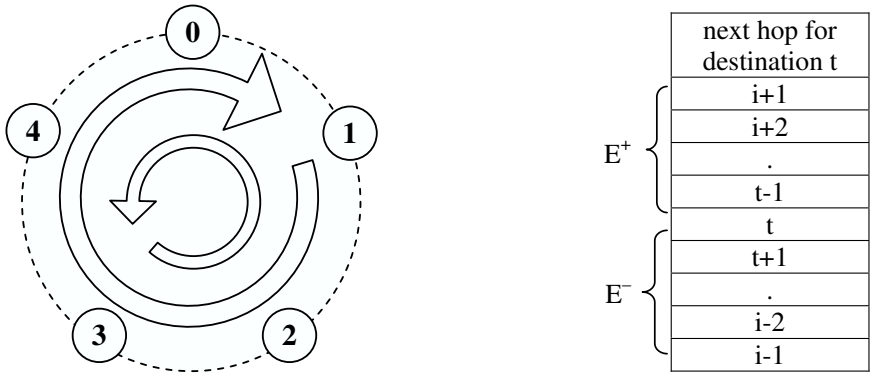
Figure 3.3: An routing table example from SF to NY constructed based on an acyclic graph.

Computing a maximum directed acyclic subgraph on an arbitrary graph has been shown to be NP-Complete [40]. However, in our problem, the underlying graph is a fully connected mesh because our circuit provisioning algorithm setups a circuit between each ingress and egress router pair. In this special case, as indicated in [40], for any IE-pair (s, t), its maximum acyclic subgraph can be found by inducing any permutation order of nodes starting with s and ending with t. For example if we have a network with 5 nodes and use the the node list from 0 to 4 for IE-pair (0,4), its maximum acyclic subgraph is shown in Figure 3.1 (a). We can easily prove that the directed acyclic graph in Figure 3.1 (a) is also the maximum directed acyclic graph because there is a directed edge connects every pair of nodes from left to right. If we add one more arbitrary directed edge into the graph as shown in Figure 3.1 (b), it must connect a node from right to left. As a result, any new directed edge will definitely create a cycle or routing loop like between node 0 and 2 in our example. Since there could be multiple acyclic graph solutions as shown in Figure 3.2, we use the one with the maximum circuit capacity from an ingress router s to an egress router t. Then Figure 3.3 shows an example of our routing table construction.

3.4.2 Ring-based loop-free routing table

Although acyclic loop-free routing tables give us the maximum path diversity, it creates a set of routing tables for each IE-pair. Thus each node will have $O(N^2)$ routing tables instead of $O(N)$ where N is the size of a network. The larger number of routing tables could consume more storage space on router. More important, it would require higher computational and communication cost to maintain the routing information in routing tables as discussed more detailed in Section 3.5.1. Therefore, in the following, we proposed another loop-free routing tables strategy based on a ring graph.

The basic idea is to locate nodes in a ring topology and label them in order as showing in Figure 3.4 (a). Clearly, there is no routing loop if we limit traffic to travel either in a clockwise or counter-clockwise direction on the ring. Furthermore, since all packets having the same destination and traveling in the same direction



(a) Loop free routing on a ring topology. (b) Routing table on node i for destination t.

Figure 3.4: The loop-free routing paths and routing tables based on a ring topology. Any packet is either be forwarded in clockwise or counter-clockwise direction by their corresponding routing entries E^+ and E^- in a routing table.

are using the same routing paths, we only need $O(N)$ routing tables per node with two routing tables per destination. In fact, we could even combine the two routing tables into one because no forwarding node can be in both clockwise and counter-clockwise direction except the destination. Therefore, Figure 3.4 (b) represents the routing table for any destination t on node i where E^+ is a set of clockwise routing entries and E^- is a set of counter-clockwise routing entries. When a packet arrives, a node would either select the next forwarding from E^+ or E^- depending on the location of the source. For example, if a packet of IE-pair (0, 4) arrives node 2, we know the packet is traveling in clockwise direction and must be routed by an entry in E^+ which includes node 3 and 4. Therefore, once a packet decides to route in a clockwise/counter-clockwise direction, it would always be forwarded in the same direction by the routing entries in E^+/E^- on every intermediate node without routing loop.

3.5 Propagation Protocol for Path Computation

Here, we would like to compute the path cost resulting from our loop-free routing table described in Section 3.4. In brief, we show the path cost can be

computed as a dynamic programming program and implemented as a on-line computation with the help of a propagation protocol. In particular, we first consider the acyclic loop-free routing table structure. Then simplify the results to the ring-based loop-free routing table structure.

3.5.1 Propagation protocol for acyclic loop-free table

With acyclic loop-free routing table, every node i has a routing entry for every IE-pair (s,t) to next hop j. Each routing entry is associated two values $\mathbf{W}_{ij}^{st}$ and $\mathbf{C}_{ij}^{st}$ where $\mathbf{W}_{ij}^{st}$ is the split ratio for node i to choose node j as the next hop for commodity (s,t), and $\mathbf{C}_{ij}^{st}$ is the corresponding path cost to send a packet to destination t through the next hop j for commodity (s,t). Let ℓ_{ij} be the link cost of edge from i to j. Our goal is to compute $\mathbf{C}_{ij}^{st}$ with the given value of $\mathbf{W}_{ij}^{st}$ and ℓ_{ij}. Let C_i^{st} be the path cost from any intermediate node i to destination t for commodity (s,t). Clearly, we know $\mathbf{C}_{ij}^{st} = \ell_{ij} + C_j^{st}$. Thus, we first show how to compute C_j^{st} in Theorem 4.

Theorem 4. *Assume we use node list (n_{N-1}, n_{N-2}, $\cdots$, n_1, n_0) to construct the loop-free routing tables for a commodity (s,t) where $n_{N-1} = s$ and $n_0 = t$. The path cost C_i^{st} from intermediate node i to destination t for commodity (s,t) is $\sum_{j=n_0 \to n_{i-1}} \mathbf{W}_{ij}^{st}(\ell_{ij} + C_j^{st})$ for any $i < N$.*

Proof. We iteratively compute C_i^{st} from $i = 0$ until i reaches the source node n_{N-1}. For simplicity, we use i to denote n_i in the following equations.

We already know $C_0^{st} = 0$ because there is no cost if source and destination are the same node.

When $i = 1$, n_1 can only directly forward traffic to destination n_0. Thus the path cost is equal to the link cost from n_1 to n_0 and

$$C_1^{st} = \mathbf{W}_{1,0}^{st}\ell_{1,0}.$$

When $i = 2$, n_2 could directly send to n_0 or forward to n_1 as an intermediate to n_0. Since we already computed the cost from n_1 to n_0, C_1^{st}, in previous iteration,

the cost from n_2 to n_0 through n_1 is $\ell_{2,1} + C_1^{st}$. Thus the total path cost from n_2 to n_0 is

$$C_2^{st} = \mathbf{W}_{2,0}^{st}\ell_{2,0} + \mathbf{W}_{2,1}^{st}(\ell_{2,1} + C_1^{st}).$$

Similarly, when $i = 3$, n_3 could forward traffic to n_2, n_1 or n_0. As we already computed the path cost from n_2 to n_0 and n_1 to n_0, the total path cost from n_3 to n_0 is

$$C_3^{st} = \mathbf{W}_{3,0}^{st}\ell_{3,0} + \mathbf{W}_{3,2}^{st}(\ell_{3,2} + C_2^{st}) + \mathbf{W}_{3,1}^{st}(\ell_{3,1} + C_1^{st}).$$

By repeating the same cost computation for i from 4 to $k-1$, the total path cost from n_k to n_0 is computed as

$$C_k^{st} = \sum_{j=n_0 \to n_{k-1}} \mathbf{W}_{kj}^{st}(\ell_{kj} + C_j^{st}).$$

□

Although any node i could compute its path cost $\mathbf{C}_{ij}^{st}$ with the full knowledge of the link cost and routing information from other nodes, it could be both communication and computation costly. This is due to the fact that each node has $O(N^2)$ routing tables and each routing table has $O(N)$ entries. If there are 100 nodes and a routing information $\mathbf{W}$ is stored as double, each node has to broadcast a message size with 128 Mb ($100^3 * 128$ bits) to every other nodes. Furthermore, redundant computation occurs among nodes because the value of C_k^{st} is actually the same for every nodes according to Theorem 4. Therefore, instead, we propose a online dynamic programming solution with the help of a propagation protocol for path computation. As shown in Theorem 4, C_i^{st} is computed from a series of $C_k^{st}, \forall k < i$ where node k only needs its local link cost measurements, $\ell_{kj}, \forall j$, local routing information, $\mathbf{W}_{kj}^{st}, \forall s, t, j$, and the path cost, $C_j^{st}, \forall s, t$, from its neighbor j. Therefore, the propagation protocol for our acyclic loop-free routing table structure is shown in Algorithm 2.

Algorithm 2 Acyclic graph propagation protocol

Every time interval T, each node i performs:

1. Receiving path cost C_j^{st} from each neighbor j.
2. Measure link cost ℓ_{ij} to each neighbor j.
3. Update cost in routing table $\forall$ IE-pair (s,t), neighbor j:
   $\mathbf{C}_{ij}^{st} = \ell_{ij} + C_j^{st}$.
4. Compute path cost for each IE-pair (s,t):
   $C_i^{st} = \sum_{\forall j \in \mathbf{E}_i^{st}} \mathbf{W}_{ij}^{st}(\ell_{ij} + C_j^{st})$
5. Broadcast path cost $C_i^{st}, \forall s,t$, to each neighbor j.

Algorithm 3 Ring-based propagation protocol

Every time interval T, each node i performs:

1. Receiving cost C_j^{t+} and C_j^{t-} from each neighbor j.
2. Measure link cost ℓ_{ij} to each neighbor j.
3. Update cost in routing table $\forall$ destination t, neighbor j:
   If $j \in \mathbf{E}_i^{t+}, \mathbf{C}_{ij}^{t} = \ell_{ij} + C_j^{t+}$.
   Otherwise, $\mathbf{C}_{ij}^{t} = \ell_{ij} + C_j^{t+}$.
4. Compute path cost for each destination t:
   $C_i^{t+} = \sum_{\forall j \in \mathbf{E}_i^{t+}} \mathbf{W}_{ij}^{t}(\ell_{ij} + C_j^{t+})$
   $C_i^{t-} = \sum_{\forall j \in \mathbf{E}_i^{t-}} \mathbf{W}_{ij}^{t}(\ell_{ij} + C_j^{t-})$
5. Broadcast C_i^{t+} and $C_i^{t-}, \forall t$, to each neighbor j.

3.5.2 Propagation for ring-based loop-free table

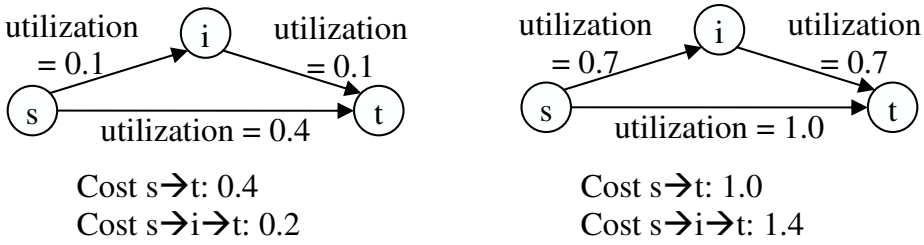
With ring-based loop-free routing table structure, we are able to reduce the complexity of path cost computation and the size of propagation message for two reasons. First, ring-based structure has fewer routing tables. Instead of computing the path cost for each commodity (s,t), $C_i^{st} \forall s,t,i$, here we only need to know the cost of clockwise path from i to t, C_i^{t+}, and the cost of counter-clockwise path from i to t, C_i^{t-}, for all destination t and intermediate node i. Second, as we mentioned in Section 3.4.2, once a packet decides to route in the clockwise or counter-clockwise direction at its source node or ingress router, it will always be forwarded in the same direction afterwards. In other words, the cost of C_i^{t+} only depends on the cost C_k^{t+} for all k in between i and t, and the same relation could be found for C_i^{t-}. Thus, similar to the proof in Theorem 4, we could compute the path cost of $C_i^{t+} = \sum_{j=n_0 \to n_{i-1}} \mathbf{W}_{ij}^{t}(\ell_{ij} + C_j^{t+})$ where $(n_i, n_i - 1, \cdots n_0)$ are the nodes from i to t in the clockwise direction, as well as the path cost of C_i^{t-}. As a result, comparing

to the computation for acyclic loop-free structure, each node only computes and propagates the clockwise and counter-clockwise cost for each destination rather than every pair of nodes. The detailed description of the propagation protocol for ring-based structure is shown in Algorithm 3.

3.6 Cost Function Assignment

After we introduce the policy to adjust split ratio of rerouting paths and the propagation protocol to compute path cost, this section discusses the cost function of links. Our goal is find an appropriate cost function that can achieve the objective of our new routing paradigm. That is to maximize network throughput and minimize router load or O/E/O conversions. We start with a naive cost function using link utilization to demonstrate the importance of cost function. Then we design our own *escalated step* cost function to achieve our network performance objectives. The cost function could be modeled as the delay of path where path delay could be either measured a ping-like mechanism or estimated by average queuing delay. Another solution is to use an escalated step function as described in this section.

Figure 3.5 illustrates why it is not trivial to design a cost function for our network performance objectives. The dummy network in the example has a direct path and a two-hop path from source s to destination t. Assume we simply use the sum of link utilization as our cost function. Intuitively, it would prefer the path with lower utilization and fewer links. Thus, the network throughput could be increased and O/E/O conversion could be reduced. However, if the network is under utilized as shown in Figure 3.5 (a), the cost of direct path could be easily as twice as much as the two-hop path. As a result, we would reroute more traffic to two-hop path, even though there is clearly plenty of residual capacity left on the direct path. Thus, unnecessary O/E/O conversions or hops occur from the adaptive routing. On the other hand, if the network is highly utilized as shown in Figure 3.5 (b), the cost of two-hop path could be much higher than the direct path. In consequence, even the direct path is already congested, our adaptive



(a) Traffic shifts to alternated path and increases unnecessary O/E/O conversions.

(b) Traffic shifts to already congested direct path and reduces network throughput.

Figure 3.5: Example of using link utilization as cost function.

routing algorithm would still prefer the direct path and could not increase network throughput using the two-hop path as we wanted. Therefore, regardless what adaptive routing algorithm we are using, if we don't have an appropriate cost function, undesired network performance could occur.

To achieve the network performance objectives, our approach is to design a *escalated step* cost function that can explicitly control the condition of using a longer path. The intuition is that we should always prefer the direct path when it is un-congested. However, as the direct link becomes congested, we would start shifting traffic to alternated longer paths. More specifically, the lengths of the alternated path should be limited by the degree of congestion on direct path. For example, if the link utilization of the direct path under 80%, we would only use one-hop path which is exactly the direct path itself. If the link utilization of the direct path above 80% but under 90%, then we will reroute traffic on the two-hop path as well. Finally, we will only reroute traffic on five-hops path when the direct path is extremely congested such as over 98% link utilization. In other words, the cost function has to explicitly capture the relation between path length and link utilization.

According to the above intuition, we design the *escalated step* cost function based on a vector of increasing link utilization values $U = (u_2, u_3, \cdots, u_k)$. Our goal is to assign a cost c_i for each of the utilization value u_i, such that the value of u_i indicates the minimum utilization requirement to use an i hops path. Before

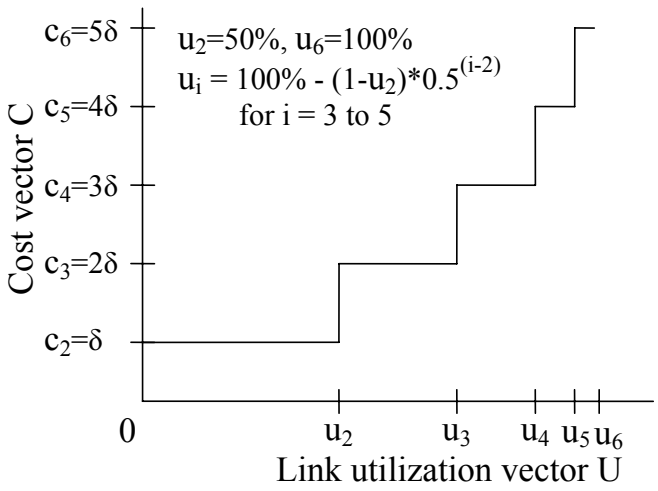
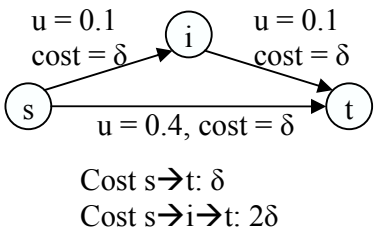
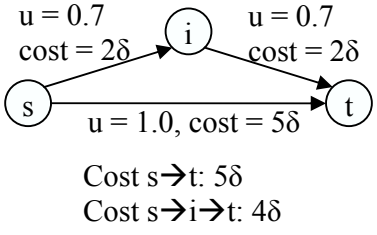


Figure 3.6: Proposed *escalated step* cost function based on a vector of link utilization U.

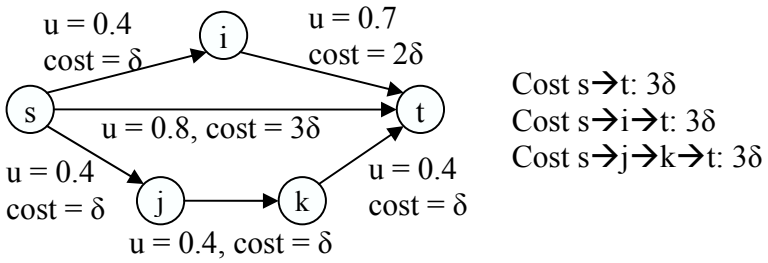
describing how to assign the corresponding cost vector $C = (c_2, c_3, \cdots, c_k)$ for U, we observe several advantages of such cost function. First, it allows network operators to determine when to use longer paths. For example, a network with more powerful router could have lower vector values, so that there is more path diversity to load balance traffic and increase network throughput. On the other hand, if a network doesn't have enough computation power or queuing space on routers, the network could set higher vector values to prevent rerouting and limit router load. Second, the value of u_2 explicitly restricts the minimum link utilization for rerouting, while the length of the vector k is the maximum number of hops or O/E/O conversions allowed. Thus a network operator could easily control the two most critical parameters for our adaptive rerouting algorithm. Last but not least, the value of vector U does not necessary to be link utilization. It could also be any network metric, such as queuing delay, latency delay, etc. However, since our primary goal is to improve network throughput, link utilization is a better indication for possibly packet drops. Also, the value of link utilization has a more general implication because its value must be in between 0% and 100%. In contrast, the value of delay is more meaningful respect to a specific network size or application requirements.



(a) Traffic shifts to direct path and reduces unnecessary O/E/O conversions.



(b) Traffic shifts to alternated path and increases network throughput.



(c) With cost of direct path is 3δ, we could either use a path with three un-congested links ($< u_2$) or a path with one un-congested links and one low-congested link ($< u_3$).

Figure 3.7: Example of using our proposed *escalated step* cost function shown in Figure 3.6 with $U = (50\%, 75\%, 87.5\%, 93.75\%, 100)$.

