Routing, Resource Allocation and Network Design for Overlay Networks

A Thesis
Presented to
The Academic Faculty

by

Yong Zhu

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

College of Computing
Georgia Institute of Technology
December 2006

Routing, Resource Allocation and Network Design for Overlay Networks

Approved by:

Professor Mostafa Ammar, Advisor
College of Computing
Georgia Institute of Technology

Professor Jun Xu
College of Computing
Georgia Institute of Technology

Professor Constantine Dovrolis
College of Computing
Georgia Institute of Technology

Professor Ellen Zegura
College of Computing
Georgia Institute of Technology

Professor Chuanyi Ji
College of Enigneering
Georgia Institute of Technology

Date Approved: October 23th, 2006

To my wife Yan, and my parents,

for their love, support and encouragement

ACKNOWLEDGEMENTS

My first, and most earnest, acknowledgment must go to my advisor Dr. Mostafa Ammar, who led me into the exciting area of networking and telecommunications. His insights, experience, encouragement, and uncompromising emphasis on quality research guided me through my entire program of study.

Much of this thesis work is collaboration with Dr. Constantine Dovrolis. I have learned substantially from his detail-oriented attitude, enthusiasm and dedication to precise and high quality research.

I owe a lot to Dr. Admela Jukan for her compassion, dedication, and constant encouragement during my early work in optical networks. This experience provided many important insights and valuable techniques to the thesis work presented here.

Dr. Jun Xu offered me opportunities to work on network security projects and I am grateful for his help at different stages of my graduate study.

I thank the rest of my thesis committee members: Dr. Ellen Zegura and Dr. Chuanyi Ji for their time and efforts in reviewing this work. Their valuable feedback helped me to improve the dissertation in many ways.

I also thank all students and staffs in the Networking and Telecommunications Group, whose friendship and support have turned my journey through graduate school into a pleasure.

Finally, I want to thank my parents, without whom none of these would be possible. I am especially indebted to my wonderful wife Yan, who has continually offered love, support, and inspiration. She also provided valuable help with remarkable intelligence in tackling some challenging problems in my thesis work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Overlay networks have been the subject of significant research and practical interest recently in addressing the inefficiency and ossification of the current Internet. In this thesis, we cover various aspects of overlay network design, including overlay routing algorithms, overlay network assignment and multihomed overlay networks. We also examine the behavior of overlay networks under a wide range of network settings and identify several key factors that affect the performance of overlay networks. Based on these findings, practical design guidelines are also given. Specifically, this thesis addresses the following problems:

- **Dynamic overlay routing:** We perform an extensive simulation study to investigate the performance of available bandwidth-based dynamic overlay routing from three important aspects: *efficiency*, *stability*, and *safety margin*. Based on the findings, we propose a hybrid routing scheme that achieves good performance in all three aspects. We also examine the effects of several factors on overlay routing performance, including network load, traffic variability, link-state staleness, number of overlay hops, measurement errors, and native sharing effects.

- **Virtual network assignment:** We investigate the virtual network (VN) assignment problem in the scenario of network virtualization. Specifically, we develop a basic VN assignment scheme without reconfiguration and use it as the building block for all other advanced algorithms. Subdividing heuristics and adaptive optimization strategies are presented to further improve the performance. We also develop a selective VN reconfiguration scheme that prioritizes the reconfiguration for the most critical VNs.

- **Overlay network configuration tool for PlanetLab:** We develop NetFinder, an automatic overlay network configuration tool to efficiently allocate PlanetLab resources to individual overlays. NetFinder continuously monitors the resource utilization of PlanetLab and accepts a user-defined overlay topology as input and selects the

set of PlanetLab nodes and their interconnection for the user overlay.

- **Multihomed overlay network:** We examine the effectiveness of combining multihoming and overlay routing from the perspective of an overlay service provider (OSP). We focus on the corresponding design problem and examine, with realistic network performance and pricing data, whether the OSP can provide a network service that is profitable, better (in terms of round-trip time), and less expensive than the competing native ISPs.

CHAPTER I

INTRODUCTION

The Internet has been a great success in the past few decades and has provided a whole new way to access and exchange information. The best-effort unicast service model of the Internet has served well in the past for a variety of network services and applications, including the Web, email, and instant messaging. Its success has stimulated enormous growth and wide deployment of network technology and applications.

As the Internet becomes more popular, the need for enhanced services, such as more bandwidth, better security, higher reliability, faster failure recovery, are growing. However, due to the IP addressing scheme, BGP routing paradigm, and other historical reasons, the current Internet is not efficient in addressing these increasing needs for enhanced services. For example, it has been shown that native Internet routing can lead to poor availability and performance [9]. The single route from the source network to a given destination network/prefix may not be always available, while routing policies and traffic engineering practices can (and often do) impose a heavy performance penalty on the resulting end-to-end performance [6].

To achieve improved reliability and performance, *multihoming* has become the mainstream service model for major content providers (see Figure 1). In the most common multihoming scenario, a customer uses one ISP as its primary provider and another as backup. Switching from the primary to the backup can be performed automatically by the border router of the multihomed network when it detects loss of connectivity with the primary ISP. The use of multihoming has seen a dramatic increase in the last few years. It is now estimated that more than 70% of the stub networks in the US are multihomed [24]. In the more advanced form of this model, known as *intelligent route control*, the multihomed source network selects the upstream ISP for every significant destination prefix based on performance and cost considerations [1, 30].

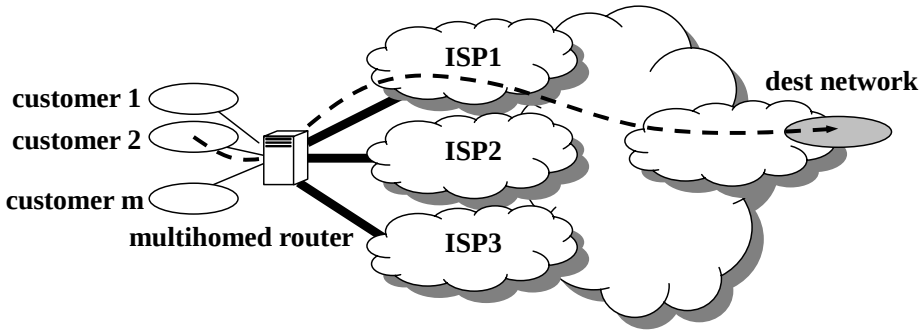


Figure 1: Multihoming architecture

A different approach to improve end-to-end availability and performance is to use *overlay routing* (see Figure 2). Here the traffic between the source and destination is sent through one or more intermediate overlay nodes that are connected through IP tunnels [9,48]. While the IP layer simply provides basic connectivity between overlay nodes, overlay routing has the flexibility to select path based on application specific objectives and current network situation. This decoupling of path selection and data forwarding enables novel, on-demand and evolving services through overlay networks:

First, to deal with the slow fault recovery and routing convergence of BGP [43], overlay networks can bypass broken paths by rerouting traffic through intermediate overlay nodes. The detection of broken paths by overlay nodes can be quickly performed through active probing. Second, the IP routing model is basically a "one-size-fits-all" service, providing the same route independent of performance requirements. Instead, overlay networks can offer different routes to the same destination, depending on the performance metric (*e.g.,* delay, throughput, loss rate) that each application cares for [9]. Third, the fact that interdomain IP routing is largely determined by ISP commercial policies often results in suboptimal paths [72]. Overlay networks can provide better end-to-end performance by routing through intermediate overlay nodes, essentially forcing the flow of traffic in end-to-end paths that would otherwise not be allowed by ISP policies.