To design our cost function, we model it as a step function based on vectors U and C as shown in Figure 3.6. Thus, if the utilization of a link e is u and $u < u_i$, the cost of the link is $\ell_e = c_i$. Considering a one-hop direct path P and a alternated multi-hops path P', we start assign the cost vector C from c_2 to c_k. Assume we set the cost of c_2 to be some constant δ. When the link utilization of P is less than u_2, the cost of P' can never be smaller than P because P' is consisted of at least two links and the cost of each link is at least c_2. In other words, $\ell_P = u_2 < 2 * u_2 < \ell_{P'}$. Therefore, we would always prefer direct path when the link congestion is less u_2. When the link utilization of P is less than u_3, we have to set c_3 small enough, so that any alternated path P' with more than three hops has higher cost than c_3. Again because the minimum cost of a link is δ, c_3 should be at least 2δ. Similarly, if we don't want any alternated path P' with i hops has lower cost than our direct path with utilization less than u_i, c_i must be at least $(i-1)\delta$. Therefore, it is

actually rather easy to set the cost of our vector C as $c_i = (i-1)\delta, \forall i \in [2,k]$ as shown in Figure 3.6.

Finally, Figure 3.7 re-examines the same network scenario of Figure 3.5, but using our proposed cost function with the utilization vector U=(50%,75%,87.5%, 93.75%,100%) shown in Figure 3.6. When network is under utilized, Figure 3.7 (a) shows that all traffic would be routed on direct path as long as the link utilization is lower than 50%. On the other hand, when network is congested as Figure 3.7 (b), our cost function would have higher cost for direct path and reroute traffic to alternated path, so that the network throughput can be increased. Finally, since we assign a fixed cost difference among each level of link congestion, when ever the congestion of direct path increased by one level, we could either use an alternated path with one more hop or a link with more congested link. For example, in Figure 3.7 (c), the link utilization of direct path is increased from δ to 3δ. Therefore, we could also reroute traffic on a path consisted of three links with minimum cost, or a path with only two links while one of them is more congested.

3.7 Conclusion

In this chapter, we address how to apply a game theory based adaptive routing algorithm to the rerouting mechanism of COPLAR. Specifically, we propose two loop-free routing table structures based on acyclic graph and ring topology to fully explore path diversity without creating routing loop. We also give the propagation protocol to compute path cost online without broadcasting routing information. Finally, we demonstrate the criticality of cost function to our COPLAR rerouting mechanism, and propose escalated step function to explicitly capture the relation between path length and network congestion, so that the network throughput could be maximized while O/E/O conversion could be minimized. The in-depth evaluations of proposed approach are giving in Chapter 4.

Chapter 4

Evaluation of COPLAR

Our COPLAR approach was extensively evaluated using a well-known realistic network simulator, NS2 on two real large PoP-level backbone networks, namely Abilene [2] and GEANT [44], with real traffic trace data over two months. We first describes the detail of our simulation setup and algorithm parameters in Section 4.1 followed by the evaluation results of COPLAR in three aspects. Section 4.2 shows the effectiveness of our circuit provisioning. Section 4.3 analyzes the design of our rerouting algorithm. Finally, we demonstrate the overall performance of COPLAR by comparing with the two well-known packet switching routing, OSPF and ECMP, in Section 4.4. The chapter is concluded in Section 4.5.

4.1 Experimental Setup

We used NS2 simulator to extensively evaluate our COPLAR routing paradigm using the adaptive rerouting scheme described in Chapter 3 on two separate real large PoP (point of presence)-level backbone networks, namely Abilene [2] and GEANT [44]. Abilene is a public academic network in the US with 11 nodes interconnected by OC192, 10 Gb/s, links. Its network topology has been shown in Figure 2.5. GEANT is a network connects with variety of research and education networks in Europe. The topology of GEANT has been slightly changed and evolved in the past few years. Our experiments were based on an December 2004

Table 4.1: The traffic matrices information for Abilene and GEANT.

Network	Circuit provision history traffic matrix	Simulated traffic matrix	Time Interval
Abilene	03/01/04−04/21/04	04/22/04-04/26/04	5 minutes
GEANT	01/01/05−04/10/05	04/11/05-04/15/05	15 minutes

network snapshot [1], in which has 23 nodes and 74 links varied from 155 Mb/s to 10 Gb/s. Both networks have been studied and discussed in previous research literatures, and their datasets, including network topology, routing information, and traffic measurements, are available in the public domain. In the following, we first explain the traffic matrices used in the experiments. Then we describe our implementation of adaptive rerouting algorithm in NS2. Finally, we summarize the routing algorithms compared in our evaluation and their implementation detail.

4.1.1 Traffic Matrices

Traffic matrices are used in our experiments both for deriving circuit configurations using the multi-path utility max-min allocation algorithms described in [27, 28] as well as for creating simulation traffic for our performance evaluations. A traffic matrix consists of the demand rate (kb/s) of every IE pair (Ingress-Egress router pair) within a certain time interval (e.g. 5 minutes for Abilene and 15 minutes for GEANT). For both networks, we used the real traffic matrices provided from a third party [87]. The traffic matrix datasets for the Abilene network are available at [100], and the traffic matrix datasets of the GEANT network are available at [86].

In brief, these traffic matrices are derived based on the flow information collected from key locations of a network by traffic monitors, such as netflow [1]. Then the flow information is transformed into the demand rate of each IE pair in a traffic matrix based on the routing information in the network. We collected the traffic matrices in each network for an extended period of time to represent the historical traffic measurements and simulation traffic load. The detail information

[1]The shapshot is available at `http://www.geant.net/upload/pdf/GEANT\_Topology\_12-2004.pdf`.

Table 4.2: Parameters of the game theory adaptive rerouting algorithm.

parameter	purpose	value
α	optimization parameter	0.1
β	uniform sampling ratio	0.75
λ	maximum weight shift	0.1
T	update interval	0.2s
U	utilization vector for cost function	$u_2 = 80$ $u_i = 1 - [(1 - u_2) * 0.5^{i-2}]$

of the traffic matrices used on two networks are summarized in Table 4.1.

Furthermore, as we know, current backbone networks are designed to be under utilized. To demonstrate network routing can improve network performance in a network with high utilization, we normalized our simulated traffic matrices by scaling their offering load by some factor, such that at least one traffic matrix had fully saturated links under OSPF routing. The scale factor for Abilene and GEANT were roughly equal to 4 and 2, respectively.

4.1.2 NS2 Implementation Detail of Routing Algorithms

Here summarizes the algorithms compared in this chapter.

Packet Switching - OSPF/ECMP

Open Shortest Path First (OSPF) is a dynamic routing protocol widely used in IP networks. It routes traffic through the path with minimum aggregated link weight. Extend from OSPF, Equal-Cost Multi-Path (ECMP) [8] is a Cisco router implementation of a multi-path load balancing scheme. In Abilene network, we used the actual OSPF link weight provided from [100] to determine the OSPF and ECMP paths. However, we don't have the link weight information for GEANT network. Thus, we simply use equal weight for all links in the GEANT network to determine its OSPF and ECMP paths.

COPLAR with Game Theory Algorithms

To simulate our circuit switching routing, we used network links to represent a set of virtual circuits. In other words, we re-established a fully connected mesh network with exactly one link between each IE pair, and the link capacity is the allocated bandwidth for the IE pair computed from from the circuit provisioning algorithm proposed in [27]. Therefore, the term link is interchangeable with circuit or virtual circuit in the following implementation description. Note that, for simplicity, we didn't create circuit at physical layer. Therefore, we didn't consider the finer grain traffic distribution among the physical circuits of an IE pair. We assume a router has the ability to fully utilize its virtual circuit capacity by load balancing the incoming traffic to all physical circuits.

We also implemented our own classifier module and routing module in NS2. Each classifier object represents a routing table. For simplicity we only consider ring-based loop-free routing table structure in our evaluation section, so we created a routing table for each destination on every node. A routing table contains a set of entries for all possible next hop for the destination, and each entries associated with a weight and cost for the outgoing path. When a packet arrives, we first find its routing table based on the destination, then randomly select a outgoing link/circuit from the table with the probability proportional to its weight. If the output queue of the selected link/circuit is full, the packet gets dropped. The routing module implemented the propagation protocol described in Figure 3 which periodically measure the cost of link and re-compute path cost. The path cost is then propagated to neighbors as a regular message. Therefore, the propagation information could also be dropped on congested links. Finally, other detailed setting of parameters used in the evaluation is summarized in Table 4.2.

COPLAR with Greedy Rerouting Algorithm

Finally, besides the game theory rerouting algorithm, we also consider a greedy rerouting scheme which always forwards traffic to its direct outgoing circuit until the output queue is 95% full. Once the output queue is full, a router simply load balances the excess traffic to all possible outgoing circuits proportional to the

residual capacity of the circuit. Comparing to the game theory, greedy approach only requires local information on the residual capacity of outgoing circuits. However, because the lacks of the path information, greedy algorithm could perform worse than the game theory algorithm as shown in Section 4.3.3.

4.1.3 Performance Metrics

Based on the performance objective of COPLAR, we evaluate network performance using the following metrics.

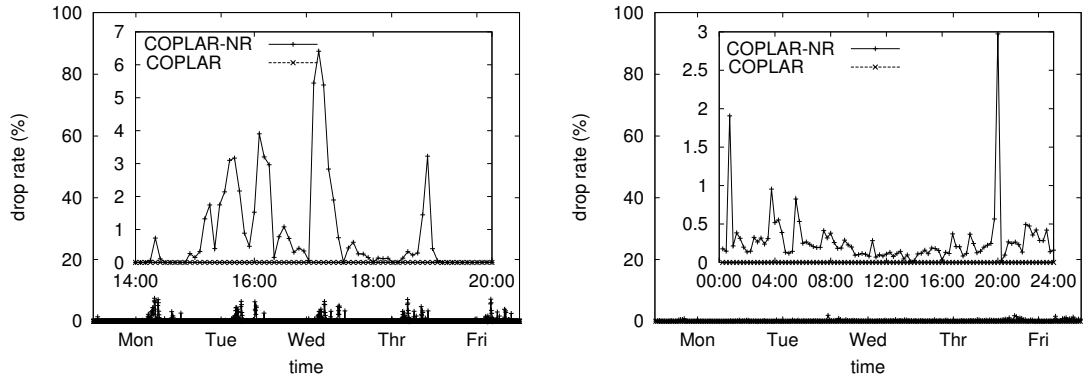
- Drop rate: the complement of network throughput.
- O/E/O conversion: the number of hops over virtual circuits for COPLAR, and the number of hops over physical links for packet switching approach like OSPF and ECMP.
- Router load: the amount of traffic required forwarding process at intermediate routers.

Our evaluations compare routing algorithms based on their performance metric values measured under each simulated time interval traffic matrix. Specifically, for a given traffic matrix of a time interval, we simulate the traffic under each routing algorithms and record its corresponding performance metric values over time. As shown in Figure 4.4, network states will converge and stabilized after about 10 seconds. Thus, we use the average number over 5 seconds after a warmup period of 25 seconds as the performance measurement of a given time interval, and the rest of chapter presents the results based on these performance measurements at each simulated time interval.

4.2 Circuit Provisioning Analysis

4.2.1 With vs. without rerouting

This section evaluates the performance of COPLAR using the real data traffic trace across a 5-day period (Monday to Friday). COPLAR re-configured its circuits

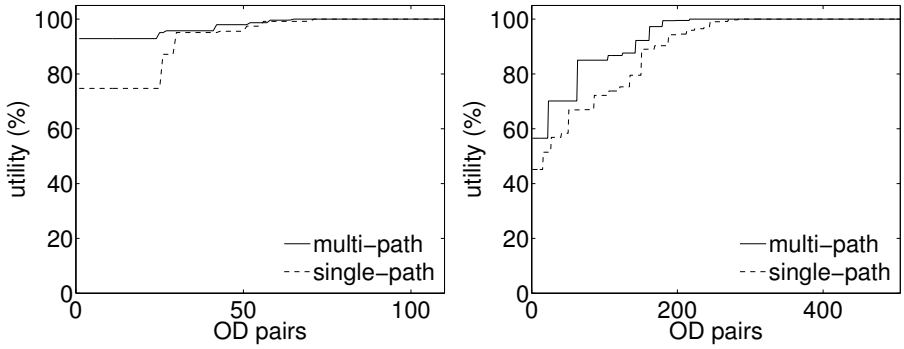


(a) Abilene (b) GEANT

Figure 4.1: Traffic drop rate across a week under COPLAR and COPLAR-NR. The inner graphs enlarge the results of Abilene from 2pm to 8pm on Wednesday and the results of GEANT on Wednesday.

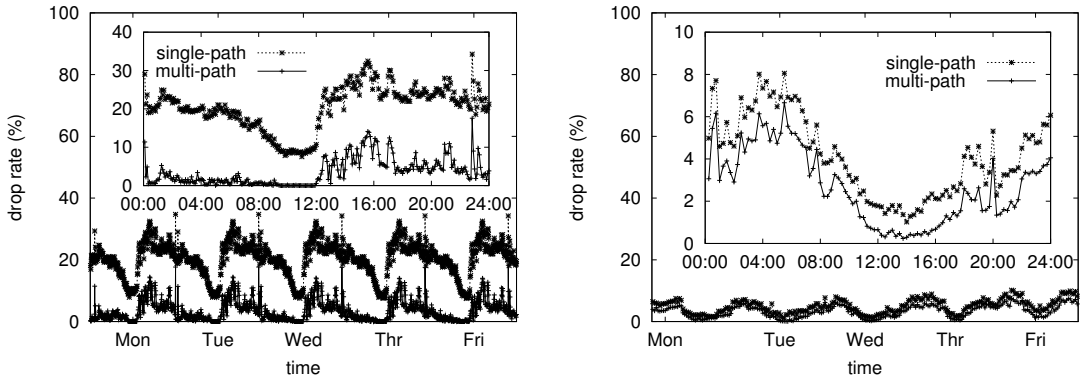
at each hour based on the historical traffic measurements. In particular, we compare the drop rate of COPLAR and COPLAR without rerouting scheme. COPLAR without rerouting is referred to as COPLAR-NR. Since the time granularity of traffic matrices in Abilene is 5 minutes, there are a total of 1,440 consecutive data points over the 5-day simulation period. Similarly, the GEANT network has 288 data points from its 15-minute traffic matrices.

The drop rates of each of the traffic matrix across 5-day simulation are reported in Figure 4.1. As shown in the figure, even without our rerouting scheme, COPLAR-NR achieves relatively low drop rates over all traffic matrices. In particular, the inner graphs of Figure 4.1 zoom in the drop rates on a particular day (Wednesday) and show the maximum drop rate of COPLAR-NR are only 7% and 3% in Abilene and GEANT, respectively. Therefore, our provisioning algorithm did effectively allocate bandwidth and minimize drop rate even in a congested network. Furthermore, by combining circuit configurations with rerouting, COPLAR is able to achieve 0% drop rates for all simulated traffic by utilizing the spare capacity of circuits.



(a) Abilene (b) GEANT

Figure 4.2: Utility achieved for each IE flow under single-path and multi-path utility max-min problem formulation.



(a) Abilene (b) GEANT

Figure 4.3: Traffic drop rate across a week under single-path and multi-path circuit configurations. The inner graphs enlarge the results on Wednesday for both networks.

4.2.2 Single-path vs. multi-path circuit provisioning

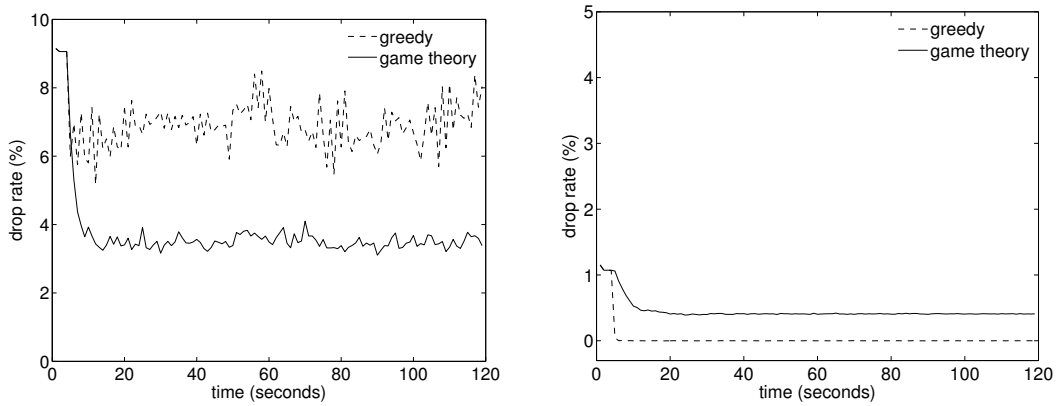
Here, we analyze the performance improvement from solving our circuit provisioning algorithm as a multi-path problem rather than a single-path problem. Because circuit provisioning plays a more important role when a network has higher utilization, and link capacity is more scarce, we present the circuit provisioning results under the normalized traffic matrices with a scale factor of 2 in this subsection. The multi-path solution was computed by the algorithm proposed in Chapter 2, while the single-path solution was derived by the traditional single-path "water-filling" algorithm [24, 79].

First, with a given network topology and utility functions of traffic distribution, we compare the utility (acceptance probability) can be achieved under multi-path solution and single-path solution. As an example, we presents the bandwidth allocation results computed at peak hour 5pm on Wednesday for both networks. Figure 4.2 plots the utilities of each IE flow, and the IE flows are sorted by their utilities in the figures. As shown in the figure, multi-path algorithm achieves strictly higher utility for all IE pairs in both networks. In particular, comparing to single-path algorithm, our multi-path algorithm maximizes the minimum utility from 74.74% to 92.90% in Abilene, and from 45.15% to 56.54% in GEANT. Furthermore, the number of IE pairs with 100% utility also increases from 40 to 46 in Abilene and 223 to 290 in GEANT.

We then show the impact of multi-path formulation on circuit configuration in practice. Figure 4.3 plots the drop rates of single-path and multi-path solutions. Again, we present the drop rates across 5-day simulated traffic. Notice the multi-path results shown in Figure 4.3 are higher than the drop rate of COPLAR-NR in Figure 4.1 because Figure 4.3 has twice as much traffic as in Figure 4.1. As shown in the figure, multi-path solution significantly reduces drop rates at all time instances in both networks, especially in the Abilene network where single-path routing suffers higher drop rates because of the existence of bottleneck links. For example, at 4AM Wednesday in Abilene, the multi-path circuit configuration reduces the drop rate of the single-path solution by 91.30% from 20.12% to 1.75%. Across all the simulated traffic matrices, the multi-path solution reduces the drop rate of the single-path solution by 46.69% to 100% in Abilene and 15.88% to 98.51% in GEANT. Therefore, the results in Figure 4.2 and Figure 4.3 strongly indicate the importance and advantage of multi-path routing to circuit provisioning.

4.3 Adaptive Rerouting Analysis

In this section, we investigate the properties of our adaptive rerouting algorithms including the converge time of the algorithms, the impact of loop-free routing and the effectiveness of cost functions.



(a) Abilene. (b) GEANT.

Figure 4.4: Converge time comparison.