Another motivation for overlay networks is that they provide a feasible strategy to test and deploy new network technologies and overcome the Internet ossification. The concept of network virtualization was proposed to allow different virtual networks to co-exist within

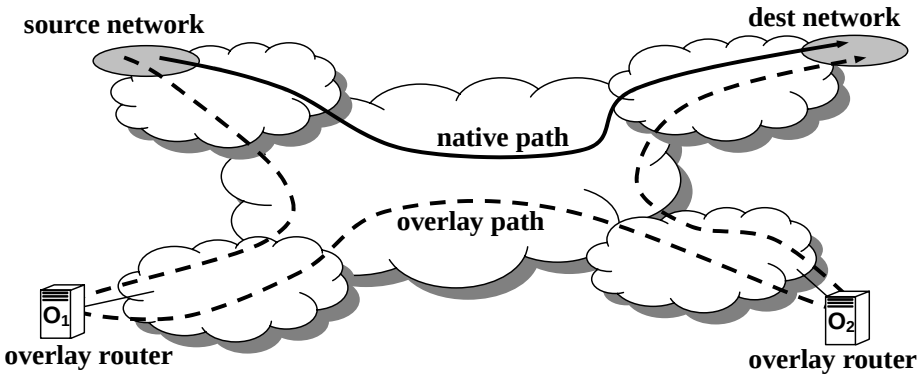


Figure 2: Overlay routing architecture

a shared substrate [57]. In such an approach, a substrate network provider (SNP) provides a common substrate to support a number of diversified virtual networks (DVN). These DVNs, which are essentially specialized overlay networks implemented by *virtual routers* connected through *virtual links*, in turn would provide a range of different communication services to applications, using a variety of protocols and packet formats. In doing so, the burdens of providing an "everything-to-everybody" network architecture or requiring universal agreements on architecture and protocol changes are released. Furthermore, the diversified Internet would make it possible for an individual or organization to rapidly deploy a DVN with potentially global scope, without making any direct investment in physical infrastructure [84].

A common characteristics of all overlay networks is the existence of two or more layers of networks and the separation of routing and packet forwarding between these two layers. First, there is a substrate network composed of substrate nodes (*e.g.,* end-hosts, IP routers), and substrate links. There is also an associated routing functionality that provides the native datagram delivery between substrate nodes. On the top layer is the overlay network that is composed of overlay nodes and links. Specifically, several substrate nodes are selected as the overlay nodes and they are interconnected through overlay links to form the overlay network. Each overlay link is actually a tunnel over the substrate network. Furthermore, the substrate network could be the either the native Internet, or another overlay network built on top of some other substrate network. In all cases, the lowest layer is always the native Internet. An important feature of this overlay architecture is the existence of multiple

alternative overlay paths between a pair of source and destination. When a user flow arrives, the ingress overlay node determines the best overlay path to the destination based on the users objective and the current network states.

Figure 3 illustrates the process of constructing an overlay network on top of a substrate network [1]. The first step is the *overlay topology design* to determine the size of the overlay network and how it is connected. This procedure is usually performed based on application specific requirements, customer information and monetary considerations. For example, full mesh overlay is desirable for small overlay network to provide maximum resilience to network failures. Sparse or hierarchical topology is more desirable for large overlay networks. Structured peer-to-peer overlay networks may require a special overlay topology. Once the overlay topology is determined, an *overlay network assignment* process is performed to map the user-defined overlay topology into subset of the underlying substrate network. Given the overlay topology and the substrate network status, the assignment process will select the substrate node and path for each individual overlay node and link respectively. Again, different applications have different strategies in the overlay network assignment. For example, a content delivery network (CDN) may deploy its replicas close to its users to minimize the delay. A substrate provider supporting DVNs may want to achieve load balancing to avoid hot spots in its network. Finally, an *overlay routing mechanism* needs to be implemented to deliver the user traffic within the overlay network.

This thesis covers various aspects of overlay network design, including the overlay routing algorithms, overlay network assignment and multihomed overlay networks. We also examine the performance of overlay networks under a wide range of network settings and identify key factors that affects the performance of overlay networks. Based on these findings, practical design guidelines are also given. The following sections detail the specific problems we consider.

[1] The virtual network in the scenario of network virtualization is essentially an overlay network constructed on top of the substrate network, therefore, in the rest of this chapter, we will use the terms virtual network and overlay network interchangeably.

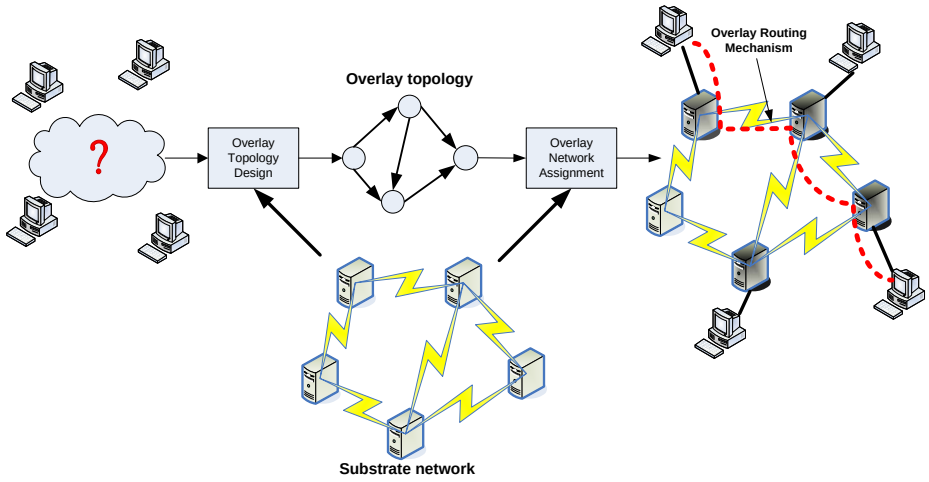


Figure 3: Overview of overlay and virtual networks

1.1 Dynamic overlay routing

It is clear that there is much to be learnt about overlay networks, and that the key debates on the scalability, efficiency, and stability of overlay networks have to be addressed before their wider-scale deployment. In this work, we focus on an aspect of dynamic overlay networks that has been largely unexplored previously, namely, *the use of available bandwidth (avail-bw) measurements in the path selection process*. Previous work on overlay routing assumed that the only information that can be measured or inferred about the underlying native network is related to delays, loss rate, and sometimes TCP throughput. The problem with these metrics is that they are not direct indicators of the traffic load in a path: delays can be dominated by propagation latencies (which do not depend on network load), losses occur after congestion has already taken place, while measurements of TCP throughput can be highly intrusive and they can be affected by a number of factors (such as flow size, advertised window, or TCP stack). The avail-bw, on the other hand, directly represents the additional traffic rate that a path can carry before it gets saturated. Consequently, an overlay node can route a traffic flow (or an aggregation of many flows) to a path only if the maximum throughput of that flow is lower than the avail-bw of the path. The use of avail-bw in overlay routing has recently become possible, based on recent advances in avail-bw measurement techniques and tools [8, 34, 38, 67, 78]. Obviously, if an application has additional requirements on the end-to-end delay or loss rate, then those requirements

can be jointly considered with avail-bw in the path selection process.

This thesis presents an extensive simulation study of dynamic overlay routing based on avail-bw estimation. We first focus on two algorithms that represent two different and general approaches: *proactive* and *reactive* routing. The former attempts to always route a flow in the path that provides the maximum avail-bw, so that the flow can avoid transient congestion due to cross traffic (and overlay traffic) fluctuations. The latter reroutes a flow only when the flow cannot meet its throughput requirement in the current path, and there is another path that can provide higher avail-bw. The routing algorithms are compared in terms of efficiency, stability, and safety margin (or headroom). We show that reactive routing has significant benefits in terms of throughput and stability, while proactive routing is better in providing flows with a wider safety margin. We then propose a hybrid routing scheme that combines the best features of the previous two algorithms. We also examine the effect of several factors, including network load, traffic variability, link-state staleness, number of overlay hops, measurement errors, and native sharing effects. Some of our results are rather surprising. For example, we show that a significant measurement error, even up to 100% of the actual avail-bw value, has a negligible impact on the efficiency of overlay routing. Also, we show that a naive overlay routing algorithm that ignores native sharing between overlay paths performs equally well with an algorithm that has a complete view of the native topology and of the avail-bw in each native link.

1.2 Virtual network assignment

One of the important practical problems in developing a virtual network (VN) is to map the user-defined virtual topology into the substrate network. This is an important issue to both the substrate network provider and the VN users. From the provider's point of view, the mapping or assigning of substrate network resources to individual virtual network is a fundamental functions to support network virtualization. Specifically, each virtual node is assigned to one of the substrate nodes and each virtual link is assigned to a substrate path that connects the corresponding end nodes. In our envisioned scenario, VN demands would arrive at different time instances and request to set up virtual networks with different

topologies and lifetimes. Allowing VNs to be assigned to the substrate network *efficiently* and *on-demand* is desirable for the following reasons:

- Increasing efficiency in the substrate resource utilization would allow the substrate network to accommodate more virtual networks with limited resources and reduce hot spots or congestion.

- On-demand assignment means that the assignment for each virtual network is determined based on the current network situation as the demand arrives. Given that VN demands can arrive at any moment and the substrate provider would not have information regarding future arrivals, it is important for the substrate network provider to be able to make the assignment decision in response to each individual demands as they arrive.

The above function is also important from the VN user's perspective since a good assignment could fully deliver the performance enhancement of the overlay network while poor VN assignments would limit the potential of an overlay network. For example, in designing a resilient overlay network, if all overlay paths between the source and the destination are configured to share a common native link, then the connection will fail when the common link is broken. This is true even if all paths are disjoint at the overlay level.

In this thesis, we are motivated by the above considerations to study the following on-demand VN assignment problem: Upon the arrival of a VN request, assign its topology to the substrate network to achieve *low* and *balanced* load on both substrate nodes and links. A special case of the VN assignment problem can be formulated as an unsplittable flow problem which is NP-hard. Therefore, the VN assignment problem is intrinsically difficult and heuristics will be used to solve the problem. We developed a basic scheme for VN assignment without reconfiguration and use it as a building block for all other advanced algorithms. Subdividing heuristics and adaptive optimization strategies are presented to further improve the performance. We also developed a selective VN reconfiguration scheme that prioritizes the reconfiguration for the most critical VNs. Extensive simulation experiments demonstrate that the proposed algorithms can achieve good performance under a

wide range of network conditions.

1.3 Automatic overlay network configuration in PlanetLab

PlanetLab has been widely used in the networking community to test and deploy new network technologies [58]. It can serve as a testbed for overlay networks. Research groups are able to request a PlanetLab slice in which they can experiment with a variety of wide-area networks and services, such as file sharing, content distribution networks, routing and multicast overlays, QoS overlays, and network measurement tools. One problem faced by PlanetLab users is selecting a set of PlanetLab nodes and interconnecting them to form the desired overlay network.

PlanetLab also serves as a meta testbed on which multiple, more narrowly-defined virtual testbeds can be deployed. For example, the "Internet-in-a-Slice"(*IIAS*) service aims at recreating the Internet's data plane and control plane in a PlanetLab slice [36]. Network researchers can use this infrastructure to experiment with modifications and extensions to the Internet's protocol suite. Currently, IIAS does not provide resource discovery and overlay assignment services and the overlay configuration has to be given explicitly.

Having an automated overlay network assignment service for PlanetLab is important for two reasons. From the PlanetLab operator's perspective, efficient assignments would result in better resource utilization so that the PlanetLab could accommodate more user overlays with limited resources and avoid hot spots or congestion. From the overlay user's perspective, having good assignment service would provide overlay users better choices in building their overlays by avoiding congested links/nodes. A recent study reported that available resources vary significantly in PlanetLab, suggesting that wise placement of application instances can be beneficial [54].

Unfortunately, such a task is usually carried out manually by individual users and sometimes in an ad-hoc manner. Manual configuration is time consuming (especially for networks) and prone to human error. It is not efficient even for small overlay networks since the underlying PlanetLab network has hundreds of nodes and its state fluctuates over time due to failures, congestion and load variation. In this work, we are motivated by the above

considerations and focus on the on-demand overlay network assignment problem: Upon the arrival of a user-defined overlay topology and associated overlay resource requirements (such as minimum available bandwidth, minimum available CPU), find the set of PlanetLab nodes and their interconnections to satisfy the user request.

In this thesis, we develop NetFinder, an automatic overlay network configuration tool in PlanetLab. This tool continuously collects information about the resource utilization of PlanetLab and accepts a user-defined overlay topology as input. NetFinder then performs overlay network assignment by selecting the set of PlanetLab nodes and their interconnection for the desired user overlay.

1.4 Multi-homed overlay networks

To avoid the inefficiencies and fragility of the native Internet routing, the concept of intelligent routing (IR) has been proposed to utilize better alternative paths from the source to the destination instead of sticking to the native IP path. Among various proposals are two most dominant architectures: multihoming and routing overlays. In multihoming, a stub network can dynamically choose among a number of upstream Internet providers. A multihomed network can split its ingress traffic among different providers, and it can steer its egress traffic to the provider with the best performance or least cost. Routing overlays, on the other hand, require a distributed infrastructure of overlay routers. Packets can be forwarded from their source to the destination through one or more intermediate overlay routers, bypassing congested Internet links or networks with poor reliability. In comparison, multihoming is a localized IR architecture that is already quite popular in stub networks, while routing overlays can provide greater flexibility but they require a distributed infrastructure and a greater investment. A comparison study of multihoming and overlay routing has been conducted in [7].