4.3.1 Converge Time Analysis

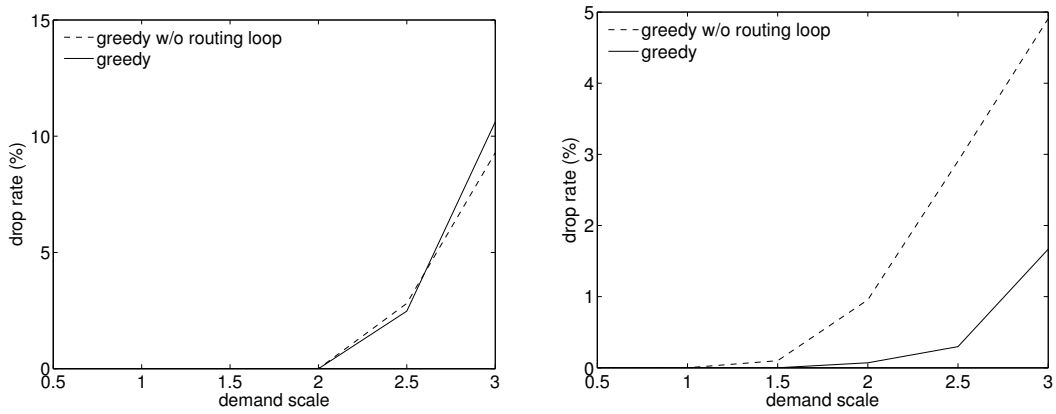
First of all, we evaluate the convergence property of adaptive rerouting algorithms. In particular, we are interested in whether an adaptive rerouting algorithm can reach a steady state and how long does it take to converge. If an adaptive rerouting algorithm cannot converge, the network could be unstable and cause unnecessary packet drops. On the other hand, if the converge time is too long, an adaptive rerouting algorithm could never reach its optimal performance because traffic is dynamically changing. Here, we use the peak hour traffic at time interval 4pm scaled by a factor of 2 as an example to demonstrate the converge property. Figure 4.4 plots the change of network drop rate over 2 minutes after the network traffic was initiated in the network. The network drop rate is the complement of network throughput (i.e. the percentage of offering traffic routed through a network). In the figure, we compare the results of a greedy load balanced algorithm without routing loop and the adaptive rerouting algorithm based on game theory described in Section 4.1.2.

At the beginning of a simulation, all traffic use its direct circuit, then the adaptive rerouting algorithms start to adjust the split ratio among other rerouting paths when excess traffic occur on the direct path. Thus, as expected, the network drop rate of GEANT network decreases over time as shown in Figure 4.4 (b).

Furthermore, both adaptive rerouting algorithms reached a steady state within 10 seconds. Especially, the greedy algorithm was able to keep using the same routing after achieving 0% drop rate. However, in the results of Abilene network shown in Figure 4.4 (a), the network had higher utilization and both adaptive rerouting algorithm cannot achieve 0% drop rate. Even though the greedy algorithm was still able to reduce drop rate within 10 seconds, it cannot reach steady state and its drop rate was oscillated between 6% to 8%. On the other hand, the adaptive rerouting algorithm based on game theory had a much more stable converge state with less than 0.5% drop rate variation after 10 seconds. Therefore, while both adaptive rerouting algorithms could effectively reduce network drop rate, the algorithm based on game theory could guarantee a network quickly converges to a steady state within less than 10 seconds.

4.3.2 Routing Loop Comparison

Next, we analysis the impact of routing loop for adaptive rerouting. Since our adaptive rerouting algorithm based on game theory must have loop-free paths, we evaluate routing loop using the greedy rerouting algorithm. Specifically, we compare the greedy routing with routing loop where a node can forward traffic any node, and the greedy routing without routing loop where loop is eliminated by using our ring-based loop-free routing table. Figure 4.5 plots the drop rate when network traffic is scaled by a factor of 0.5 to 3. At each scale factor, we plot the average number of results from 24 time intervals evenly scatter across the fist day simulated traffic matrices. For the Abilene network shown in Figure 4.5 (a), we have similar results for both with and without using loop-free routing tables. However, under a lower utilized network like GEANT network, greedy algorithm with routing loop can achieve much lower drop rate as shown Figure 4.5 (a). Without loop-free routing table can achieve lower drop rate for two reasons. First, the loop-free routing table limits the path diversity to prevent routing loop. Thus, less routing path options are available to reroute traffic. Second, greedy algorithm could easily make wrong routing decision because it only has the next hop information, and routing loop allow the greedy algorithm to regret its routing decision.

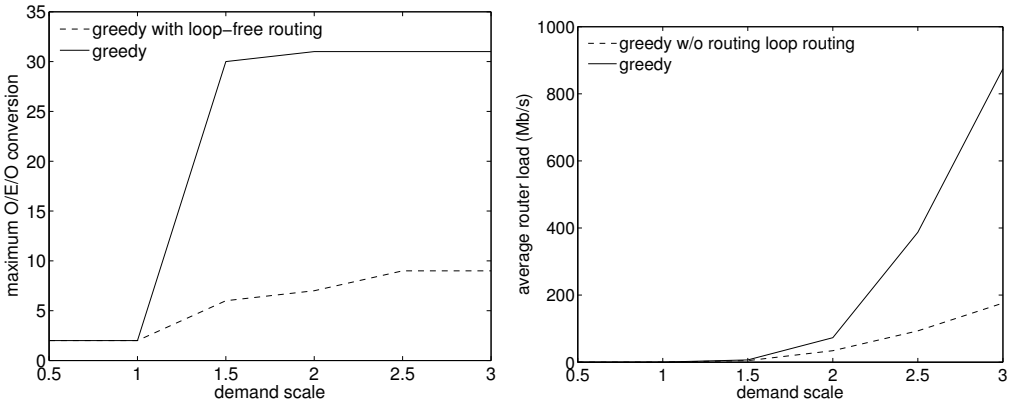


(a) Abilene. (b) GEANT.

Figure 4.5: Drop rate comparison between with and without routing loop.

For example, assume node s can use node A or B as intermediate node to reach destination node t. If there is lots of residual circuit capacity left from s to A but no residual capacity left from A to t, the greedy algorithm without routing loop will send more traffic through A and drop all packets at circuit from A to t. On the other hand, the greedy algorithm with routing loop could send traffic from A back to s, then pick node B instead to achieve lower drop rate.

Although allowing routing loop could achieve higher network throughput, it also cause serious routing problem for optical networks. Similar to Figure 4.5, we plot the maximum number of O/E/O conversion and router load for GEANT network in Figure 4.6. As shown in Figure 4.6 (a), the maximum number of O/E/O conversion with routing loop quickly reaches its maximum limit 32. In contrast, our loop-free routing successfully limits the maximum O/E/O conversion at 8. Furthermore, the greedy algorithm with routing loop causes much higher router load because it keeps rerouting packets until a packet reaches its destination or maximum forwarding limit. In particular, the average router increases at an exponential rate for the greedy algorithm with routing loop because as the network getting more congested, it becomes harder to find residual capacity. As a result, the greedy algorithm makes more wrong routing decision and causes more routing loop and redundant router load. Therefore, loop-free routing is necessary to prevent an



(a) Maximum number of O/E/O. (b) Average router load.

Figure 4.6: Network performance comparison between with and without routing loop in GEANT network.

explosion for router load and number of O/E/O conversion.

4.3.3 Cost Function Comparison

Finally, we compare the network performance of using different cost functions for the game theory based adaptive rerouting algorithm. Specifically, we consider three cost functions. The first cost function is the link utilization, u. The second cost function is the queuing delay, q, which is actually a function of link utilization according to the M/D/1 queuing model [67] where $q = \frac{1}{2*(1-u)}+1$. Thus, the queuing delay is tail-up as the link utilization growing to 100%. The last cost function is our escalated function shown in Table 4.2 where $u_i = 1-[(1-u_2)*0.5^{i-2}]$ and rerouting starts after link utilization is larger than 80%.

Figure 4.7, 4.8 and 4.9 plots the measurements under varied traffic scale factor for drop rate, maximum O/E/O conversion and router load, respectively. As shown in Figure 4.7, using link utilization and queuing delay as cost function did not result in lower drop rate than our proposed cost function. In fact, their drop rates were even higher than the greedy algorithm without routing loop. On the other hand, our cost function achieve a low drop rate that almost close to the greedy algorithm with routing loop. In addition, as shown in Figure 4.8 and 4.9, the maximum number of O/E/O conversion and router load grow slowly using our

escalated step cost function. From Figure 4.8 and 4.9, we also found the maximum number of O/E/O conversion and router load for the other two cost functions have higher values at smaller scale factor and slightly decrease when traffic scale factor is larger. This is the caused by the problem we mentioned in Figure 3.5 where a bad cost function could not appropriately decide when to reroute traffic. Even though the cost function of queuing delay does have tail-up when link utilization grows, clearly it starts to reroute when network utilization is still low while not reroute enough traffic when network utilization is high. Therefore, overall, our proposed cost function described in Section 3.6 has the lowest drop rate among all. In particular, our cost function achieves a drop rate close to the greedy algorithm with routing loop, while we prevent the network from suffering the high router load and number of O/E/O conversion.

4.4 Overall Performance Analysis

We extensively evaluate the performance of our COPLAR routing paradigm by comparing with the two well-known packet switching routing, OSPF and ECMP.

4.4.1 Time Series Performance Evaluation

First, we plot the time series network performance at each of simulated time intervals across the first day (288 intervals for Abilene, 96 intervals for GEANT) in Figure 4.10 and Figure 4.11. Again, to show the results at a higher utilized network, the traffic was scaled at a factor of 2. As shown in Figure 4.10, both networks suffer higher drop rate at peak hour (4pm for Abilene and 8AM for GEANT). In Abilene network, ECMP achieved lower drop rate because traffic was load balanced across multiple paths and network congestion could be less. In GEATN network, because only paths with the exact same length were considered by ECMP, we didn't observe multi-path routing occur often. Thus, the results of OSPF are similar to ECMP. Nevertheless, our COPLAR architecture with adaptive rerouting consistently achieved lowest drop rate in most of time intervals. However, as shown in Figure 4.11, COPLAR with adaptive rerouting achieved better network

throughput with much lower router load. Specifically, in Abilene network, the highest average router load for OSPF, ECMP and COPLAR is 6.56 Gb/s, 5.82 Gb/s and 0.29 Gb/s. In GEANT network, the highest average router load for OSPF, ECMP and COPLAR is 801 Mb/s, 801 Mb/s and 57 Mb/s. In other words, COPLAR reduces router load by 92.1% to 97.2 among all time intervals in GEANT network, and by 94.32% to 99.8% in Abilene network.

4.4.2 Scaled Traffic Performance Evaluation

Then, we plot the performance comparison under varied traffic scale factors in Figure 4.12-4.13, and we summarize results in Table 4.3. Figure 4.12 shows that COPLAR has significant lower drop rate than OSPF and ECMP at all scale factors. OSPF starts dropping packet after scale factor 1 for both networks because we already normalized the traffic so that at least one link was fully utilized under OSPF. Furthermore, we observed the drop rate of OSPF and ECMP grow faster in Abilene because more links congested at higher scale factors. In contrast, the number of congested links still remains the same in GEANT as scale factor increases because we have much higher link utilization at a few links due to their smaller capacity. As a result, drop rate is linear increased for OSPF and ECMP which only use fix routing paths.

Figure 4.14 and 4.15 show that COPLAR can provide much higher network throughput with significantly less O/E/O conversion. The number of O/E/O conversion for OSPF and ECMP remains constant regardless scale factor because their routing are static. The number of O/E/O conversion of OSPF and ECMP are both solely dependent on the network topology, and it is around 2.5 in average and 6 in maximum. In contrast, benefit from establishing direct circuit among IE-pairs, the average O/E/O conversion of COPLAR is almost remains at 1 and only slightly grows as traffic scale increases. Even though the maximum O/E/O conversion could increase more, COPLAR still has less worst case O/E/O conversion than OSPF and ECMP in Abilene and one additional O/E/O conversion in GEANT.

Finally, directly caused by the O/E/O conversion, Figure 4.13 shows COPLAR also has significant less router load than OSPF and ECMP. This is due to the fact

that majority of the traffic can be carried by its direct circuit without the need of intermediate routers. Furthermore, we found the router load increased rapidly for OSPF and ECMP, while COPLAR almost maintain constant load. Thus COPLAR is able to handle much more offering traffic without overwhelming electronic routers.

Concludes all, our results show our COPLAR routing paradigm with proposed adaptive rerouting algorithm successfully improves network throughput while significantly reduces router load and O/E/O conversion for optical networks. Therefore, the COPLAR architecture could be suitable on optical networks for both performance and feasibility purposes.

4.5 Conclusion

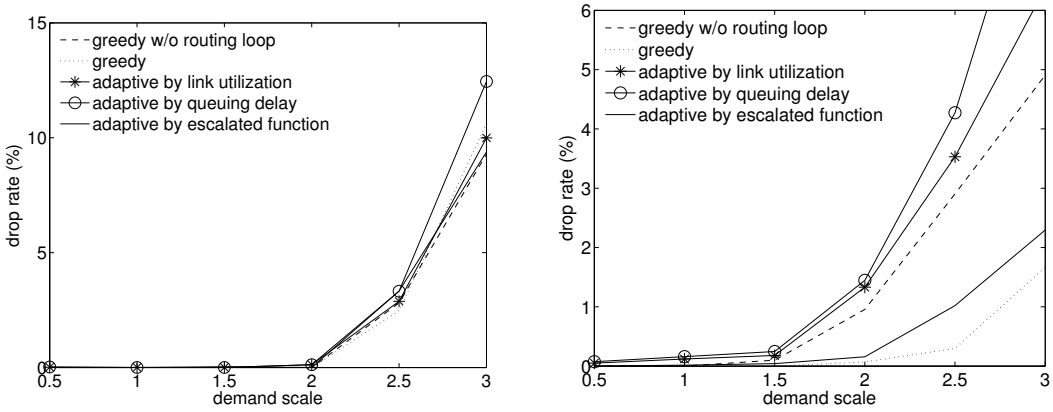
In this chapter we extensively evaluate our COPLAR approach with realistic simulator and network datasets. Our evaluation results show a number of interesting results. First, the majority of the traffic was able to be carried by the circuits computed from our circuit provisioning algorithm even during peak traffic hours. Throughout a full week of traffic simulation, the maximum drop rate is merely 7% with 0% drop for the most of time intervals. The results also show that we can achieve such low drop rate because of the ability of considering multi-path routing in our bandwidth allocation algorithm for circuit provisioning.

Second, the results show our adaptive rerouting approach can quickly converge a steady state to prevent network becomes oscillated and unstable. Furthermore, our loop-free routing table for rerouting significantly reduces redundant overhead for electronic routers and O/E/O conversion for traffic by eliminating routing loop. Finally, we demonstrate our cost function design successfully achieves our network performance objectives, so that the network throughput is maximized and router load is minimized.

Last but not least, we demonstrate with the help of adaptive rerouting, COPLAR was able to achieve much lower drop rate. In addition, when the actual traffic were scaled up, COPLAR was surprisingly able to handle a higher traffic load than conventional packet routing [66, 8]. The effectiveness of COPLAR can be

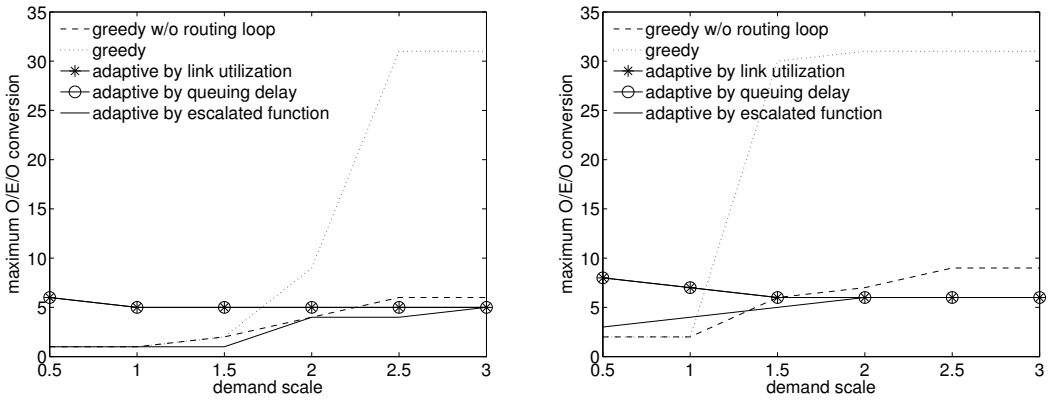
attributed to the path diversity available for the adaptive rerouting of traffic as well as the ability of rerouting algorithm to adjust routing on-the-fly. Therefore, COPLAR could achieve much higher network throughput by utilizing residual circuit capacity in a more effective manner.

Summarize all, COPLAR achieved much higher network throughput with significantly less overhead on electronic routers and O/E/O conversions for traffic. Therefore, it is possible to use COPLAR for optical networks to provide even better network performance while alleviating the bottleneck of electronic routers.



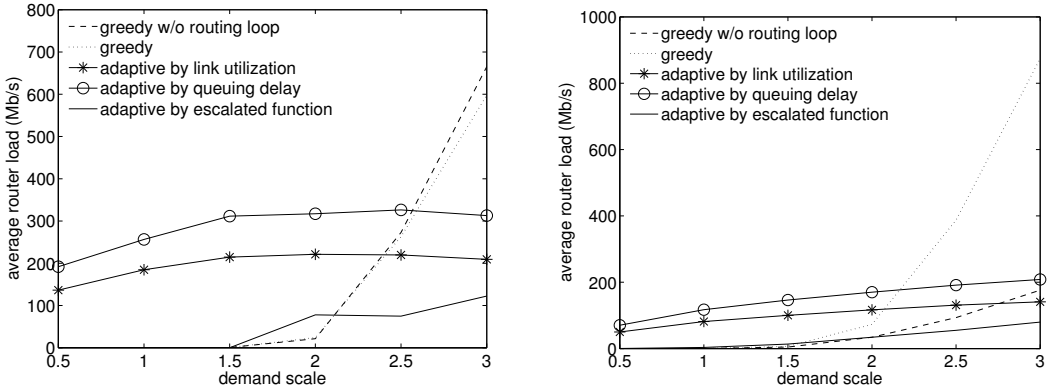
(a) Abilene. (b) GEANT.

Figure 4.7: Drop rate comparison using different cost functions.



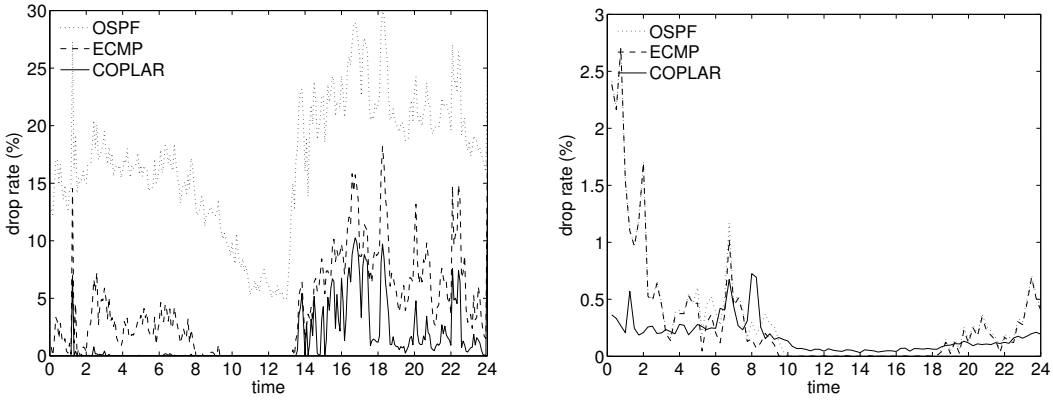
(a) Abilene. (b) GEANT.