Given the previous two approaches, it is interesting to consider a scenario in which both models are used. Specifically, we envision a new type of Internet provider referred to as *Overlay Service Provider (OSP)* (to distinguish from an ISP) that attempts to offer its customers the combined benefits of multihoming and overlay routing, in terms of improved

performance and reduced cost. The OSP operates a *Multihomed Overlay Network (MON)*, with each MON node being a multihomed router (shown in Figure 4). MON nodes are placed at "key" Internet locations, mostly Internet Exchange Points (IXPs), and the OSP purchases Internet connectivity for each MON node from several locally present ISPs. An OSP customer can connect directly to a MON node if that customer is collocated at the same IXP with that MON node. Major content providers are usually collocated at major IXPs to avoid the cost of leased lines. On the other hand, the OSP is responsible to route a customer's traffic with greater reliability and performance than the customer's current ISP. Note that this is similar to the Internap service model [1].

Further, we envision that the OSP can also provide overlay routing utilizing MON nodes as intermediate overlay routers. Based on the findings of [92], we limit the number of intermediate MON nodes in an overlay route to one. It is rarely the case that more intermediate nodes are needed to improve performance significantly.

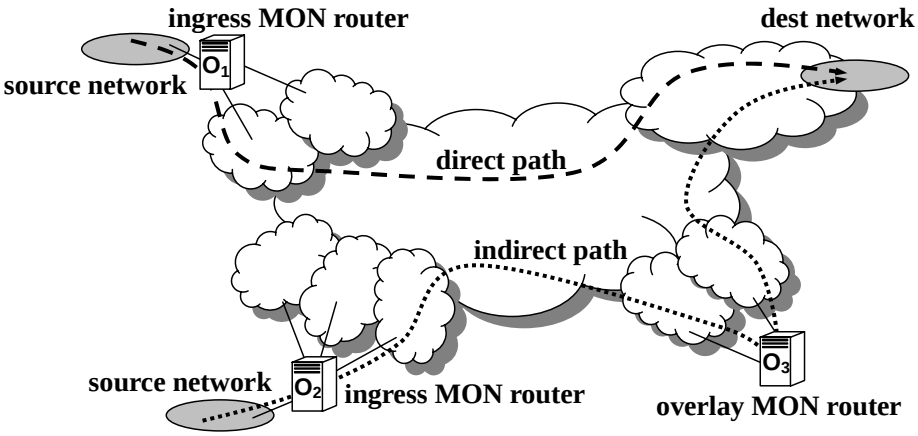


Figure 4: MON architecture

It is interesting that an OSP is "an Internet provider that does not own a network", in the sense that the OSP does not operate any long-distance links or a backbone. Its infrastructure is located at the network edges, and its long-distance communication is conducted from the underlying *native-layer ISPs*. As a matter of fact, early ISPs were often built in the same way, leasing long-distance links from telecommunication providers and placing IP routers only at major aggregation points at the network edge.

In this thesis, we focus on the *MON design problem*, *i.e.,* , where to place MON nodes and how to select the upstream ISPs for each node. We are interested to examine, with realistic network performance and pricing data, whether an OSP can combine multihoming and overlay routing to provide a network service that is, first, profitable, second, better in terms of performance than the competing native ISPs, and third, less expensive than the competing native ISPs. Perhaps surprisingly, we find out that the OSP can meet all three previous objectives. We also observe, however, that the MON design process is crucial (*e.g.,* operating more than 10 overlay nodes, or always routing traffic through the minimum-delay indirect path, rarely leads to profitability in our simulations).

1.5 Outline

The rest of this thesis is organized as follows. Chapter 2 discusses the existing work related to this thesis. Chapter 3 presents our simulation study on dynamic overlay routing based on available-bandwidth measurements. Chapter 4 investigates the problem of virtual network assignment. In Chapter 5, we develop NetFinder, an automatic overlay network configuration tool for PlanetLab. The multihomed overlay network architecture and the corresponding network design is presented in Chapter 6. We finally summarize the thesis and discuss the future work in Chapter 7.

CHAPTER II

RELATED WORK

This chapter provides an overview of the related work on the topics of overlay networks, virtual network assignment, and multihomed overlay network design.

2.1 Overlay networks

An important motivation of overlay networks comes from studies on the limitations of BGP based Internet routing: Paxson finds that the likelihood of encountering a major routing pathology more than doubled between the end of 1994 and the end of 1995, rising from 1.5% to 3.3% [56]. Labovitz *et al.* analyze the delayed behavior of the Internet routing convergence [43]. They report that the delay in Internet interdomain path failovers average three minutes, and some percentage of failovers triggers routing table oscillations lasting up to fifteen minutes. Internet path inflation caused by BGP-policies is studied in [76, 82].

Due to its flexibility to provide enhanced services, overlay networks have become an increasingly active topic in recent years. Over the last few years much has been learnt about overlay networks.The Detour project quantifies the Internet's inefficiencies and argues that Internet behavior can be improved by spreading intelligent routers at key access and interchange points to actively manage traffic [71]. Savage *et al.* conduct a measurement-based study comparing the performance seen using the default Internet path with the potential performance available using some alternate path, and find that in 30-80% of the cases, there is an alternate path with significantly superior quality. Using RTT as the routing metric and packet drop rate as the performance metric, Bauer *et al.* investigate the overlay routing performance under different overlay sizes and traffic patterns [16]. Andersen *et al.* examine the nature of loss and failure in the Internet and their implications on the overlay routing schemes [11]. Rewaskar and Kaur characterize and evaluate the tradeoff between the efficiencies of alternate path routing in improving end-to-end delay and loss, and the overheads

introduced by alternate routing methodology [65]. The impact of the overlay topology on the resulting routing performance and overhead is studied in [47], suggesting that knowledge of the native network topology can significantly benefit the overlay construction. Qiu *et al.* address the effects of selfishness to the overlay routing performance [60]. Tao *et al.* examine the diversity among alternative paths and their implications to the overlay routing performance [83].

Another research thread focuses on enhanced services that can be provided by overlay networks. To name a few, the Resilient Overlay Network (RON) is the first wide-scale overlay implementation and testbed, over which several measurement studies are performed [9]. It relies on probing to monitor the quality of underlying Internet connection between overlay nodes and uses this information to route packet through one of the alternative paths to achieve fast failure detection and recovery. The MBONE is an overlay network layered on top of portions of the physical Internet to support routing of IP multicast packets [25]. Chu *et al.* use an end system overlay approach to efficiently support all multicast related functionality including membership management and packet replication [35]. OverQos is an architecture for providing Internet QoS using overlay networks [79]. Overlay path selection algorithms, focusing on QoS-aware routing, is studied in [48]. The objective of QRON is to find paths between source and destination that satisfy both bandwidth and computational requirement. Keromytis *et al.* propose an Secure Overlay Services (SOS) architecture using a combination of secure overlay tunneling, routing via consistent hashing, and filtering to proactively prevents DoS attacks [39]. Byers *et al.* study how to optimize throughput of large transfers across richly connected, adaptive overlay networks, focusing on the potential of collaborative transfers between peers to supplement ongoing downloads [18].