Figure 4.8: Maximum O/E/O comparison using different cost functions.



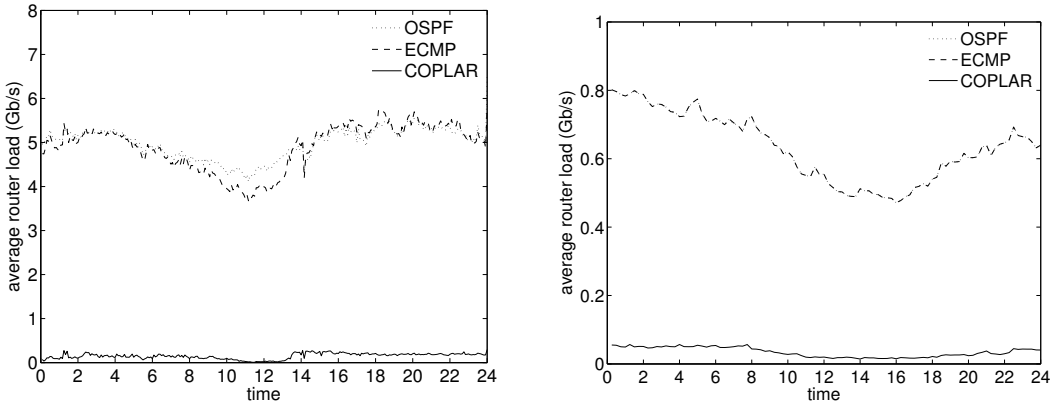
(a) Abilene. (b) GEANT.

Figure 4.9: Average router load comparison using different cost functions.



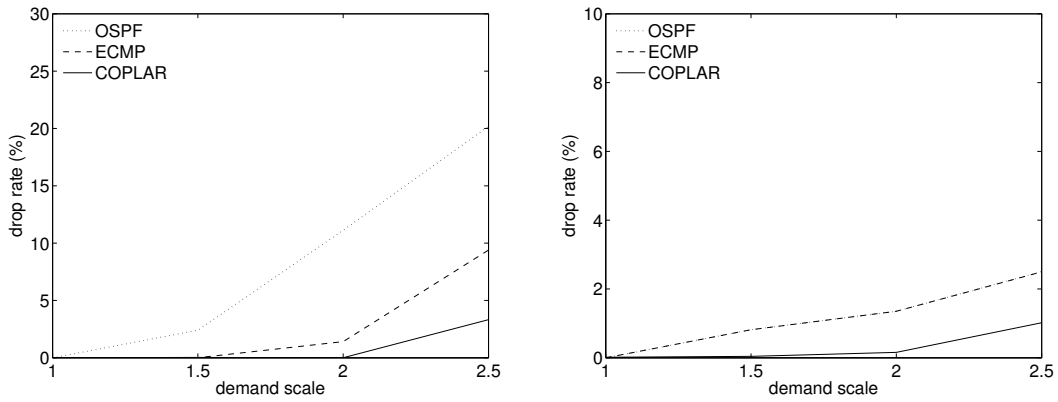
(a) Abilene. (b) GEANT.

Figure 4.10: Drop rate comparison over 24 hours when traffic is scaled by a factor of 2.



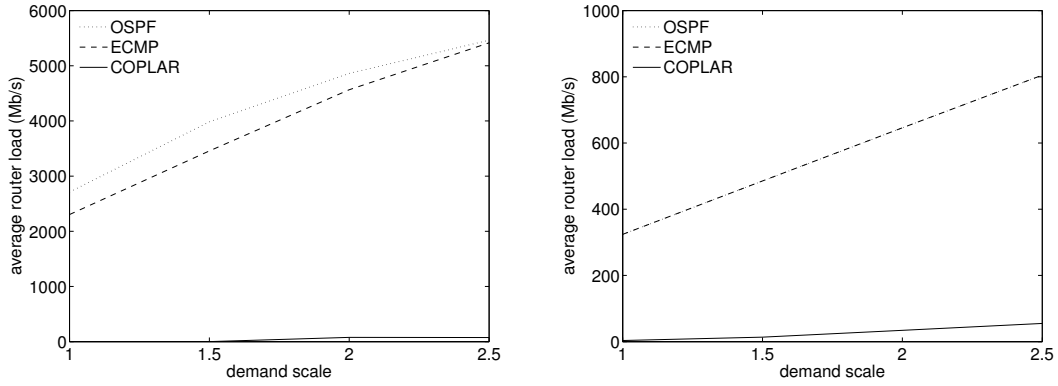
(a) Abilene. (b) GEANT.

Figure 4.11: Average router load comparison over 24 hours when traffic is scaled by a factor of 2.



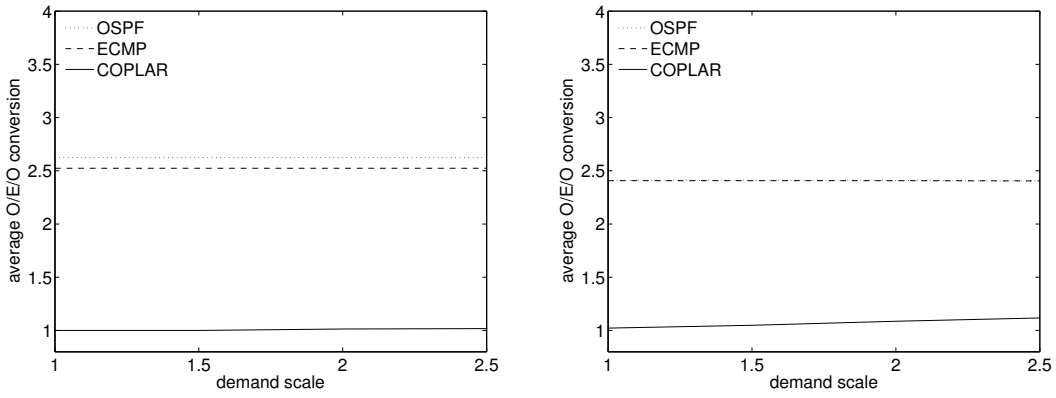
(a) Abilene. (b) GEANT.

Figure 4.12: Drop rate comparison with traffic scale varied from 1 to 3.



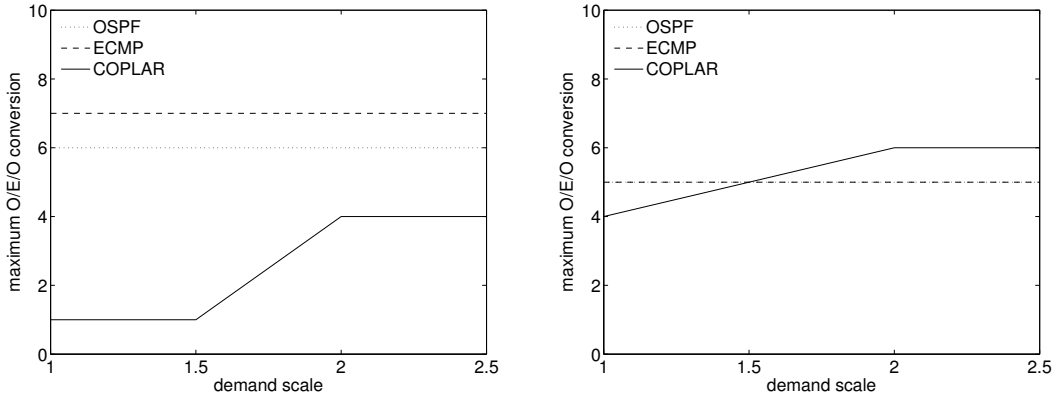
(a) Abilene. (b) GEANT.

Figure 4.13: Average router load comparison with traffic scale varied from 1 to 3.



(a) Abilene. (b) GEANT.

Figure 4.14: Average number of O/E/O conversion comparison with traffic scale varied from 1 to 3.



(a) Abilene. (b) GEANT.

Figure 4.15: Maximum number of O/E/O conversion comparison with traffic scale varied from 1 to 3.

Table 4.3: COPLAR performance comparison with OSPF and ECMP under varied traffic scale factor.

Matric	Network	Routing	Traffic Scale Factor				
			1.0	1.5	2.0	2.5	3
drop rate (%)	Abilene	OSPF	0.00	2.40	11.12	20.19	27.79
		ECMP	0.00	0.00	1.41	9.39	18.53
		COPLAR	0.00	0.00	0.03	3.33	9.38
	GEANT	OSPF	0.00	0.82	1.35	2.51	5.53
		ECMP	0.00	0.82	1.35	2.50	4.83
		COPLAR	0.02	0.04	0.16	1.02	2.30
router load (Mb/s)	Abilene	OSPF	2711	3983	4860	5465	6020
		ECMP	2304	3456	4565	5416	6079
		COPLAR	0	0	75	78	122
	GEANT	OSPF	324	485	646	805	952
		ECMP	324	485	646	805	952
		COPLAR	4	14	34	55	80
average O/E/O	Abilene	OSPF	2.625	2.625	2.625	2.625	2.625
		ECMP	2.523	2.523	2.523	2.523	2.523
		COPLAR	1.000	1.000	1.014	1.018	1.024
	GEANT	OSPF	2.410	2.410	2.410	2.410	2.410
		ECMP	2.405	2.405	2.405	2.405	2.405
		COPLAR	1.022	1.049	1.087	1.117	1.154
maximum O/E/O	Abilene	OSPF	6	6	6	6	6
		ECMP	6	6	6	6	6
		COPLAR	1	1	4	4	5
	GEANT	OSPF	5	5	5	5	5
		ECMP	5	5	5	5	5
		COPLAR	4	5	6	6	6

Chapter 5

Network Security Application - Proactive Surge Protection

5.1 Introduction

A coordinated attack can potentially disable a network by flooding it with traffic. Such attacks are also known as bandwidth-based distributed denial-of-service (DDoS) attacks and are the focus of our work. Depending on the operator, the provider network may be a small-to-medium regional network or a large core network. For small-to-medium size regional networks, this type of bandwidth-based attacks has certainly disrupted service in the past. For core networks with huge capacities, one might argue that such an attack risk is remote. However, as reported in the media [9], large botnets already exist in the Internet today. These large botnets combined with the prevalence of high speed Internet access can quite easily give attackers multiple tens of Gb/s of attack capacity. Moreover, core networks are engineered to support normal traffic loads reliably and not to support maximum traffic load from all subscribers. For example, in the Abilene network [2], some of the core routers have an incoming capacity of larger than 30 Gb/s from the access networks, but only 20 Gb/s of outgoing capacity to the core. Although commercial ISPs do not publish their oversubscription levels, they are generally substantially higher than the ones found in the Abilene network due to

commercial pressures of maximizing return on investments.

Considering these insights, one might wonder why we have not seen multiple successful bandwidth-based attacks to large core networks in the past. The answer to this question is difficult to assess. Partially, attacks might not be occurring because the organizations which control the botnets are interested in making money by distributing SPAM, committing click frauds, or extorting money from mid-sized websites. Therefore, they would have no commercial interest in disrupting the Internet as a whole. Another reason might be that network operators are closely monitoring network utilization and actively balancing traffic flow and blocking DDoS attacks. Nonetheless, recent history has shown that if such an attack possibility exists, it will eventually be exploited. For example, SYN flooding attacks were described in [4] years before such attacks were used to disrupt servers in the Internet.

To defend against large bandwidth-based DDoS attacks, a number of defense mechanisms currently exist, but many are reactive in nature (i.e., they can only respond after an attack has been identified in an effort to limit the damage). However, the onset of large-scale bandwidth-based attacks can occur almost instantaneously, causing potentially a huge *surge* in traffic that can effectively knock out substantial parts of a network before reactive defense mechanisms have a chance to respond. To provide a broad first line of defense against DDoS attacks when they happen, we propose a new protection mechanism called Proactive Surge Protection (PSP). In particular, under a flooding attack, traffic loads along attack routes will exceed link capacities, causing packets to be dropped indiscriminately. Without proactive protection, even for traffic flows that are not under direct attack, substantial packet loss will occur if these flows pass through links that are common to attack routes, resulting in significant *collateral damage*. The PSP solution aims to provide *bandwidth isolation* between flows so that the collateral damage to traffic flows not under direct attack is substantially reduced.

This bandwidth isolation is achieved through a combination of traffic data collection, bandwidth allocation of network capacity based on traffic measurements, metering and tagging of packets at the network perimeter into two differ-

entiated priority classes based on capacity allocation, and preferential dropping of packets in the network when link capacities are exceeded. It is important to note that PSP has no impact on the regular operation of the network if no link is overloaded. It therefore introduces no penalty in the common case. In addition, PSP is deployable using existing router mechanisms that are already available in modern routers, which makes our approach scalable, feasible, and cost effective. Further, PSP is resilient to IP spoofing as well as changes in the underlying traffic characteristics such as the number of TCP connections. This is due to the fact that we focus on protecting traffic between different ingress-egress interface pairs in a provider network and both the ingress and egress interface of an IP datagram can be directly determined by the network operator. Therefore, the network operator does not have to rely on unauthenticated information such as a source or destination IP address to tag a packet.

Specifically, we propose three policies, Mean-PSP, CDF-PSP and GCDF-PSP. Mean-PSP is solely based on the average traffic demand while CDF-PSP takes into consideration of the traffic variability observed in historical traffic measurements. CDF-PSP aims to maximize the acceptance probability (or equivalently the min-max minimization of the drop probability) of packets by using the cumulative distribution function over historical data sets as the objective function, and it can be solved as an utility max-min fair bandwidth allocation problem. Finally, GCDF-PSP is a variant of the CDF-PSP policy in which the traffic variability is modeled as a Gaussian distribution, and the problem is simplified to a weighted max-min bandwidth allocation problem. Furthermore, GCDF-PSP allows network operators to model future traffic variability scenarios in which historical datasets are not applicable.

To test the robustness of our proposed approach, we evaluated the PSP mechanism using both *highly distributed* attack scenarios involving a high percentage of ingress and egress routers, as well as *targeted* attack scenarios in which the attacks are concentrated to a small number of egress destinations. Our extensive evaluation across two large commercial backbone networks shows that up to 95.5% of the network could suffer collateral damage, but our solution was able to

significantly reduce the amount of collateral damage by up to 97.58% in terms of the number of packets dropped and up to 90.36% in terms of the number of flows with packet loss. In addition, we show that PSP can maintain low packet loss rates even when the intensity of attacks is increased significantly. Beyond evaluating extensively the impact of our protection scheme on packet drops, we also present detailed analysis on the impact of our scheme at the level of flow aggregates between individual ingress-egress interface pairs in the network.

The rest of this chapter is organized as follows. Section 5.2 outlines related work. Section 5.3 presents a high-level overview of our proposed PSP approach. Section 5.4 describes in greater details the central component of our proposed architecture that deals with bandwidth allocation policies. Section 5.5 describes our experimental setup, and Section 5.6 presents extensive evaluation of our proposed solutions across two large backbone networks. Finally, we discuss the limitations of the approach in Section 5.7 and conclude the chapter in Section 5.8.

5.2 Related Work

DDoS protection has received considerable attention in the literature. The oldest approach, still heavily in use today, is typically based on coarse-grain traffic anomalies detection [59, 3]. Traceback techniques [93, 80, 84] are then used to identify the true attack source, which could be disguised by IP spoofing. After detecting the true source of the DDoS traffic the network operator can block the DDoS traffic on its ingress interfaces by configuring access control lists (ACL) or by using DDoS scrubbing devices such as [6]. However, rarely are the true sources of DDoS traffic identified, which limits use of ACLs as a last resource means to block attack traffic since legitimate traffic will likely also be blocked. DDoS scrubbers tend to attract more traffic into the core network toward scrubbing facilities, which is counter to our objective to protect the core network. Furthermore, although these approaches are practical, they do not allow for an instantaneous protection of the network. As implemented today, theses approaches require multiple minutes to detect and mitigate DDoS attacks, which does not match the time sensitivity of

today's applications. Similarly, network management mechanisms that generally aim to find alternate routes around congested links also do not match the time sensitivity of today's applications.

Work has also focused on enhancing the current Internet protocol and routing implementations. For example, multiple proposals have suggested to limit the best effort connectivity of the network using techniques such as capabilities models [72, 94, 69] , filtering schemes [57, 14] or routing modification [16, 38]. The main focus of these papers is the protection of customers connecting to the core network rather than protecting the core itself, which is the focus of our work. To illustrate the difference, consider a scenario in which an attacker controls a large number of zombies. These zombies could communicate with each other, granting each other capabilities or similar rights to communicate. If planned properly, this traffic is still sufficient to attack a core network. The root of the problem is that the core cannot rely on either the sender or the receiver of the traffic to protect itself.

Recently, a couple of novel defense mechanisms deployed in a core network are also proposed to mitigate suspicious attack traffic. One of them is prime [89] and the other is pushback [45]. Similar to the proposals limiting connectivity cited above, prime focuses on protecting individual customers. This leads again to an issue of reliance in that a service provider should not rely on its customers for protection. Furthermore, their solution relies heavily on the operator and customers knowing *a priori* who are the good and bad network entities, and their solution has a scalability issue in that it is not scalable to maintain detailed per-customer state for all customers within the network. On the other hand, pushback is a reactive defense mechanism which detects suspicious traffic at congested routes then sends filtering messages to upstream routers. Not only does pushback require communication and cooperation between routers, it simply needs time to react and propagate the filtering messages.

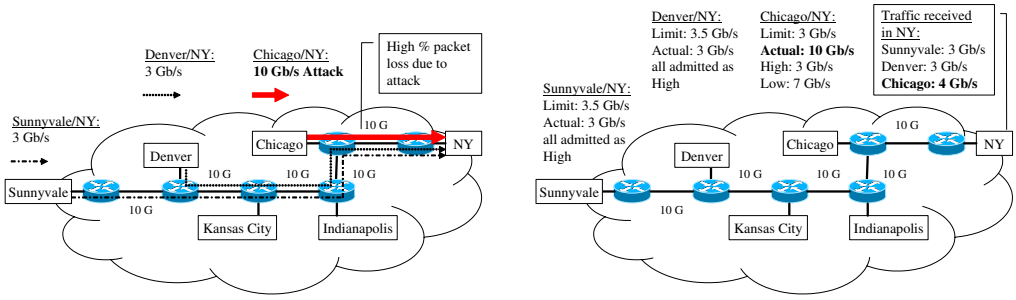
Our work builds on the existing body of literature on max-min fair resource allocation [22, 88, 42, 24, 75, 79, 71] to the problem of proactive DDoS defense. However, our work here is different in that we use max-min fair allocation for the

purpose of differential tagging of packets with the objective of minimizing collateral damage when a DDoS attack occurs. Our work here is also different than the server-centric DDoS defense mechanism proposed in [96], which is aimed at protecting end-hosts rather than the network. In their solution, a server explicitly negotiates with selected upstream routers to throttle traffic destined to it. Max-min fairness is applied to set the throttling rates of these selected upstream routers. Like [89] discussed above, their solution also has a scalability issue in that the selected upstream routers must maintain per-customer state for the requested rate limits.

Finally, our work also builds on existing preferential dropping mechanisms that have been developed for providing Quality-of-Service (QoS) [29, 32]. However, for providing QoS, the service-level-agreements that dictate the bandwidth allocation are assumed to be either specified by customers or decided by the operator for the purpose of traffic engineering. There is also a body of work on measurement-based admission control for determining whether or not to admit new traffic into the network, e.g. [39, 47]. With both service-level-agreement-based and admission-control-based bandwidth reservation schemes, rate limits are enforced. Our work here is different in that we use preferential dropping for a different purpose to provide bandwidth isolation between traffic flows to minimize the damage that attack traffic can cause to regular traffic.

5.3 Proactive Surge Protection

In this section, we present a high-level architectural overview of a DDoS defense solution called Proactive Surge Protection (PSP). To illustrate the basic concept, we will depict an example scenario for the Abilene network. That network consists of 11 core routers that are interconnected by OC192 (10 Gb/s) links. For the purpose of depiction, we will zoom in on a portion of the Abilene network, as shown in Figure 5.1(a). Consider a simple illustrative situation in which there is a sudden bandwidth-based attack along the origin-destination (OD) pair Chicago/NY, where an OD pair is defined to be the corresponding pair of ingress and egress nodes. Suppose that the magnitude of the attack traffic is 10 Gb/s.



(a) Attack along Chicago/NY

(b) Shielded Sunnyvale/NY and Denver/NY traffic from collateral damage

Figure 5.1: Attack scenario on the Abilene network.

This attack traffic, when combined with the regular traffic for the OD pairs Sunnyvale/NY and Denver/NY ($3 + 3 + 10 = 16$ Gb/s), will significantly oversubscribe the 10 Gb/s Chicago/NY link, resulting in a high percentage of indiscriminate packet drops. Although the OD pairs Sunnyvale/NY and Denver/NY are not under *direct* attack, these flows will also suffer substantial packet loss on links which they share with the attack OD pair, resulting in significant *collateral damage*. The flows between Sunnyvale/NY and Denver/NY are said to be caught in the *crossfire* of the Chicago/NY attack.

5.3.1 PSP Approach

The PSP approach is based on providing *bandwidth isolation* between different traffic flows so that the amount of collateral damage sustained along crossfire traffic flows is minimized. This bandwidth isolation is achieved by using a form of *soft* admission control at the perimeter of a provider network. In particular, to avoid saturation of network links, we impose *rate limits* on the amount of traffic that gets injected into the network for each OD pair. However, rather than imposing a *hard* rate limit, where packets are *blocked* from entering the network, we classify packets into two priority classes, *high* and *low*. Metering is performed at the perimeter of the network, and packets are tagged *high* if the arrival rate is below a certain threshold. But when a certain threshold is exceeded, packets will get tagged as *low* priority. Then, when a network link gets saturated, e.g. when

an attack occurs, packets tagged with a low priority will be dropped preferentially. This ensures that our solution does not drop traffic unless a network link capacity has indeed been exceeded. Under normal network conditions, in the absence of sustained congestion, packets will get forwarded in the same manner as without our solution.

Consider again the above example, now depicted in Figure 5.1(b). Suppose we set the high priority rate limit for the OD pairs Sunnyvale/NY, Denver/NY, and Chicago/NY to 3.5 Gb/s, 3.5 Gb/s, and 3 Gb/s, respectively.This will ensure that the total traffic admitted as high priority on the Chicago/NY link is limited to 10 Gb/s. Operators can also set maximum rate limits to some factor below the link capacity to provide the desired headroom (e.g. set the target link load to be 90%). If the limit set for a particular OD pair is *above* the *actual* amount of traffic along that flow, then all packets for that flow will get tagged as high priority. Consider the OD pair Chicago/NY. Suppose the actual traffic under an attack is 10 Gb/s, which is above the 3 Gb/s limit. Then, only 3 Gb/s of traffic will get tagged as high priority, and 7 Gb/s will get tagged as low priority. Since the total demand on the Chicago link exceeds the 10 Gb/s link capacity, considerable packets would get dropped. However, the packets drop will come from the OD pair Chicago/NY since all packets from Sunnyvale/NY and Denver/NY would have been tagged as high priority. Therefore, the packets for the OD pairs Sunnyvale/NY and Denver/NY would be shielded from collateral damage.

Although our simple illustrative example shown in Figure 5.1 only involved one attack flow from one ingress point, the attack traffic in general can be highly distributed. As we shall see in Section 5.6, the proposed PSP method is also quite effective in such distributed attack scenarios.

5.3.2 PSP Architecture

Our proposed PSP architecture is depicted in Figure 5.2. The architecture is divided into a policy plane and an enforcement plane. The traffic data collection and bandwidth allocation components are on the policy plane, and the differential tagging and preferential drop components are on the enforcement plane.

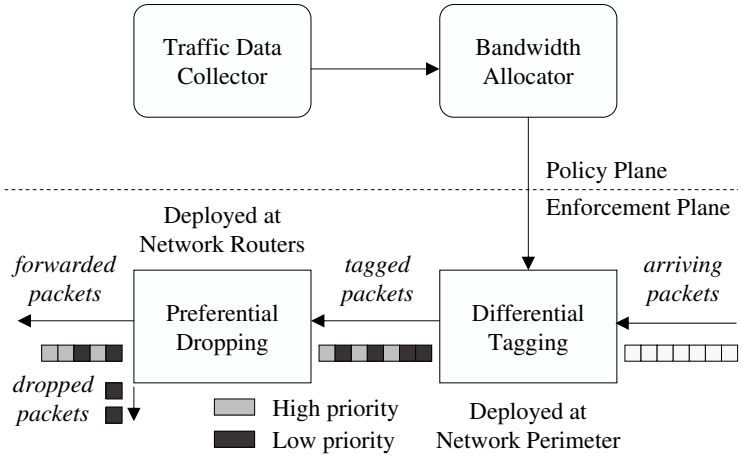


Figure 5.2: Proactive Surge Protection (PSP) architecture.

Traffic Data Collector: The role of the traffic data collection component is to collect and summarize historical traffic measurements. For example, the widely deployed Cisco sampled NetFlow mechanism can be used in conjunction with measurement methodologies such that those outlined in [34] to collect and derive traffic matrices for different times throughout a day, a week, a month, etc, between different origin-destination (OD) pairs of ingress-egress nodes. The infrastructure for this traffic data collection already exists in most service provider networks. The derived traffic matrices are used to estimate the range of expected traffic demands for different time periods.

Bandwidth Allocator: Given the historical traffic data collected, the role of the bandwidth allocator is to determine the *rate limits* at different time periods. For each time period t, the bandwidth allocator will determine a *bandwidth allocation matrix*, $B(t) = [\ b_{s,d}(t)\ ]$, where $b_{s,d}(t)$ is the rate limit for the corresponding OD pair with ingress node s and egress node d for a particular time of day t. For example, a different bandwidth allocation matrix $B(t)$ may be computed for each hour in a day using the historical traffic data collected for same hour of the day. Under normal operating conditions, network links are typically under-utilized. Therefore, traffic demands from historical measurements will reflect this under-utilization. Since there is likely to be *room* for admitting more traffic into

the high priority class than observed in the historical measurements, we can fully allocate in some fair manner the available network resources to high priority traffic. By fully allocating the available network resources beyond the previously observed traffic, we can provide *headroom* to account for estimation inaccuracies and traffic burstiness. The bandwidth allocation matrices can be computed offline, and operators can remotely configure routers at the network perimeter with these matrices using existing router configuration mechanisms.

Differentiated Tagging: Given the rate limits determined by the bandwidth allocator, the role of the differential tagging component is to perform the metering and tagging of packets in accordance to the determined rate limits. This component is implemented at the perimeter of the network. In particular, packets arriving at ingress node s and destined to egress node d are tagged as high priority if their metered rates are below the threshold given by $b_{s,d}(t)$, using the bandwidth allocation matrix $B(t)$ for the corresponding time of day. Otherwise, they are tagged as low priority. These traffic management mechanisms for metering and tagging are commonly available in modern routers at linespeeds.

Preferential Drops: With packets tagged at the perimeter, low priority packets can be dropped preferentially over high priority packets at a network router whenever a sustained congestion occurs. Again, this preferential dropping mechanism [29] is commonly available in modern routers at linespeeds [5]. By using preferential drop at interior routers rather than simply blocking packets at the perimeter when a rate limit has been reached, our solution ensures that no packet gets dropped unless a network link capacity has indeed been exceeded. Under normal network conditions, in the absence of sustained congestion, packets will get forwarded in the same manner as without our surge protection scheme.

5.4 Bandwidth Allocation Polices

Intuitively, PSP works by fully allocating the available network resources into the high priority class in some fair manner so that the high priority class rate limits for the different OD pairs are *at least* as high as the *expected* normal traffic.

This way, should a DDoS attack occur that would saturate links along the attack route, *normal* traffic corresponding to *crossfire* OD pairs would be *isolated* from the attack traffic, thus minimizing collateral damage. In particular, packets for a particular crossfire OD pair would only be dropped at a congested network link if the *actual* normal traffic for that flow is *above* the bandwidth allocation threshold given to it. Therefore, bandwidth allocation plays a central role in affecting the *drop probability* of normal crossfire traffic during an attack. As such, the goal of bandwidth allocation is to allocate the available network resources with the objective of minimizing the drop probabilities for all OD pairs in some fair manner.

5.4.1 Formulation

To achieve the objectives of minimizing drop probability and ensuring fair allocation of network resources, we formulate the bandwidth allocation problem as a utility max-min fair allocation problem [22, 24, 79, 71]. The utility max-min fair allocation problem can be stated as follows. Let $\vec{x} = (x_1, x_2, \ldots, x_N)$ be the allocation to N flows, and let $(\beta_1(x_1), \beta_2(x_2), \ldots, \beta_N(x_N))$ be N utility functions, with each $\beta_i(x_i)$ corresponding to the utility function for flow i. An allocation $\vec{x}$ is said to be *utility max-min fair* if and only if increasing one component x_i must be at the expense of decreasing some other component x_j such that $\beta_j(x_j) \leq \beta_i(x_i)$.

Conventionally, the literature on max-min fair allocation uses the vector notation $\vec{x}(t) = (x_1(t), x_2(t), \ldots, x_N(t))$ to represent the allocation for some time period t. The correspondence to our bandwidth allocation matrix $B(t) = [\ b_{s,d}(t)\ ]$ is straightforward: $b_{s_i,d_i}(t) = x_i(t)$ is the bandwidth allocation at time t for flow i, with the corresponding OD pair of ingress and egress nodes (s_i, d_i). Unless otherwise clarified, we use the vector notation $\vec{x}(t) = (x_1(t), x_2(t), \ldots, x_N(t))$ and our bandwidth allocation matrix notation interchangeably.

The utility max-min fair allocation problem has been well-studied, and as shown in [24, 79], the problem can be solved by means of a "water-filling" algorithm. We briefly outline here how the algorithm works. The basic idea is to iteratively calculate the utility max-min fair share for each flow in the network. Initially, all flows are allocated rate $x_i = 0$ and are considered free, meaning that

its rate can be further increased. At each iteration, the water-filling algorithm aims to find largest increase in bandwidth allocation to free flows that will result in the maximum common utility with the available link capacities. The provided utility functions, $(\beta_1(x_1), \beta_2(x_2), \ldots, \beta_N(x_N))$, are used to determine this maximum common utility. When a link is saturated, it is removed from further consideration, and the corresponding flows that cross these saturated links are *fixed* from further increase in bandwidth allocation. The algorithm converges after at most L iterations, where L is the number of links in the network, since at least one link becomes saturated in each iteration. The reader is referred to [24, 79] for detailed discussions.

In the context of PSP, the utility max-min fair algorithm is used to implement different bandwidth allocation policies. In particular, we describe in this section three bandwidth allocation policies, Mean-PSP, CDF-PSP and GCDF-PSP. All of them are based on traffic data collected from historical traffic measurements. The first policy, Mean-PSP, simply uses the average historical traffic demands observed as *weights* in the corresponding utility functions. Mean-PSP is based on the simple intuition that flows with higher average traffic demands should receive proportionally higher bandwidth allocation. However, this policy does not directly consider the traffic variance observed in the traffic measurements.

To directly account for traffic variance, we propose a second policy, CDF-PSP, that explicitly aims to minimize drop probabilities by using the *Cumulative Distribution Functions* (CDFs) [22] derived from the empirical distribution of traffic demands observed in the traffic measurements. These CDFs can be used to capture the probability that the actual traffic will not exceed a particular bandwidth allocation. When these CDFs are used as utility functions, maximizing the utility corresponds directly to the minimization of drop probabilities.

Finally, GCDF-PSP is proposed when we consider the CDF of historical traffic can be approximated by a Gaussian distribution. Specially, we show the utility max-min allocation of CDF-PSP can be reduced to weight max-min in GCDF-PSP by selecting the weight of each flow to be the variance of traffic demand in Gaussian distribution. Each of these three policies is further illustrated next.

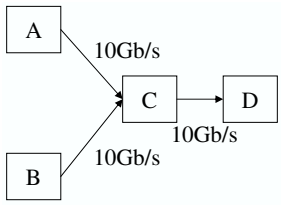
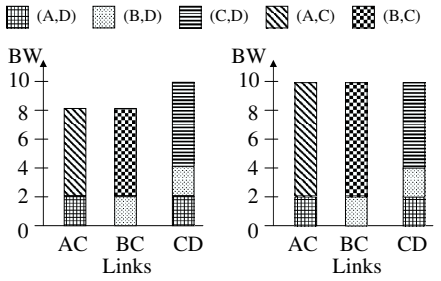
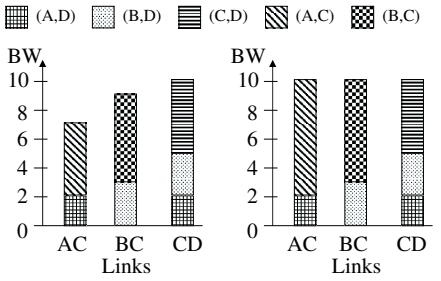


Figure 5.3: Network.



(a) 1st iteration. (b) 2nd iteration.

Figure 5.4: Mean-PSP water-filling illustrated.



(a) 1st iteration. (b) 2nd iteration.

Figure 5.5: CDF-PSP water-filling illustrated.

5.4.2 Mean-PSP: Mean-based Max-min Fairness

Our first allocation policy, Mean-PSP, simply uses the mean traffic demand as the utility function. In particular, the utility function for flow i is a simple linear function $\beta_i(x) = \frac{x}{\mu_i}$, where μ_i is the mean traffic demand of flow i, which simplifies to an easier weighted max-min fair allocation problem.

To illustrate how Mean-PSP works, consider the small example shown in Figure 5.3. It depicts a simple network topology with 4 nodes that are interconnected by 10 Gb/s links. Consider the corresponding traffic measurements shown in Table 5.1. For simplicity of illustration, each flow is described by just 5 data points, and the corresponding mean traffic demands are also indicated in Table 5.1. Consider the first iteration of the Mean-PSP water-filling procedure shown in Figure 5.4(a). The maximum common utility that can be achieved by all *free* flows is $\beta(x) = 1$, which corresponds to allocating 2 Gb/s each to the OD pairs (A, D) and (B, D) and 6 Gb/s each to the OD pairs (C, D), (A, C), and (B, C). For example, $\beta_{A,D}(x) = \frac{x}{\mu} = 1$ corresponds to allocating $x = 2$ Gb/s since μ for (A, D)

Table 5.1: Traffic demands and the corresponding bandwidth allocations for Mean-PSP and CDF-PSP.

Flows	Historical traffic measurements						BW allocation			
	Measured demands (sorted)					Mean	Mean-PSP		CDF-PSP	
							1st	2nd	1st	2nd
(A,D)	1	1	2	2	4	2	2	2	2	2
(B,D)	1	1	1	3	4	2	2	2	3	3
(C,D)	4	5	5	5	11	6	6	6	5	5
(A,C)	4	5	5	5	11	6	6	8	5	8
(B,C)	5	5	6	6	8	6	6	8	6	7

is 2. Since all three flows, (A, D), (B, D), and (C, D), share a common link CD, the sum of their first iteration allocation, $2 + 2 + 6 = 10$ Gb/s, would already saturate link CD. This saturated link is removed from consideration in subsequent iterations, and the flows (A, D), (B, D), and (C, D) are fixed at the allocation of 2 Gb/s, 2 Gb/s, and 6 Gb/s, respectively.

On the other hand, link AC is only shared by flows (A, C) and (A, D), which has an aggregate allocation of $2 + 6 = 8$ Gb/s on link AC after the first iteration. This leaves $10-8 = 2$ Gb/s of residual capacity for the next iteration. Similarly, link BC is shared by flows (B, C) and (B, D), which also has an aggregate allocation of $2 + 6 = 8$ Gb/s on link BC after the first iteration, with 2 Gb/s of residual capacity. After the first iteration, flows (A, C) and (B, C) remain free.

In the second iteration, as in shown Figure 5.4(b), the maximum common utility is achieved by allocating the remaining 2 Gb/s on link AC to flow (A, C) and the remaining 2 Gb/s on link BC to flow (B, C), resulting in each flow having 8 Gb/s allocated to it in total. The final Mean-PSP allocation is shown in Table 5.1.

5.4.3 CDF-PSP: CDF-based Max-min Fairness

Our second allocation policy, CDF-PSP, aims to explicitly capture the *traffic variance* observed in historical traffic measurements by using a Cumulative Distribution Function (CDF) model as the utility function. The use of CDFs [22] captures the *acceptance probability* of a particular bandwidth allocation as follows.

Let $X_i(t)$ be a random variable that represents the *actual* normal traffic for flow i at time t, and let $x_i(t)$ be the bandwidth allocation. Then the CDF of $X_i(t)$ is denoted as

$$Pr[X_i(t) \leq x_i(t)] = \Phi_{i,t}(x_i(t)),$$

and the drop probability is simply the complementary function

$$Pr[X_i(t) > x_i(t)] = 1 - \Phi_{i,t}(x_i(t)).$$

Therefore, when CDFs are used to maximize the acceptance probabilities for all flows in a max-min fair manner, it is equivalent to minimizing the drop probabilities for all flows in a min-max fair manner.

In general, the expected traffic can be modeled using different probability density functions with the corresponding CDFs. One probability density function is to use the empirical distribution that directly corresponds to the historical traffic measurements taken. In particular, let $(r_{i,1}(t), r_{i,2}(t), \ldots, r_{i,M}(t))$ be M measurements taken for flow i at a particular time of day t over some historical data set. Then the empirical CDF is simply defined as

$$\Phi_{i,t}(x_i(t)) = \frac{\#\text{ measurements } \leq x_i(t)}{M}$$

$$= \frac{1}{M}\left(\sum_{k=1}^{M} I(r_{i,k}(t) \leq x_i(t))\right),$$

where $I(r_{i,k}(t) \leq x_i(t))$ is the indicator that the measurement $r_{i,k}(t)$ is less than or equal to $x_i(t)$. For the example shown in Table 5.1, the corresponding empirical CDFs are shown in Figure 5.6. For example in Figure 5.6(a) for OD pair (A, D), a bandwidth allocation of 2 Gb/s would correspond to an acceptance probability of 80% (with the corresponding drop probability of 20%).