Overlay networks rely heavily on active probing, raising questions about their scalability and long-term viability. For instance, Nakao *et al.* argue that independent probing by various overlay networks is untenable, and that a "routing underlay" service is needed that will be shared by different overlays [52]. The high cost of overlay network probing is the motivation for the tomography-based monitoring scheme reported in [22]. Furthermore, an ongoing debate focuses on the "selfishness" of overlay routing, and on the potential

performance inefficiency and instability that it can cause [5, 60, 69, 74, 88].

2.2 Virtual network assignment

Network virtualization provides a promising way for addressing the ossification of the Internet by allowing multiple virtual networks co-exist on top of a shared substrate [57]. Different virtual networks provide alternate end-to-end packet delivery systems and may use different protocols and packet formats. One of the fundamental functions of network virtualization is the virtual network assignment, *i.e.,* the mapping or assigning of substrate network resources to individual virtual network nodes and links.

The virtual network assignment problem shares similarities with the virtual circuit routing and virtual private network (VPN) design problems [27, 31, 33, 81]. All of them involve finding paths for source/destination pairs. In both the routing and VPN design problems, locations of source/destination pairs that need to be connected are given as input parameters. In network virtualization, however, the mapping between the VN and the substrate network could be arbitrary, this extra degree of freedom increases the complexity of the problem. Furthermore, in the VPN design as well as the static routing problem, only link utilizations are considered while the VN assignment problem considers load balancing of both substrate nodes and links.

Another unique feature of the VN assignment problem is that requests arrive in terms of a graph, as opposed to a source/destination pair for the ordinary routing problem. Therefore, the VN assignment problem has both online-routing and off-line routing features: within the same VN request, multiple requests for virtual links arrive at the same time (similar to off-line routing), while no information is available for future VN requests (similar to on-line routing). This batch-arrival process allows us to consider multiple concurrent virtual link requests together to use the network resources more efficiently.

The load balancing problem on networks is a generalization of the load balancing problem on unrelated parallel machines [14]. A competitive strategy to minimize congestions in online virtual circuit routing is developed by Aspnes *et al.* to achieve a competitive ratio of $O(\log n)$ for permanent (*i.e.,* infinite holding time) virtual circuits, where n is the number

of nodes in the network [13]. It is extended to the case of finite holding time circuits in [15].

The node selection and placement problem is studied in the context of web server replication and access network design [17, 42, 73]. Shi *et al.* investigate the server placement problem in overlay networks to ensure desired service quality to all its customers [75]. The VN assignment problem considered in this thesis also includes selecting substrate nodes for VN requests. However, the node selection is only a part of the problem and our goal is to map the VN topology to the substrate topology to efficiently use network resources.

2.3 Automatic overlay network configuration in PlanetLab

Our overlay network configuration tool, NetFinder, is motivated by the findings in [54], which examines PlanetLab resource utilization data from the perspective of the designer of a resource discovery system. The authors find that, for some resources, the quantity available at a fixed time differs significantly across nodes, suggesting a potential benefit to using a resource discovery system to wisely place application instances.

NetFinder is based on two technical foundations: end-to-end bandwidth/throughput measurement results and overlay network assignment algorithms. Lee *et al.* assess the capabilities of several existing bandwidth measurement tools and describe the difficulties in choosing suitable tools as well as using them on PlanetLab [45]. Specifically, the authors report that pathchar [37], pchar [50], pathChirp [66] do not work with the current PlanetLab. bprobe, cprobe [20] and SProbe [70] can partially run on PlanetLab.

There are a number of existing tools for node selection in PlanetLab. SWORD is a scalable resource discovery tool for wide-area distributed systems [80]. The particular type of resource that SWORD is intended to discover is the set of nodes on which to deploy a service. However, SWORD focuses on selecting the set of nodes that satisfy user requirement without considering the effects of the selection on the overall PlanetLab performance. Furthermore, SWORD has little support on the inter-node performance and currently does not support available-bandwidth performance between nodes.

Our work is based a number of PlanetLab performance monitoring services. Specifically, CoMon provides a monitoring statistics for PlanetLab at both a node level and a slice

level [23]. The data gathered by CoMon can be directly accessed through CoMon daemon on port 3121 at each PlanetLab node. Our link performance results are currently obtained from the S^3 project [89], which periodically provides end-to-end latency, bottleneck bandwidth capacity, and end-to-end available bandwidth between 400+ PlanetLab nodes. We should note that the contribution of this work is the NetFinder tool that configures the user-defined overlay network in a efficient and on-demand manner. Instead of developing new measurement schemes for PlanetLab, we rely on existing services to collect measurement data. Although its current version uses measurement results form CoMon and S^3, NetFinder is not restricted by these services and can easily adopt new monitoring services and measurement results when they are available.

The overlay assignment problem is challenge since we are trying to optimize both the node performance and link performance. The algorithm used in NetFinder is based on results in Chapter 4, which provides theoretical algorithms for virtual network assignment aiming at reducing the stress among substrate nodes and links. However, the abstract definition of stress can not be easily mapped to realistic network performance metrics such as the CPU usage and available bandwidth. Therefore, modifications and extensions need to be made for the algorithm to work in our scenario.

The output of NetFinder could be used directly as IIAS inputs [36]. Which in turn would start an overlay data plane in PlanetLab following the calculated results.

2.4 Multihomed overlay network

Multihoming has traditionally been employed by stub networks to enhance the reliability of their network connectivity. With the advent of commercial intelligent route control products [26, 53, 61], stubs now leverage multihoming to improve performance. The performance of multihoming route control to improve end-to-end performance and resilience is studied in [6]. The authors find that multihoming can improve performance significantly and that not choosing the right set of providers could result in a performance penalty as high as 40%. Wang *et al.* propose algorithms for selecting a set of upstream ISPs with a monetary cost minimization objective [85]. Intelligent route control products use the idea

of dynamic switching among upstream ISPs to select the best path for any prefix in real time [29]. Dhamdhere and Dovrolis propose methodologies for the provisioning of egress routing at a multihomed content provider, taking into account monetary cost, interdomain level performance and path diversity [24].

Detour [71] and RON [9] demonstrate that using overlay routing to bypass BGPs policy-driven routing enables quicker reaction to failures and improved end-to-end performance. More recently, a comparison between overlay networks and multihoming in terms of round-trip latency, TCP connection throughput, and path availability is reported in [6,7], suggesting that multihoming may be capable to offer almost the same performance benefits with overlay networks, but in a much simpler and more cost-effective way.