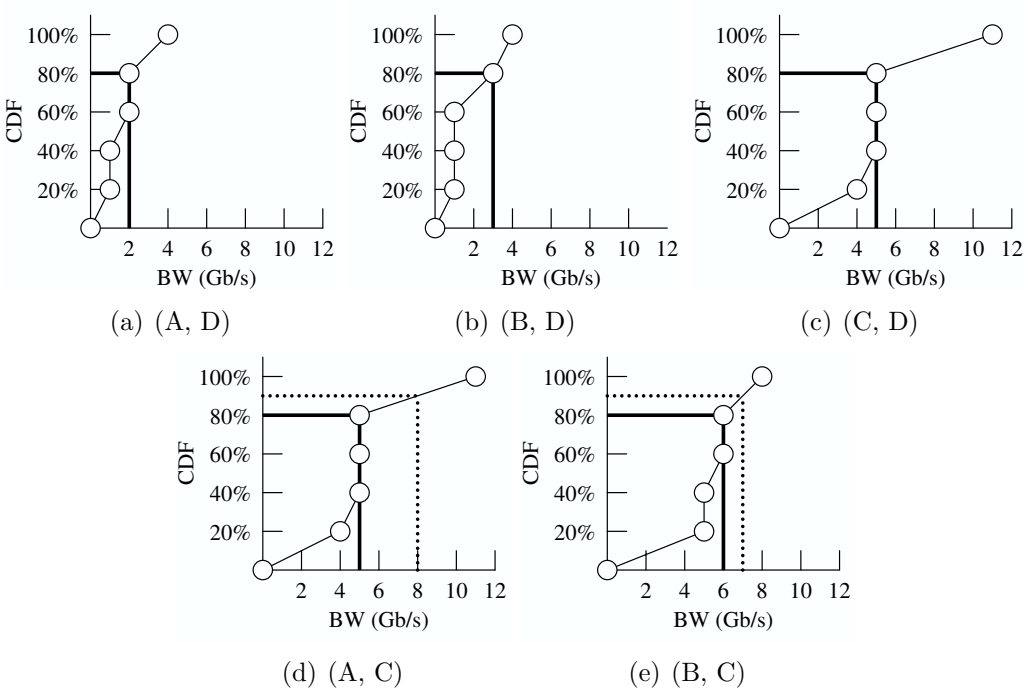
To illustrate how CDF-PSP works, consider again the example shown in Figure 5.3 and Table 5.1. Consider the first iteration of the CDF-PSP water-filling procedure shown in Figure 5.5(a). To simplify notation, we will simply use for example $\beta_{A,D}(x) = \Phi_{A,D}(x)$ to indicate the utility function for flow (A, D) for some time period t, and we will use analogous notations for the other flows.

In the first iteration, the maximum common utility that can be achieved by all free flows is an acceptance probability of $\beta(x) = 80\%$, which corresponds to allocating 2 Gb/s to (A, D), 3 Gb/s to (B, D), 5 Gb/s each to (C, D) and (A, C), and 6 Gb/s to (B, C). This first iteration allocation is shown in bold black lines in Figure 5.6. With this allocation in the first iteration, link CD is again saturated since the sum of the first iteration allocation to flows (A, D), (B, D), and (C, D) is $2 + 3 + 5 = 10$ Gb/s, which would already reach the link capacity of CD. Therefore, the saturated link CD is removed from consideration in subsequent iterations, and the flows (A, D), (B, D), and (C, D) are fixed at the allocation of 2 Gb/s, 3 Gb/s, and 5 Gb/s, respectively.

For link AC, which is shared by flows (A, C) and (A, D), the first iteration allocation is $2 + 5 = 7$ Gb/s, leaving $10 - 7 = 3$ Gb/s of residual capacity. Similarly, for link BC, which is shared by flows (B, C) and (B, D), the first iteration allocation is $3 + 6 = 9$ Gb/s, leaving $10 - 9 = 1$ Gb/s of residual capacity.

In the second iteration, as in shown Figure 5.5(b), the maximum common utility 90% is achieved for the remaining free flows (A, C) and (B, C) by allocating the remaining 3 Gb/s on link AC to flow (A, C) and the remaining 1 Gb/s on link BC to flow (B, C), resulting in a total of 8 Gb/s allocated to (A, C) and 7 Gb/s allocated to (B, C). This second iteration allocation is shown in dotted lines in Figure 5.6. The final CDF-PSP bandwidth allocation is shown in Table 5.1.

Comparing the results for CDF-PSP and Mean-PSP shown in Figure 5.6 and Table 5.1, we see that CDF-PSP was able to achieve a higher worst-case acceptance probability for all flows than Mean-PSP. In particular, the CDF-PSP results shown in Figure 5.6 and Table 5.1 show that CDF-PSP was able to achieve a minimum acceptance probability of 80% for all flows whereas Mean-PSP was only able to achieve a lower worst-case acceptance probability of 70%. For example, for flow (B, D), the bandwidth allocation of 3 Gb/s determined by CDF-PSP corresponds to an 80% acceptance rate whereas the 2 Gb/s determined by Mean-PSP only corresponds to a 70% acceptance rate. The better worst-case result is because CDF-PSP specifically targets the max-min optimization of the *acceptance probability* by using the cumulative distribution function as the objective.



(a) (A, D) (b) (B, D) (c) (C, D)

(d) (A, C) (e) (B, C)

Figure 5.6: Empirical CDFs for flows (A, D), (B, D), (C, D), (A, C), (B, C).

5.4.4 GCDF-PSP: Gaussian-based Max-min Fairness

Another probability Another distribution model that we can use with historical traffic measurements is the Gaussian distribution. GCDF-PSP allows network operators to model future traffic variability scenarios in which historical datasets are not applicable. In addition, we show the problem is simplified to a weighted max-min bandwidth allocation problem.

We denote the CDF for flow i in a Gaussian distribution as $\Phi_{\mu_i,\sigma_i^2}(x)$, where μ_i and σ_i are the mean and standard deviations for flow i, which can be derived from the traffic measurements. To simplify the notation in our bandwidth allocation problem, we will simply use μ_i, σ_i, and $\Phi_{\mu_i,\sigma_i}(x_i)$ to denote the mean, standard deviation, and CDF for a particular time of day t, without any subscript t, unless otherwise needed.

To apply the Gaussian CDF model as utility functions in the water-filling algorithm, at each iteration, instead of finding a bandwidth allocation $\vec{x}(t)_{\bar{p}} = (x_1(t), x_2(t), \ldots, x_N(t))$ that can achieve the largest acceptance probability $\bar{p}$, we

find a bandwidth allocation $\vec{x}(t)_{\bar{\lambda}}$ that can achieve the largest scaling factor $\bar{\lambda}$ in the following equation:

$$x_i = \mu_i + \sigma_i \, \bar{\lambda}.$$

This is equivalent to solving the weighted max-min fair allocation problem using the standard deviations $(\sigma_1, \sigma_2, \ldots, \sigma_N)$ as *weights* and the means $(\mu_1, \mu_2, \ldots, \mu_N)$ as *offsets*.

The reduction rests upon the observation that the inversed CDF for a Gaussian distribution can be written as

$$x_i = \Phi^{-1}_{\mu_i, \sigma_i^2}(p) = \mu_i + \sigma_i \, \Phi^{-1}(p) = \mu_i + \sigma_i \, \sqrt{2} \, \mathrm{erf}^{-1}(2p - 1)$$

which is interpreted as the minimum allocation of x_i to ensure an *acceptance probability* of p, where

$$\Phi^{-1}(p) = \sqrt{2} \, \mathrm{erf}^{-1}(2p - 1)$$

is the inverse *standard* normal cumulative distribution function defined for $\mu = 0$ and $\sigma = 1$. Therefore, we can see that $\bar{\lambda}$ can be derived from the maximum acceptance probability $\bar{p}$ as follows:

$$\bar{\lambda} = \Phi^{-1}(\bar{p}) = \sqrt{2} \, \mathrm{erf}^{-1}(2\bar{p} - 1).$$

Given that $\Phi^{-1}(p)$ is an increasing function, it follows that maximizing $\bar{\lambda}$ in the weighted max-min fair allocation is is equivalent to maximizing $\bar{p}$ in the utility max-min fair allocation problem. This reduction simplifies the bandwidth allocation problem.

5.5 Experimental Setup

We employed NS2 based simulations to evaluate our PSP methods on two large real networks.

US: This is the backbone of a large service provider in the US, and consists of around 700 routers and thousands of links ranging from T1 to OC768 speeds.

EU: This is the backbone of a large service provider in Europe. It has a similar network structure as the US backbone, but it is larger with about 150 more routers and 500 more links.

While the results for the individual networks cannot be directly compared to each other because of differences in their network characteristics and traffic behavior, multiple network environments allow us to explore and understand the performance of our PSP methods for a range of diverse scenarios.

5.5.1 Normal Traffic Demand

For each network, using the methods outlined in [34], we build ingress router to egress router traffic matrices from several weeks worth of sampled Netflow data that record the traffic for that network [31]: US (07/01/07−09/03/07) and EU (11/18/06−12/18/06 & 07/01/07−09/03/07). For each time interval τ, the corresponding OD flows are represented by a $N \times N$ traffic matrix where N is the number of access routers providing ingress or egress to the backbone, and each entry contains the average demand between the corresponding routers within that interval. The above traffic data are used both for creating the normal traffic demand for the simulator as well as for computing the corresponding bandwidth allocation matrices for the candidate PSP techniques. One desirable characteristic from a network management, operations and system overhead perspective is to avoid too many unnecessary fine time scale changes. Therefore, one goal of our study was to evaluate the effectiveness of using a single representative bandwidth allocation matrix for an extended period of time. An implicit hypothesis is that the bandwidth allocation matrix does not need to be computed and updated on a fine timescale. To this end, in the simulations, we use a finer timescale traffic matrix with $\tau = 1$ min for determining the normal traffic demand, and a coarser timescale 1 hour interval for computing the bandwidth allocation matrix from historical data sets.

5.5.2 DDoS Attack Traffic

To test the robustness of our PSP approach, we used two different types of attack scenarios for evaluation – a *distributed* attack scenario for the US backbone and a *targeted* attack scenario for the EU backbone. As we shall see in Section 5.6, PSP is very effective in both types of attacks. In particular, we used the following.

US DDoS: For the US backbone, the attack matrix that we used for evaluation is based on large DDoS alarms that were actually generated by a commercial DDoS detection system deployed at key locations in the network. In particular, among the actual large DDoS alarms there were generated during the period of 6/1/05 to 7/1/06, we selected the largest one involving the most number of attack flows as the attack matrix. This was a *highly distributed* attack involving 40% (nearly half) of the ingress routers as attack sources and 25% of the egress routers as attack destinations. The number of attack flows observed at a single ingress router was up to 150 flows, with an average of about 24 attack flows from each ingress router. The attacks were distributed over a large number of egress routers. Although the actual attacks were large enough to trigger the DDoS alarms, they did not actually cause overloading on any backbone link. Therefore, we scaled up each attack flow to an average of 1% of the ingress router link access capacity. Since there were many flows, this was already sufficient to cause overloading on the network.

EU DDoS: For the Europe backbone, we had no commercial DDoS detection logs available. Therefore, we created our own synthetic DDoS attack data. To evaluate PSP under different attack scenarios, we created a *targeted* attack scenario in which all attack flows are targeted to only a small number of egress routers. In particular, to mimic the US DDoS attack data, we randomly selected 40% of ingress routers to be attack sources. However, to create a targeted attack scenario, we purposely selected at random only 2% of the egress routers as attack destinations. With only 2% of the egress routers involved as attack destinations, we concentrated the attacks from each ingress router to just 1-3 destinations with demand set at 10% of the ingress router link access capacity.

5.5.3 NS2 Simulation Details

Our experiments are implemented using NS2 simulations. This involved implementing the 2-class bandwidth allocation, and simulating both the normal and DDoS traffic flows.

Bandwidth Allocation and Enforcement: The metering and class differentiation of packets are implemented at the perimeter of each network using the differentiated service module in NS2, which allows users to set rate limits for each individual OD pair. Our simulation updates the rate limits hourly by pre-computing the bandwidth allocation matrix based on the historical traffic matrices that were collected several weeks prior to the attack date: US (07/01/07−09/02/07) and EU (11/18/06−12/17/06 & 07/01/07−09/02/07).

The differentiated service module marks incoming packets into different priorities based on the configured rate limits set by our bandwidth allocation matrix and the estimated incoming traffic rate of the OD pair. Specifically, we implemented differentiated service using TSW2CM (Time Sliding Window with 2 Color Marking), an NS2 provided policer. As its name implies, the TSW2CM policer uses a sliding time window to estimate the traffic rate.

If the estimated traffic exceeds the given threshold, the incoming packet is marked into the low priority class; otherwise, it is marked into the high priority class. We then use existing preferential dropping mechanisms to ensure that lower priority packets are preferentially dropped over higher priority packets when memory buffers get full. In particular, WRED/RIO[1] is one such preferential dropping mechanism that is widely deployed in existing commercial routers [29, 7, 5]. We used this WRED/RIO mechanism in our NS2 simulations.

Traffic Simulation: For simulation data (testing phase), we purposely used a different data set than the traffic matrices used for bandwidth allocation (learning phase). In particular, for each network, we selected a week-day outside of the days used for bandwidth allocation, and we considered 48 1-minute time intervals (one every 30-minutes) across the entire 24 hours of this selected day. The exact

[1]RIO is WRED with two priority classes.

date that we selected to simulate normal traffic is 09/03/07 for both the US and EU networks. Recall that for a given time interval τ, we compute normal and DDoS traffic matrices that give average traffic rates across that interval. These matrices are used to generate the traffic flows for that time interval. Both DDoS and network traffic are simulated as constant bandwidth UDP streams with fixed packet sizes of 1 kB.

5.6 Experimental Results

We begin our evaluations in Section 5.6.1 by quantifying the potential extent and severity of the problem that we are trying to address – the amount of collateral damage in each network in the absence of any protection mechanism. We then develop an understanding of the damage mitigation capabilities and properties of our PSP mechanism, first at the network level in Section 5.6.2 and then at the individual OD-pair level in Section 5.6.3. Section 5.6.4 explores the effectiveness of the proposed schemes under scaled attacks, and Section 5.6.5 presents the results under multi-path routing.

In the following results, we shall use the term No-PSP to refer to the baseline scenario with no surge protection, while we use the terms Mean-PSP, CDF-PSP and GCDF-PSP to refer to the PSP schemes based on mean, empirical CDF and Gaussian CDF water-filling bandwidth allocation algorithms respectively. Recall that an OD pair is considered as (i) an ***attacked OD pair*** if there is attack traffic along that pair, (ii) a ***crossfire OD pair*** if it shares at least one link with an OD pair containing attack traffic, and (iii) a ***non-crossfire OD pair*** if it is neither an *attacked* nor a *crossfire* OD pair.

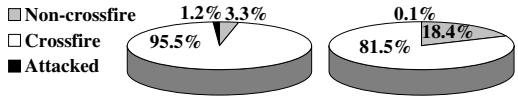
5.6.1 Potential for Collateral Damage

We first explore the extent to which OD pairs and their offered traffic demands are placed in potential harm's way because they share network path segments with a given set of attack flows. In Figure 5.7, we report the relative

proportion of OD pairs in the categories of *attacked*, *crossfire*, and *non-crossfire* OD pairs for both the US and EU backbones.

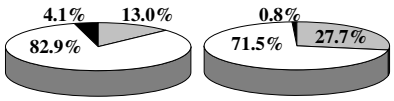
As described in Section 5.5.2, 40% of the ingress routers and 25% of the egress routers were involved in the DDoS attack on the US backbone. In general, for a network with N ingress/egress routers, there are N^2 possible OD pairs (the ratio of routers to OD pairs is 1-to-N). For the US backbone, with about 700 routers, there are nearly half a million OD pairs. Although 40% of the ingress routers and 25% of the egress routers were involved in the attack, the number of attack destinations from each ingress router was on average about 24 egress routers, resulting in just 1.2% of the OD pairs under direct attack. In general, because the number of OD pairs grows quadratically with N (i.e. N^2), even in a highly distributed attack scenario where the attack flows come from all N routers, the number of OD pairs under direct attack may still only correspond to a small percentage of OD pairs. For the EU backbone, there are about 850 routers and about three quarters of million OD pairs. For the targeted attack scenario described in Section 5.5.2, 40% of the ingress routers were also involved in the DDoS attack, but the attacks were concentrated to just 2% of the egress routers. Again, even though 40% of the ingress routers were involved, only 0.1% of the OD pairs, among N^2 OD pairs, were under direct attack.

In general, the percentage of OD pairs that are in the crossfire of attack flows depends on where the attacks occurred and how traffic is routed over a particular network. For the US backbone, we observe that the percentage of crossfire OD pairs is very large (95.5%), causing substantial collateral damage even though the attacks were directed over only 1.2% the OD pairs. This is somewhat expected given the distributed nature of the attack where a high percentage of both ingress and egress routers were involved in the attack. For the EU backbones, the observed percentage of crossfire OD pairs is also very large (83.5%). This is somewhat surprisingly because the attacks were targeted to only a small number of egress routers. This large footprint can be attributed to the fact that even a relatively small number of attack flows can go over common links that were shared by a vast majority of other OD pairs.



(a) US. (b) Europe.

Figure 5.7: The percentage of the number of the three OD pair types classified under an attack traffic.



(a) US. (b) Europe.

Figure 5.8: The proportion of normal traffic demand corresponding to the three types of OD pairs.

We next depict the relative proportions of the overall normal traffic demand corresponding to each type of OD pairs. While the classification of the OD pairs into the 3 categories is fixed for a given network and attack matrix, the relative traffic demand for the different classes is time-varying, depending on the actual normal traffic demand in a given time interval. Figure 5.8 presents a breakdown of the total normal traffic demands for the 3 classes across the 48 time intervals that we explored. Note that for both the networks, crossfire OD pairs account for a significant proportion of the total traffic demand. Figures 5.7 and 5.8 together suggest that an attack directed even over a relatively small number of ingress-egress interface combinations, could be routed around the network in a manner that can impact a significant proportion of OD pairs and overall network traffic.

The results above provide us an indication of the potential "worst-case" impact footprint that an attack can unleash, if its strength is sufficiently scaled up. This is because a crossfire OD pair will suffer collateral packet losses only if some link(s) on its path get congested. While the above results do not provide any measure of actual damage impact, they do nevertheless point to the existence of a real potential for widespread collateral damage, and underline the importance and urgency of developing techniques to mitigate and minimize the extent of such damage.

Table 5.2: Collateral damage in the absence of PSP with the 10^{th} and 90^{th} percentile indicated in the brackets.

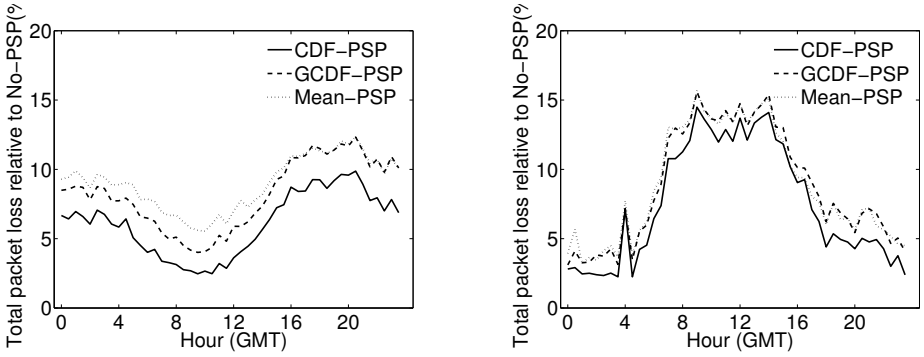
	Impacted OD Pairs(%)	Impacted Demand(%)	Mean packet loss rate of impacted OD pairs(%)
US	41.37 [39.64, 42.72]	37.79 [35.16, 39.37]	49.15 [47.62, 50.43]
EU	43.18 [38.48, 47.81]	45.33 [38.90, 52.05]	68.11 [65.51, 70.46]

We next consider the actual collateral damage induced by the specified attacks in the absence of any protection scheme. We define a crossfire OD pair to be *impacted* in a given time interval, if it suffered some packet loss in that interval. Table 5.2 presents (i) the total number of, and (ii) traffic demand for the impacted OD pairs as a percentage of the corresponding values for all crossfire OD pairs, and (iii) the mean packet loss rate across the impacted OD pairs. To account for time variability, we present the average value (with the 10^{th} and 90^{th} percentile indicated in the brackets) for the three metrics across the 48 attacked time intervals. Overall, the tables show that not only can the attacks impact a significant proportion of the crossfire OD pairs and network traffic, but that they can cause severe packet drops in many of them. For example, in the EU network, in 90% of the time intervals, (i) at least 39.64% of the cross-fire OD pairs were impacted, and (ii) the average packet loss rate across the impacted OD pairs was 47.62% or more. To put these numbers in proper context, note that TCP, which accounts for the vast majority of traffic today, is known to have severe performance problems once the loss rate exceeds a few single-digit percentage points.

5.6.2 Network-wide PSP Performance Evaluation

We start the evaluation of PSP by focusing on network-wide aggregate performance for crossfire OD pairs and note the consistent substantially lower loss rates under either Mean-PSP, GCDF-PSP or CDF-PSP across the entire day.

Total Packet Loss Rate: For each attack time interval, we compute the ***total***



(a) US (b) EU

Figure 5.9: The crossfire OD pair total packet loss rate ratio over No-PSP across 24 hours.(48 attack time intervals, 30 minutes apart).

Table 5.3: The time-averaged crossfire OD-pair total packet loss rate with the 10^{th} and 90^{th} percentile indicated in the brackets.

	No-PSP	Mean-PSP	GCDF	CDF-PSP
US	17.93 [16.40, 18.79]	1.63 [1.02, 2.14]	1.49 [0.77, 2.12]	1.11 [0.47, 1.71]
EU	30.48 [27.22, 32.86]	2.73 [1.21, 4.54]	2.72 [1.12, 4.58]	2.32 [0.79, 4.22]

packet loss rate which is the total number of packets lost as a percentage of the total offered load from all crossfire OD pairs. Table 5.3 summarizes the mean, 10^{th} and 90^{th} percentile of the total packet loss rates across 48 attack time intervals. The mean loss rates under No-PSP in US and EU networks are 17.93% and 30.48%, respectively. The loss rate is relatively stable across time as indicated by the tight interval between the 10^{th} and 90^{th} percentile numbers. In contrast, the mean loss rate is much smaller, less than 3%, for either PSP scheme. Figure 5.9 shows the loss rate across time, for the 3 PSP schemes, expressed as a percentage of the corresponding loss rates under No-PSP. Note that even though the attack remains the same over all 48 attack time intervals, the normal traffic demand matrix is time-varying, and hence the observed variability in the time series. In particular, we observe comparatively smaller improvements during the the network traffic peak times, such as 12PM (GMT) in the EU backbone and 6PM (GMT) in the US

Table 5.4: The time-averaged total packet loss reduction with the 10^{th} and 90^{th} percentile indicated in the brackets.

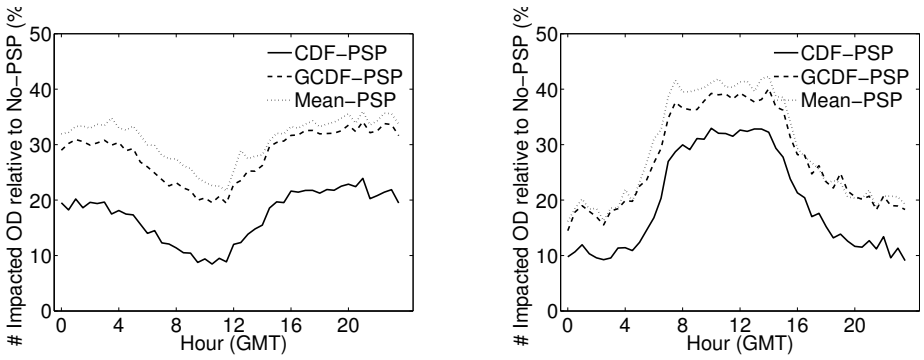
	Reduction ratio from No-PSP to Mean-PSP	Reduction ratio from No-PSP to CDF-PSP	Reduction ratio from Mean-PSP to CDF-PSP	Reduction ratio from GCDF-PSP to CDF-PSP
US	91.00 [88.56, 93.89]	93.90 [90.77, 97.21]	34.75 [20.06, 53.09]	27.36 [19.47, 38.67]
EU	91.17 [85.79, 96.17]	92.51 [86.46, 97.58]	19.90 [4.01, 41.58]	19.90 [6.21, 35.45]

Table 5.5: The time-averaged number of impacted OD-pairs with the 10^{th} and 90^{th} percentile indicated in the brackets.

	No-PSP	Mean-PSP	GCDF-PSP	CDF-PSP
US	41.37 [39.06, 42.73]	12.85 [9.58, 14.58]	11.73 [8.38, 13.84]	7.16 [3.94, 9.24]
EU	43.18 [38.43, 47.94]	12.81 [7.28, 19.70]	12.10 [7.05, 18.48]	8.79 [3.84, 15.46]

backbone. This behavior is because the amount of traffic that could be admitted as high priority is bounded by the network's carrying capacity. During high demand time intervals, on one hand, links will be more loaded increasing the likelihood of congestion and overload. On the other hand, more packets will get classified as low priority, increasing the population size that can be dropped under congestion and overload. Table 5.4 summarizes the performance improvements for the PSP schemes in terms of relative loss rate reduction of Mean-PSP and CDF-PSP across the different time intervals. For each network, on average, all PSP schemes reduce the loss rate in a time interval by more than 90% from the corresponding No-PSP value. In addition, CDF-PSP has consistently better performance than Mean-PSP and GCDF-PSP in both networks, while GCDF-PSP has a lower loss rate than Mean-PSP in most of the time. Take US network as an example, CDF-PSP reduces the loss rate of Mean-PSP and GCDF-PSP by 34.74% and 27.36, respectively.

Number of impacted crossfire OD pairs: We next determine the number of impacted OD pairs, ie., the crossfire OD pairs that suffer some packet loss at each time interval. It is desirable to minimize this number, since many important



(a) US (b) EU

Figure 5.10: The ratio of number of crossfire OD-pairs with packet loss over No-PSP across 24 hours.(48 attack time intervals, 30 minutes apart).

Table 5.6: The time-averaged reduction of number of impacted OD-pairs with the 10^{th} and 90^{th} percentile indicated in the brackets.

	Reduction ratio from No-PSP to CDF-PSP	Reduction ratio from No-PSP to CDF-PSP	Reduction ratio from Mean-PSP to CDF-PSP	Reduction ratio from GCDF-PSP to CDF-PSP
US	69.05 [65.20, 75.64]	82.82 [78.11, 90.22]	45.47 [35.12, 59.30]	40.30 [31.71, 53.53]
EU	71.18 [58.62, 81.49]	80.42 [67.66, 90.36]	34.94 [21.72, 47.60]	31.45 [16.33, 44.70]

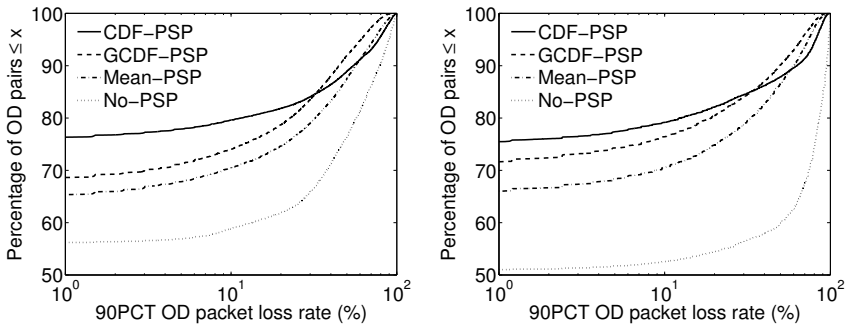
network applications including real-time gaming and VOIP are very sensitive to and experience substantial performance degradations even under relatively low packet loss rates. For each of the 48 attack time intervals, we determine the number of impacted crossfire OD pairs as a percentage of the total number of crossfire OD pairs with non-zero traffic demand in that time interval. We summarize the mean and the 10^{th} and 90^{th} percentiles from the distribution of the resulting values across the 48 time intervals in Table 5.5 for No-PSP and the three PSP schemes. The mean proportion of impacted OD pairs drops from a high of 41.37% under No-PSP to 12.85% for Mean-PSP, 11.73% for GCDF-PSP and 7.16% for CDP-PSP. We present the time series of the proportion of impacted OD pairs for the three PSP schemes (normalized by the corresponding value for No-PSP) across the 48 time intervals in Figure 5.10, and summarize the savings from the three PSP

schemes in Table 5.6. Across all the time intervals, we note that a high percentage of crossfire OD pairs had packet losses under-No-PSP, and that both PSP schemes dramatically reduce this proportion, with CDF-PSP consistently having the lowest proportion of impacted OD pairs. Considering the Table 5.6 , the proportion of impacted OD pairs in the US network is reduced, on average, by over 69% going from No-PSP to Mean-PSP. From Mean-PSP to CDF-PSP, the proportion drops, on average, by a further substantial 45.47%.

5.6.3 OD pair-level Performance

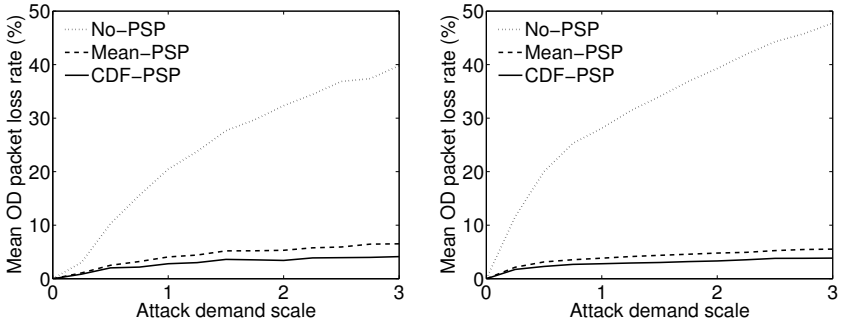
In Section 5.6.2, we explored the performance of the PSP techniques from the overall network perspective. We focus the analysis below on the performance of individual crossfire OD pairs across time.

In particular, we analyze the magnitude of packet losses for different crossfire OD pairs. An OD-pair can have different loss rates at different attack time intervals, and here for each crossfire OD pair, we consider the 90^{th} percentile of these loss rates across time, where we consider only time intervals where that OD pair had non-zero traffic demand. Figure 5.11 shows the cumulative distribution function (CDF) of this 90^{th} ***percentile packet loss rate*** across all crossfire OD-pairs, except those that had no traffic demand during the entire 48 attack time intervals. In the figure, a given point (x, y) indicates that for y% of crossfire OD-pairs, in 90% of the time intervals in which that OD pair had some traffic demand, the packet loss was at most x%. The most interesting region from a practical performance perspective lies to the left of the graph for low values of the loss rate. This is because many network applications and even reliable transport protocols like TCP have very poor performance and are practically unusable beyond a loss rate of a few percentage points. Focussing on $0 - 10$% loss rate range which is widely considered to include this 'habitable zone of loss rates', the figure shows that both Mean-PSP, GCDF-PSP and CDF-PSP have substantially higher percentage of OD pairs in this zone, compared to No-PSP, and that CDF-PSP has significantly better performance. For example, the US network, the percentage of OD pair with less than 10% loss rate increases from just 58.84% for No-PSP



(a) US (b) EU

Figure 5.11: CDF of the 90 percentile packet loss rate for all crossfire OD pairs.



(a) US (b) EU

Figure 5.12: The time-averaged mean crossfire OD-pair packet loss rate as the attack volume scaling factor increases from 0 to 3.

to 70.48% for Mean-PSP, 74.03% for GCDF-PSP and 79.62% for CDF-PSP. The trends are similar for the EU network.

5.6.4 Performance under scaled attacks

Given the growing penetration of broadband connections and the ever-increasing availability of large armies of botnets "for hire", it is important to understand the effectiveness of the PSP techniques with respect to increasing attack intensity. To study this, for each network, we vary the intensity of the attack matrix by scaling the demand of every attack flow by a factor ranging from 0 to 3, in steps of size 0.25. For each value of the scaling factor, we measure the

Table 5.7: Collateral damage under multi-path in the absence of PSP with the 10^{th} and 90^{th} percentile indicated in the brackets. The difference from single-path routing is indicated in parenthesis.

	Impacted OD Pairs(%)	Impacted Demand(%)	Mean packet loss rate of impacted OD pairs(%)
US	37.19 (-4.18) [33.76, 39.10]	36.02 (-1.77) [31.59, 38.32]	47.81 (-1.34) [45.89, 49.73]
EU	44.91(+1.73) [39.50, 48.58]	47.63 (+2.30) [43.20, 50.91]	50.13 (-17.98) [46.10, 54.42]

time-averaged *mean OD packet loss rate*, which measures the average packet loss rate across all crossfire OD pairs with non-zero traffic demand, across eight 1-min. time intervals, equally spaces across 24 hours. Figure 5.12 shows that the loss rate under No-PSP increases much faster than under Mean-PSP and CDF-PSP, as the attack intensity increases. This is because under No-PSP, all the normal traffic packets have to compete for limited bandwidth resources with the attack traffic, while with our protection scheme only normal traffic marked in low priority class is affected by the increasing attack. Therefore, even in the extreme case when the attack traffic demand is sufficient to clog all links, our protection scheme can still guarantee that the normal traffic marked in the high priority class goes through the network. Consequently, our PSP schemes are much less sensitive to the degree of congestion, as evident by the much slower growth of the drop rate. For example, in the US network, as the scale factor increases from 1 to 3, under No-PSP, the mean drop rate jumped from slightly above 20% to almost 40%. In comparison, under CDF-PSP, the mean loss rate increases very little from less than 3% to 4% over the same range of attack intensities. The trends demonstrate that across the range of scaling factor values, both the PSP schemes are very effective in mitigating collateral damage by keeping loss rates low, with CDF-PSP having an edge over Mean-PSP.

5.6.5 Multi-path Evaluation

Finally, we investigate the impact of multi-path routing to our attack scenarios and protection schemes. Specifically, as an example, we consider a Cisco

Table 5.8: The time-averaged crossfire OD-pair total packet loss rate with the 10^{th} and 90^{th} percentile indicated in the brackets. The difference from single-path routing is indicated in parenthesis.

	No-PSP	Mean-PSP	GCDF	CDF-PSP
US	16.33 (-1.60) [14.81, 17.46]	1.46 (-0.17) [0.93, 1.96]	1.34 (-0.15) [0.70, 1.93]	0.98 (-0.13) [0.43, 1.50]
EU	22.60 (-7.88) [19.86, 25.41]	2.28 (-0.45) [1.12, 3.49]	2.10 (-0.62) [0.84, 3.33]	1.93 (-0.39) [0.72, 3.23]

router implementation of a multi-path load balancing scheme called Equal-Cost Multi-Path (ECMP) [8] routing. Here, we revisit the potential of collateral damage in Table 5.7 and the network-wide performance evaluation in Table 5.8. Besides the mean, 90^{th} percentile and 10 th percentile numbers, we also indicate the difference of mean from the results under single-path experiments.

As shown in Table 5.7, when a routing algorithm has the ability to route traffic on multiple paths, the degree of damage could be reduced in term of packet loss because link congestion can be alleviated by load balance traffic. For example, the packet loss rate in US and Europe networks were reduced by 1.34% and 17.98%, respectively. However, the range of damage would also be extended because attack traffic is spread across more links. For example, the number of impacted OD pairs and demand in Europe were increased by 1.73% and 2.30%, respectively. Therefore, while multi-path routing could temporarily alleviate the degree of damage, it also creates more potential damage of collateral damage, especially when the volume of attacks could easily be further increased.

We then observe the impact of multi-path routing on our schemes. As shown in Table 5.8, with multi-path routing, the total packet loss rate is reduced under each of our protection schemes. The improvement of our PSP protection scheme in multi-path routing is slightly less than single-path routing for two reasons. One reason is multi-path routing has greater impact to No-PSP because the packet loss rate of No-PSP is heavily depending on link congestions. Second reason is the traffic load distribution on links in multi-path routing may not accurately match to the static load estimation in our bandwidth allocation algorithm. As a result, our PSP scheme could over or under allocate certain link capacity and limit the

improvement. Nevertheless, our PSP protection schemes still significantly reduce loss rate and with CDF-PSP performed the best followed by GCDF-PSP and Mean-PSP.

5.7 Discussion

Although our PSP protection mechanism could effectively reduce the collateral damage of a DDoS attack, there are some limitations and shortcoming of the approach. In this section, we would like to address each of them as the following. 1) Our approach is designed to reduce the collateral damage of a flooding DDoS attack in a network. While the problem itself is important and interesting to network operators, our approach cannot protect endhost nor defense against those non-bandwidth based attacks which target the network protocol or endhost resources instead of the intermediate network. 2) Our approach is a first-line defense mechanism which aims to effectively mitigate attack damage in a timeliness fashion. While our approach has the strength of scalability and cost, it doesn't have the ability, like previous defense mechanisms, to accurately identify individual attack flows and eliminate them from network by blocking or filtering. Therefore, our approach is orthogonal to those traditional approaches, and we will still recommend to deploy other sophisticated defense systems along with PSP mechanism to further improve network performance. 3) A fundamental limitation of our approach is that we rely on traffic stability to allocate bandwidth without exploring application level packet information. As a result, our approach could treat any bursty traffic as suspicious attack traffic. It will particularly causes problem when the burst traffic is consistent of a bunch of legitimate flows, such as flash crowd. However, the flash crowd traffic would not always be dropped by PSP for two reasons. First, flash crowd traffic only gets dropped when a link congestion actually occurs. Second, because we tend to fully allocate bandwidth based on traffic statistic not simply average traffic, the OD pair with flash crowd traffic could receive higher rate limit. Finally, although flash crowd traffic could be dropped preferentially at congested links, it seems to be a reasonable decision for network operator

from the fairness stand point of view because the flash crowd traffic shouldn't grab majority of the bandwidth regardless the users are legitimate or not.