The overlay network design problem is the subject of several recent studies. Han *et al.* propose algorithms for both ISP selection and overlay node selection within each ISP, aiming at maximizing path independence without degrading performance [32]. Although multiple ISPs are involved, the paper dose not consider the scenario where an overlay node is multi-homed to several ISPs. An overlay network construction strategy based on a binning scheme to partittion nodes based on network latency is presented in [64]. Lao *et al.* study the overlay node placement and link selection for multicast service overlay network (MSON) [44]. Their node selection objective is to minimize the delay from all users to the proxies. More recently, Cha *et al.* address the overlay node placement problem in a single domain to minimize the overlaps between the default and overlay paths [21].

Other than overlay network design, the node selection problem is also studied in the context of content delivery network (CDN), web proxy and cache, where a set of nodes in the Internet need to be selected to replicate popular content or to serve the users. Qiu *et al.* study the online problem of placing Web server replicas in CDNs to minimize the cost for clients to access data [59]. Cahill and Sreenan propose replica placement algorithms to minimize resource costs [19]. Li *et al.* investigate the optimal placement policy of web proxies for a target web server in the Internet to minimize the overall latency [46].

Our work differs from all existing work in two major aspects. First, we consider a more generalized model of overlay network where each overlay node is multi-homed to several

ISPs. Therefore, the MON network design problem is more challenging since it covers both node selection and ISP selection. Second, most existing work on overlay design/node selection focus on the performance issues such delay, resilience, path diversity. In contrast,while improving the performance for OSP customers, we also take the business relationship between the OSP, customers and competing ISPs into consideration to maximize the OSP profits.

Recently, Andersen *et al.* propose the MONET (Multi-homed Overlay Network) system to use a combination of link multi-homing and a cooperative overlay network of peer proxies to obtain a diverse collection of paths between clients and Web sites [10]. Our MON network shares the similarities with the MONET (Multi-homed Overlay Network) system such that both are exploring the combinational advantage of multi-homing and overlay network to provide better services. However, they assume the existence of overlay nodes and the multi-home connectivity.

CHAPTER III

DYNAMIC OVERLAY ROUTING BASED ON AVAILABLE BANDWIDTH ESTIMATION

Dynamic overlay routing has been proposed as a way to enhance the reliability and performance of IP networks. The major premise is that overlay routing can bypass congestion, transient outages, or suboptimal paths, by forwarding traffic through one or more intermediate overlay nodes. In this chapter, we perform an extensive simulation study the performance of dynamic overlay routing and investigate the applicability of available bandwidth (avail-bw) estimation in dynamic overlay routing.

In particular, we leverage recent work on avail-bw estimation, and focus on overlay routing that selects paths based on avail-bw measurements between adjacent overlay nodes. First, we compare two overlay routing algorithms, reactive and proactive, with shortest-path native routing. We show that reactive routing has significant benefits in terms of throughput and path stability, while proactive routing is better in providing flows with a larger safety margin ("headroom"), and propose a hybrid routing scheme that combines the best features of the previous two algorithms. We then examine the effect of several factors, including network load, traffic variability, link-state staleness, number of overlay hops, measurement errors, and native sharing effects. Some of our results are rather surprising. For instance, we show that a significant measurement error, even up to 100% of the actual avail-bw value, has a negligible impact on the efficiency of overlay routing.

3.1 Dynamic overlay routing

3.1.1 Overlay routing model

We consider two layers of network infrastructure: the native network and a virtual overlay network. The native network includes end-systems, routers, links, and the associated routing functionality, and it provides best-effort datagram delivery between its nodes. The

overlay network is formed by a subset of the native layer nodes (routers and/or end-systems) interconnected through overlay links to provide enhanced services. Overlay links are virtual in the sense that they are IP tunnels over the native network, *i.e.,* overlay packets are encapsulated in IP datagrams and sent from one overlay node to another through the native network. Figure 5 shows an example of an overlay network constructed over a native network. Note that since overlay links are virtual, the overlay network topology can be a full mesh allowing maximum flexibility in choosing overlay routes.[1]

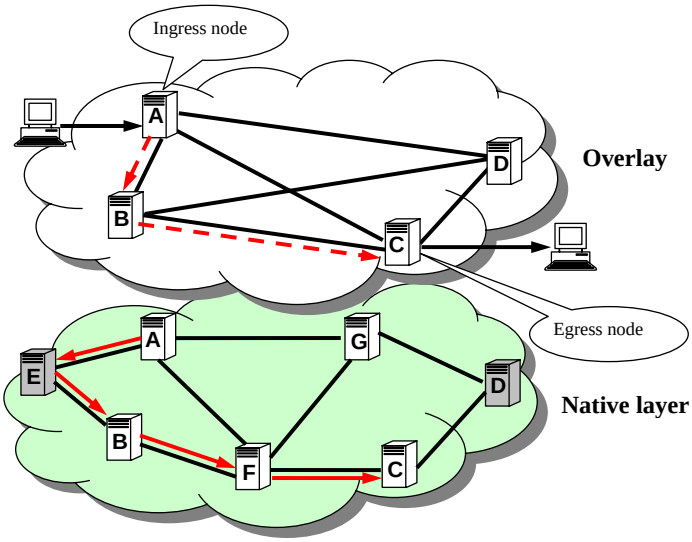


Figure 5: Overlay and native network layers.

An important service that overlay networks provide is *dynamic path selection* based on specified performance objectives. The performance of a path can be a function of the delay, loss rate, and/or avail-bw in the path, among other metrics. Additionally, different traffic classes can be associated with a different path performance metric. An overlay flow arrives at an *ingress node*, destined to a certain *egress node*. Upon the flow's arrival, the ingress node determines the best overlay path to the egress node based, ideally, on the current state and performance of the overlay links (referred to as *overlay link-state*). The chosen overlay path information is then included in the header of each packet (*source routing*), and the packet is forwarded to the corresponding sequence of overlay nodes. To provide resilience

[1]Overlay networks with hundreds of nodes may require a sparser connectivity, or some form of hierarchical routing, to deal with scalability problems in the link-state measurement and dissemination process.

to network failures and load variations, the ingress node of an active overlay flow checks for a better path at the end of every *path update period* P_u, during the lifetime of the flow. If a better path is found, the flow can be switched to that path. The previous path update events and the corresponding time scales are illustrated in Figure 6.

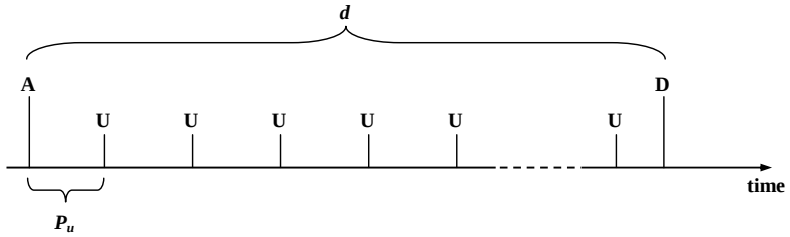


Figure 6: Overlay flow events and the related time scales. A, U and D are the flow arrival, path update, and flow departure events, respectively. d is the flow duration and P_u is the path update period.

To perform dynamic path selection, the overlay nodes need to perform *link-state measurement* and *link-state dissemination*. The overlay link-state is the input to the overlay routing algorithm. The state of an overlay link can be represented by a collection of performance metrics, such as delay, loss rate, availability, or capacity. In this work, we focus exclusively on avail-bw, leveraging recent advances in relate [8, 34, 38, 67, 78]. Of course it is possible to further limit the path selection algorithms with additional constraints on the path delay or loss rate, for example.

The avail-bw, also known as residual capacity, of a native link is defined as the capacity of the link minus its average traffic load. The avail-bw of an overlay link (or native path), on the other hand, is the minimum avail-bw among all native links that comprise that overlay link (or native path). Unlike the avail-bw of a native link, which can be easily measured passively by the corresponding router, the avail-bw of overlay links cannot be estimated passively by overlay nodes. Instead, the avail-bw of an overlay link has to be measured through active end-to-end probing techniques performed by the overlay nodes. Recent developments in end-to-end avail-bw estimation provided us with tools and techniques that can estimate the avail-bw of a network path. These techniques are based on special probing packet streams that can identify in a non-intrusive way the maximum rate that will not cause congestion in a path. The latency of the existing measurement techniques varies

from a few tens of milliseconds to tens of seconds, depending on whether the tools run continuously in the background or whether they run in a "measure-once-and-terminate" mode. Their accuracy depends on the traffic burstiness and the number of bottleneck links in the path, and relative measurement errors in the range of 10-30% should be expected [38].

Each overlay node measures the avail-bw of the paths to its adjacent overlay nodes. Periodically, the link-state information that is generated from these measurements is disseminated to all other overlay nodes. The link-state database of an overlay node is refreshed upon receiving new link-state information. Note that the link-state measurement and dissemination are performed independent of any flow-related events. There are three important time scales involved in the avail-bw measurement and dissemination process: the *measurement delay* (D_m), the *link-state refresh period* (P_r) and the *dissemination delay* (D_d). The measurement delay D_m is the time needed to generate a new avail-bw estimate. The link-state refresh period P_r (or simply, refresh period) is the time interval between consecutive updates of the local avail-bw link state. Note that P_r cannot be less than D_m, but it could be larger to reduce the link-state dissemination overhead. The end of a link-state refresh interval is determined by the end of the last measurement period. The dissemination delay D_d^{ij} refers to the time needed for the new link-state generated by the i'th overlay node to reach the j'th overlay node. We assume that D_m and P_r are constant, while D_d^{ij} varies randomly for each pair (i, j) of overlay nodes. The overlay link-state measurement and dissemination events and time scales are shown in Figure 7.

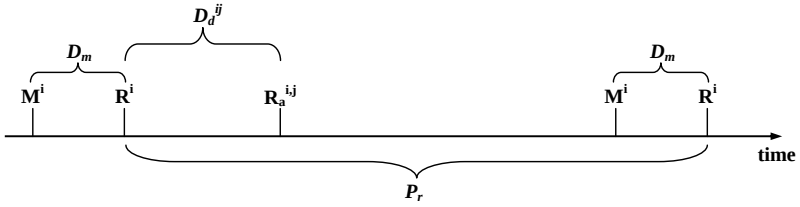


Figure 7: Time scales for the measurement and dissemination of overlay link-state at the i'th overlay node. M^i represents the start of an avail-bw measurement for all the egress overlay links of the i'th overlay node. R^i is a link-state refresh event, and it takes place at the end of the last avail-bw measurement. $R_a^{i,j}$ represents the arrival of the new link-state from overlay node i to node j.

3.1.2 Overlay routing algorithms

We model the overlay topology as a directed graph $G = (V, L)$ whose vertices and links represent the set of overlay nodes and overlay links, respectively. The avail-bw of each overlay link $l = (u, v) \in L$ is denoted by $b(l)$. An overlay path p is a sequence of one or more overlay links and its avail-bw $b(p)$ is defined as $b(p) = \min_{l \in p} b(l)$.

We use the overlay flow as the basic traffic unit for overlay routing, meaning that all packets of a flow are sent via the same path determined for that flow. Each overlay flow is modeled by four parameters $f = (v_i, v_e, d, r)$; $v_i, v_e \in V$ are the ingress and egress overlay nodes of the flow, and d is the flow duration. The last parameter r is the flow's *maximum throughput limit (max-rate limit)*, and it represents the maximum throughput that the flow can achieve. For instance, the throughput of a flow may be limited by its ingress or egress access capacity, the throughput of a streaming flow may be limited by the rate of the best-quality encoder, and the throughput of a TCP flow may be limited by the size of end-host socket buffers. Due to limited network resources, a flow's actual throughput can be lower than its max-rate-limit r. We therefore use the symbol a to represent the current value of the *achieved throughput* of a flow ($a \leq r$).

When we compare the path that a flow is currently routed on with another path we need to take into account the load that the flow already imposes on the former. To do so, we introduce another metric referred to as *headroom*. For a flow f, the headroom $h(f, l)$ at an overlay link l is defined as

$$h(f,l) = \begin{cases} b(l) + a & \text{if } f \text{ is routed on } l \\ b(l) & \text{otherwise} \end{cases} \tag{1}$$

Similar to avail-bw, the headroom of a path can be defined as the minimum headroom among all links along that path, *i.e.,* for an overlay path p,

$$h(f,p) = \min_{l \in p} h(f,l) \tag{2}$$

Note that the headroom $h(f, p)$ of path p is equal to the avail-bw $b(p)$ if flow f is *not* routed on p; otherwise, the headroom is larger than the avail-bw by the flow's achieved throughput a.

In this work, we first consider two overlay path selection schemes: *proactive overlay routing* and *reactive overlay routing*. In both schemes, a flow will be initially routed on the path that provides the maximum headroom. With the proactive algorithm, the flow is switched to the path that appears to have the maximum headroom at the end of each path update period (see Figure 2). Note that due to potential staleness in the link-state information, that path may not actually be the best choice. With the reactive algorithm, on the other hand, the flow stays at its current path if it has achieved its max-rate limit r ("satisfied flow"). Otherwise, the flow is "unsatisfied" and it is routed on the path with the maximum headroom; that path may be the same with the previously used path.

The intuition behind proactive routing is that the maximum headroom path can provide a flow with *a wider safety margin* to avoid transient congestion due to traffic load variations, measurement errors, and stale link-state. The intuition behind reactive routing is that a flow should stay at its current path if it is already satisfied, leading to fewer path changes and *more stable overlay routing*.

The path selection algorithm for the proactive and reactive schemes is based on the shortest-widest routing algorithm of [86]. The pseudo-code for both reactive and proactive overlay routing is given in Algorithm 1. Even though the algorithmic difference between the two routing schemes is minor, Section 4 shows that it can result in very different performance.