5.8 Conclusion

PSP provides network operators with a broad first line of proactive defense against DDoS attacks, significantly reducing the impact of sudden bandwidth-based attacks on a service provider network. Among its salient features, PSP is readily deployable using existing router mechanisms, and PSP does not rely on any unauthenticated packet header information. The latter feature makes the solution resilient to evading attack schemes that launch many seemingly legitimate TCP connections with spoofed IP addresses and port numbers. By taking into consideration traffic variability observed in traffic measurements, our proactive protection solution can ensure the maximization of the acceptance probability of each flow in a max-min fair manner, or equivalently the minimization of the drop probability in a min-max fair manner. Our extensive evaluation across two large commercial backbone networks, using both distributed and targeted attacks, shows that up to 95.5% of the network could suffer collateral damage even though the attacks were directed over only 1.2% of the OD pairs. Our solution was able to significantly reduce the amount of collateral damage by up to 97.58% in terms of the number of packets dropped and 90.36% in terms of the number of flows with packet loss. In addition, we show that PSP can maintain low packet loss rates even when the intensity of attacks is increased significantly, and PSP has similar protection performance under multi-path routing.

Chapter 5, in full, is a reprint of the material as it appears in Transaction on Networking. Chou, Jerry; Lin, Bill; Sen, Subhabrata; Spatscheck, Oliver, IEEE/ACM Press, 2009. The dissertation author was the primary investigator and author of this paper.

Chapter 6

Conclusions

This thesis is motivated by the growing disparity between optical capabilities and Moore's Law. While the packet switching approach used in the Internet backbone networks has thus far been able to keep up, it is unclear whether electronic routers that have been used at the core of backbone networks will continue to scale to match future traffic growth or optical link rates. In this thesis, we propose a novel network architecture called COPLAR. In contrast to previously proposed optical network architectures that are based on on-the-fly dynamic reconfigurations of optical circuits and switches, COPLAR is based on send traffic over statically provisioned circuits by default and rerouting excess traffic via other circuits with residual capacity. The proposed approach has been extensively evaluated on real actual networks using a well-known network simulato.. Finally, we demonstrate that our work on bandwidth allocation and routing can be applied to other common network problems and applications, such as network security, quality of services, and resource management. Therefore, the contributions in this thesis are not limited to the field of optical networks.

In fact, we consider our COPLAR framework as an initial step towards investigating the possibilities of using coarse-grain circuits and rerouting-over-circuits in optical networks. In the future, there remains several questions that need to be answered and several approaches that should be further explored. For example, is there a more accurate or precise model to characterize traffic for coarse-grain circuit provisioning? Can we provision circuits to guarantee the worst-case performance

instead of using max-min allocation to strike the balance between throughput and fairness? How do we better coordinate circuit provisioning and adaptive rerouting to achieve even better network performance? Finally, there are a number of implementation details that could be better addressed in the future.

Bibliography

[1] Cisco IOS Netflow, http://www.cisco.com/warp/public/732/Tech/netflow/.

[2] Advanced networking for leading-edge research and education. http://abilene.internet2.edu.

[3] Arbor peakflow. http://www.arbor.net.

[4] CERT CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks.

[5] CISCO CRS-1. www.cisco.com/en/US/products/ps5763/index.html.

[6] CISCO Guard. www.cisco.com/en/US/products/ps5888/index.html.

[7] Distributed weighted random early detection. http://www.cisco.com/univercd/cc/td/doc/product/software/ios111/cc111/wred.pdf.

[8] RFC2992: Analysis of an Equal-Cost Multi-Path algorithm.

[9] Washington Post: The Botnet Trackers, February 16, 2006.

[10] Y. Afek, Y. Mansour, and Z. Ostfeld. Convergence complexity of optimistic rate based flow control algorithms. In *ACM Symposium on Theory of Computing*, pages 89–98, 1996.

[11] Y. Afek, Y. Mansour, and Z. Ostfeld. Phantom: a simple and effective flow control scheme. *ACM SIGCOMM Computer Communication Review*, 26(4):169–182, 1996.

[12] M. Allalouf and Y. Shavitt. Centralized and distributed algorithms for routing and weighted max-min fair bandwidth allocation. *IEEE/ACM Transactions on Networking*, 16(5):1015–1024, October 2008.

[13] D. Applegate and E. Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs. In *Proceedings of ACM conference on Applications, technologies, architectures, and protocols for computer communications*, pages 313–324, 2003.

[14] K. Argyraki and D. R. Cheriton. Active Internet traffic filtering: real-time response to denial-of-service attacks. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 10–10, 2005.

[15] B. Awerbuch and Y. Shavitt. Converging to approximated max-min flow fairness in logarithmic time. In *IEEE INFOCOM*, pages 1350–1357, March 1998.

[16] H. Ballani, Y. Chawathe, S. Ratnasamy, T. Roscoe, and S. Shenker. Off by default! In *ACM HotNets Workshop*, November 2005.

[17] A. Banerjee, J. Drake, J. Lang, B. Turner, D. Awduche, L. Berger, K. Kompella, and Y. Rekhter. Generalized multiprotocol label switching: An overview of signaling enhancements and recovery techniques. *IEEE Communication Magazine*, 39(1):144–150, January 2001.

[18] D. Banerjee and B. Mukherjee. Wavelength-routed optical networks: Linear formulation, resource budgeting tradeoffs, and a reconfiguration study. *IEEE/ACM Transactions on Networking*, 8(5):598–607, 2000.

[19] Y. Bartal, M. Farach-Colton, S. Yooseph, and L. Zhang. Fast, fair, and frugal bandwidth allocation in ATM networks. In *SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 92–101, 1999.

[20] M. Beckmann, C. McGuire, and C. Winsten. *Studies in the Economics and Transportation*. Yale University Press, 1956.

[21] G. Bernstein, B. Rajagopalan, and D. Spears. *OIF UNI 1.0 – controlling optical networks*. White paper, Optical Internetworking Forum, 2001.

[22] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 1987.

[23] N. Buchbinder and J. Naor. Fair online load balancing. In *SPAA '06: Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 291–298, 2006.

[24] Z. Cao and E. W. Zegura. Utility max-min: An application-oriented bandwidth allocation scheme. In *IEEE INFOCOM*, pages 793–801, 1999.

[25] V. Chan. Near-term future of the optical network in question? *IEEE Journal on Selected Areas in Communications*, pages 1–2, December 2007.

[26] S. Chen and K. Nahrstedt. Maxmin fair routing in connection-oriented networks. In *Euro-PDS '98: Proceedings of Euro-Parallel and Distributed Systems Conference*, pages 163–168, 1998.

[27] J. Chou and B. Lin. Coarse optical circuit switching by default, re-routing over circuit for adaptation. *IEEE OSA Journal of Optical Communications and Networking*, 18(1):33–50, January 2009.

[28] J. Chou and B. Lin. Optimal multi-path routing and bandwidth allocation under utility max-min fairness. In *IEEE International Workshop on Quality of Service*, July 2009.

[29] D. Clark and W. Fang. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Transactions on Networking*, 6(4):362–373, August 1998.

[30] Y. Dinitz, N. Garg, and M. X. Goemans. On the single-source unsplittable flow problem. In *IEEE Symposium on Foundations of Computer Science*, pages 290–299, 1998.

[31] N. Duffield, C. Lund, and M. Thorup. Estimating flow distributions from sampled flow statistics. *IEEE/ACM Transactions on Networking*, 13(5):933–946, August 2005.

[32] M. El-Gendy, A. Bose, and K. Shin. Evolution of the Internet QoS and support for soft real-time applications. *Proceedings of the IEEE*, 91(7):1086–1104, July 2003.

[33] A. Elwalid, C. Jin, S. Low, and I. Widjaja. MATE: MPLS adaptive traffic engineering. In *IEEE INFOCOM*, pages 1300–1309, 2001.

[34] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. *ACM SIGCOMM Computer Communication Review*, 30(4):257–270, June 2000.

[35] S. Fischer, N. Kammenhuber, and A. Feldmann. Replex: dynamic traffic engineering based on wardrop routing policies. In *ACM CoNEXT conference*, pages 1–12, 2006.

[36] S. Fischer, H. Räcke, and B. Vöcking. Fast convergence to Wardrop equilibria by adaptive sampling methods. In *ACM Symposium on Theory of Computing*, pages 653–662, 2006.

[37] A. Goel, A. Meyerson, and S. Plotkin. Combining fairness with throughput: online routing with multiple objectives. In *ACM Symposium on Theory of Computing*, pages 670–679, 2000.

[38] A. Greenhalgh, M. Handley, and F. Huici. Using routing and tunneling to combat DoS attacks. In *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet Workshop*, pages 1–1. USENIX Association, 2005.

[39] M. Grossglauser and D. Tse. A framework for robust measurement-based admission control. *IEEE/ACM Transactions on Networking*, 7(3):293–309, June 1999.

[40] R. Hassin and S. Rubinstein. Approximations for the maximum acyclic subgraph problem. *Information Processing Letters*, 51(3):133–140, August 1994.

[41] Y. Hou, Y. Shi, and H. Sherali. Rate allocation in wireless sensor networks with network lifetime requirement. In *Proceedings of ACM Symposium on Mobile ad hoc networking and computing*, pages 67–77, 2004.

[42] Y. Hou, H. Tzeng, and S. Panwar. A generalized max-min rate allocation policy and its distributed implementation using ABR flow control mechanism. In *IEEE INFOCOM*, pages 1366–1375, 1998.

[43] ILOG. Cplex: optimization software. `www.ilog.com/products/cplex/`.

[44] Intel-DANTE. Intel-dante monitoring project. `http://www.cambridge.intel-research.net/monitoring/dante/`.

[45] J. Ioannidis and S. M. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In *Network and Distributed System Security Symposium*, February 2002.

[46] J. M. Jaffe. Bottleneck flow control. *IEEE/ACM Transaction on Communications*, 29(7):954–962, July 1988.

[47] S. Jamin, P. B. Danzig, S. Shenker, and L. Zhang. A measurement-based admission control algorithm for integrated services packet networks. *IEEE/ACM Transactions on Networking*, 5(1):56–70, February 1997.

[48] J. Jin, W. Wang, and M. Palaniswami. Utility max-min fair flow control for multipath communication networks. In *ICSPCS*, pages 17–19, December 2007.

[49] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: responsive yet stable traffic engineering. *ACM SIGCOMM Computer Communication Review*, 35(4):253–264, 2005.

[50] G. Karakostas. Faster approximation schemes for fractional multicommodity flow problems. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 166–173, 2002.

[51] F. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. In *Journal of the Operational Research Society*, pages 237–252, 1998.

[52] J. M. Kleinberg, Y. Rabani, and E. Tardos. Fairness in routing and load balancing. In *IEEE Symposium on Foundations of Computer Science*, pages 568–578, 1999.

[53] M. Kodialam, T. V. Lakshman, J. B. Orlin, and S. Sengupta. Oblivious routing of highly variable traffic in service overlays and IP backbones. *IEEE/ACM Transactions on Networking*, 17(2):459–472, April 2009.

[54] M. Kodialam, T. V. Lakshman, and S. Sengupta. Efficient and robust routing of highly variable traffic. In *ACM HotNets Workshop*, November 2004.

[55] Y. Korilis and A. Lazar. On the existence of equilibria in noncooperative optimal flow control. *J. ACM*, 42(3):584–613, 1995.

[56] R. J. La and V. Anantharam. Optimal routing control: Game theoretic approach. In *Proc. of the 36th IEEE Conference on Decision and Control*, pages 2910–2915, 2000.

[57] K. Lakshminarayanan, D. Adkins, A. Perrig, and I. Stoica. Taming IP packet flooding attacks. In *ACM HotNets Workshop*, 2003.

[58] H.-W. Lee and S. Chong. A distributed utility max-min flow control algorithm. *The International Journal of Computer and Telecommunications Networking*, 50(11):1816–1830, 2006.

[59] X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, and A. Lakhina. Detection and identification of network anomalies using sketch subspaces. In *ACM/USENIX IMC*, pages 147–152, October 2006.

[60] Q. Ma, P. Steenkiste, and H. Zhang. Routing high-bandwidth traffic in max-min fair share networks. *ACM SIGCOMM Computer Communication Review*, 26(4):206–217, 1996.

[61] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: existing techniques and new directions. *ACM SIGCOMM Computer Communication Review*, 32(4):161–174, 2002.

[62] N. Megiddo. Optimal flows in networks with multiple sources and sinks. *Mathematical Programming*, pages 97–107, 1974.

[63] S. Mohanty, C. Liu, B. Liu, and L. Bhuyan. Max-min utility fairness in link aggregated systems. In *IEEE Workshop on High Performance Switching and Routing workshop*, pages 1–7, 2007.

[64] P. Molinero-Fernandez. *Circuit switching in the Internet*. Ph.D Thesis, Stanford University, June 2003.

[65] P. Molinero-Fernandez and N. McKeown. TCP switching: exposing circuits to IP. In *Hot Interconnects*, pages 43–48, September 2001.

[66] J. Moy. OSPF, March 1994. `http://www.ieft.org/rfc/rfc2328.txt`.

[67] M. J. Neely, E. Modiano, and Y.-S. Cheng. Logarithmic delay for n x n packet switches under the crossbar constraint. *IEEE/ACM Transactions on Networking*, 15(3):657–668, June 2007.

[68] G. Owen. *Game Theory*. Academic Press, 1995.

[69] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y.-C. Hu. Portcullis: protecting connection setup from denial-of-capability attacks. *ACM SIGCOMM Computer Communication Review*, 37(4):289–300, 2007.

[70] G. Qiao and M. Yoo. Optical burst switching (OBS) – a new paradigm for an optical Internet. *Journal of High Speed Networks*, 8(1):69–84, 1999.

[71] B. Radunović and J.-Y. L. Boudec. A unified framework for max-min and min-max fairness with applications. *IEEE/ACM Transactions on Networking*, 15(5):1073–1083, 2007.

[72] B. Raghavan and A. C. Snoeren. A system for authenticated policy-compliant routing. *ACM SIGCOMM Computer Communication Review*, 34(4):167–178, October 2004.

[73] B. Ramamurthy and A. Ramakrishnan. Virtual topology reconfiguration of wavelength-routed optical WDM networks. In *IEEE GLOBECOM*, pages 1269–1275, November 2000.

[74] F. Ricciato, S. Salsano, A. Belmonte, and M. Listanti. Off-line configuration of a MPLS over WDM network under time-varying offered traffic. In *IEEE INFOCOM*, pages 57–65, June 2002.

[75] J. Ros and W. Tsai. A theory of convergence order of maxmin rate allocation and an optimal protocol. In *IEEE INFOCOM*, pages 717–726, 2001.

[76] E. Rose, A. Viswanathan, and R. Callon. RFC 3031: Multiprotocol label switching architecture, January 2001.

[77] M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang. Experience in measuring Internet backbone traffic variability:models, metrics, measurements and meaning. In *International Teletraffic Congress (ITC)*, pages 379–388, November 2003.

[78] T. Roughgarden and E. Tardos. How bad is selfish routing? *Journal of the ACM*, 49(2):236–259, 2002.

[79] D. Rubenstein, J. Kurose, and D. Towsley. The impact of multicast layering on network fairness. *IEEE/ACM Transactions on Networking*, 10(2):169–182, April 2002.

[80] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Network support for IP traceback. *IEEE/ACM Transactions on Networking*, 9(3):226–237, June 2001.

[81] S. Shenker. Fundamental design issues for the future Internet. *IEEE Journal on Selected Areas in Communication*, 13(7):1176–1188, September 1995.

[82] S. Shenker. Making greed work in networks: A game-theoretic analysis of switch service disciplines. 3(6):819–831, December 1995.

[83] N. Skrypnyuk. *Road sensitive routing*. Master Thesis, TU München, 2006.

[84] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer. Single-packet IP traceback. *IEEE/ACM Transactions on Networking*, 10(6):721–734, December 2002.

[85] M. Tornatore, G. Maier, and A. Pattavina. WDM network optimization by ILP based on source formulation. In *IEEE INFOCOM*, pages 1813–1821, June 2002.

[86] TOTEM. Geant traffic matrices. `totem.info.ucl.ac.be/dataset.html`.

[87] TOTEM. A toolbox for traffic engineering methods, February 2005. `http://totem.info.ucl.ac.be`.

[88] H. Tzeng and K. Siu. On max-min fair congestion control for multicast ABR service in ATM. *IEEE Journal on Selected Areas in Communications*, 1997.

[89] P. Verkaik, O. Spatscheck, J. V. der Merwe, and A. C. Snoeren. PRIMED: community-of-interest-based DDoS mitigation. In *ACM LSAD Workshop*, pages 147–154, November 2006.

[90] J. Wardrop. Some theoretical aspects of road traffic research. In *Institute of Civil Engineers*, 1952.

[91] G. Weichenberg, V. W. S. Chan, and M. Medard. On the capacity of optical networks: A framework for comparing different transport architectures. *IEEE Journal on Selected Areas in Communications*, 25(6):84–101, August 2007.

[92] L. Xu, H. G. Perros, and G. Rouskas. Techniques for optical packet switching and optical burst switching. *IEEE Communication Magazine*, 2001.

[93] A. Yaar, A. Perrig, and D. Song. Pi: A path identification mechanism to defend against DDoS attacks. In *IEEE Security and Privacy Symposium*, pages 93–107, 2003.

[94] A. Yaar, A. Perrig, and D. Song. An endhost capability mechanism to mitigate ddos flooding attacks. In *IEEE Security and Privacy Symposium*, May 2004.

[95] S. Yao, B. Mukherjee, S. J. B. Yoo, and S. Dixit. A unified study of contention-resolution schemes in optical packet-switched networks. *Journal of Lightwave Technology*, 21(3):672–683, March 2003.

[96] D. K. Y. Yau, J. C. S. Lui, F. Liang, and Y. Yam. Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. *IEEE/ACM Transactions on Networking*, 13(1):29–42, 2005.

[97] M. Yoo, M. Yoo, , and S. Dixit. Optical burst switching for service differentiation in the next-generation optical Internet. *IEEE Communications Magazine*, 39(2):98–104, 2001.

[98] S. J. B. Yoo. Optical-label switching, mpls, mplambdas, and gmpls. *Special Issue on: Prospects and Challenges: Next Generation Optical Network Architectures, Optical Networking Magazine*, 4(3):17–31, May 2003.

[99] S. J. B. Yoo, Y. B. F. Xue, J. Taylor, Z. Pan, J. Cao, M. Jeon, T. Nady, G. Goncher, K. Boyer, K. Okamoto, S. Kamei, and V. Akella. High-performance optical-label switching packet routers and smart edge routers for the next generation Internet. *IEEE Journal on Selected Areas in Communications*, 21(7):1041–1051, September 2003.

[100] Y. Zhang. Abilene traffic matrices. `http://www.cs.utexas.edu/~yzhang/research/AbileneTM`.

[101] R. Zhang-Shen and N. McKeown. Designing a predictable Internet backbone network. In *ACM HotNets Workshop*, San Diego, November 2004.