Algorithm 1 Proactive overlay routing

INPUTS:
  $f = (v_i, v_e, d, r)$: overlay flow under consideration;
  $P = \{p_i\}$: set of alternative paths from v_i to v_e;
  a: achieved throughput of f (zero for new flow);
OUTPUTS:
  Selected path p';

if ((Proactive-Routing) OR (Reactive-Routing AND $a < r$)) **then**
  Update headroom $h(f, p)$ for all $p \in P$;
  $p' = argmax_{p_i \in P} h(f, p_i)$;
  Route f on path p';
end if

3.2 Simulation model and performance metrics

3.2.1 Simulation model

We have implemented a flow-level discrete-event simulator for dynamic overlay routing. The native network topology is based on the core US topology of four large ISPs (Sprint, ATT, Level3 and Verio), estimated from the measurements of the Rocketfuel project [77] (see Figure 8). These four ISPs are tier-1 providers and so they are interconnected in a full mesh, with three inter-ISP connections per ISP pair. The inter-ISP links connect routers that are located in the same city. We assume that the native-layer routes are based on the shortest path algorithm, and that they do not change with time (at least for the time scales of overlay routing that we are interested in).

The overlay network consists of 18 overlay nodes located in major US cities. Each overlay node is connected with an overlay access link to one of the four ISPs at the corresponding router that is located in the same city. The overlay nodes are interconnected in a full-mesh topology.

There are three types of native links: intra-ISP links, inter-ISP links and overlay access links. In our simulation, the capacity of these three link types is uniformly distributed in the range of $[500, 1500]$, $[400, 600]$ and $[8000, 12000]$, respectively. Note that the most likely bottlenecks are the inter-ISP links, while the overlay access links are the least likely bottlenecks.

Overlay flows are generated according to a Poisson process with average arrival rate F_a.[2] The flow duration is exponentially distributed with mean F_d. The selection of the source and destination nodes for the overlay flows follows a randomly generated (non-uniform) traffic matrix. The flow max-rate limit follows an exponential distribution with mean F_r. The simulator does not capture bandwidth sharing in saturated links or congestion control by overlay flows. Consequently, if a new flow arrives at a saturated link, then the new flow will obtain zero throughput while the existing flows will maintain their previous throughput. The subtle interactions between congestion control and dynamic overlay routing are outside

[2]The Poisson flow arrival model is reasonable, as long as there are no correlations on bursts in the overlay flow arrival process. The Poisson model has been previously validated for application session arrivals [55].

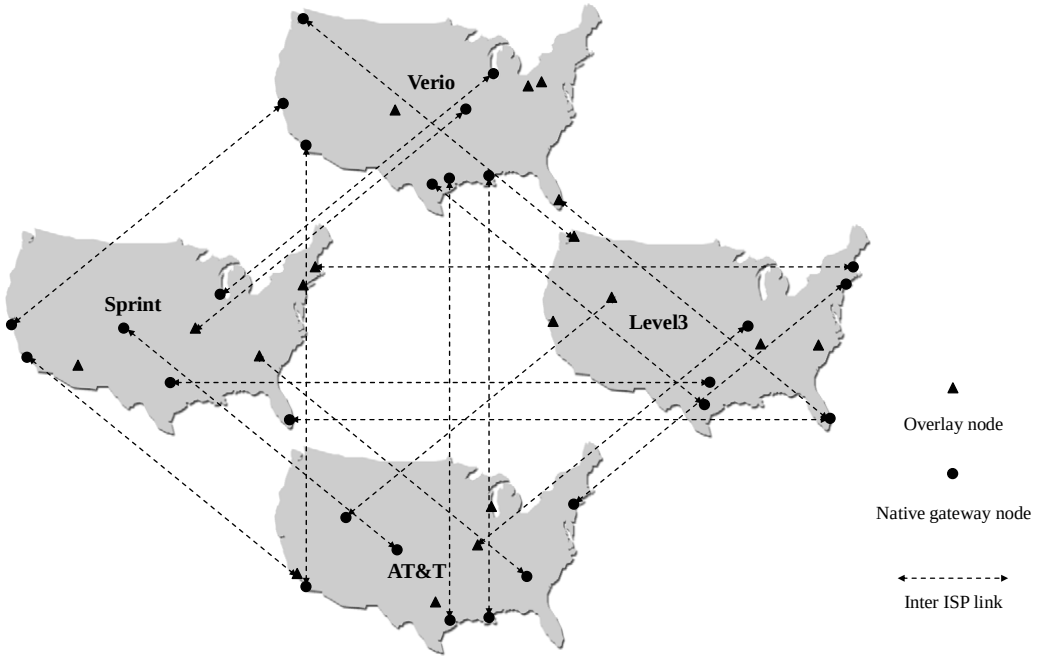


Figure 8: A sketch of the native network topology (also showing the location of the overlay nodes).

the scope of this work.

We also simulate some non-overlay traffic, referred to as *cross traffic*. The cross traffic causes random load fluctuations in the native network. Specifically, the cross traffic at each native link is modeled as a fluid process with variable rate. The rate change events take place based on a Poisson process, independent of the rate changes at other links. The average time period between rate variations is F_c. The rate of the cross traffic after a rate change event is chosen randomly as $\min(b, x \cdot C)$, where b is the avail-bw of the link, x is uniformly distributed in $[0, 1]$, and C is the link capacity. Since the cross traffic rate is at most b, this traffic can cause load variations but not congestion.

Table 1 shows the set of important parameters and their default values in our simulation study.

Each simulation result is obtained by running the simulator until 30,000 overlay flows have been serviced. Furthermore, to avoid the effect of transient simulation effects, we start to collect data after the first 10,000 overlay flows.

Table 1: Major simulation parameters and their default values

Overlay flow and cross traffic parameters		
Flow arrival rate	F_a	10.0 flows/sec (average)
Flow duration	F_d	50sec (average)
Max-rate limit	F_r	20Mbps (average)
Cross traffic rate change period	F_c	20sec (average)
Native network parameters		
Number of native nodes		275
Number of native links		1164
Intra-ISP link capacity		$[500, 1500]$Mbps
Inter-ISP link capacity		$[400, 600]$Mbps
Overlay access link capacity		$[8000, 12000]$Mbps
Overlay routing parameters		
Link-state measurement delay	D_m	0.1sec
Link-state refresh period	P_r	0.5sec
Link-state dissemination delay	D_d	$[0, 0.2]$sec
Path update period	P_u	1.0sec

3.2.2 Performance metrics

We evaluate overlay routing based on three important aspects: *efficiency*, *stability*, and *safety margin*. Efficiency refers to the ability of overlay routing to achieve higher throughput than native routing, by avoiding saturated links. Stability refers to the frequency with which overlay flows switch between different paths. The safety margin represents the robustness of overlay routing in the presence of cross traffic fluctuations, measurement errors, and stale link-state information.

Specifically, to quantify the efficiency of a routing scheme we use the *normalized average throughput* T. This is defined as the total amount of data sent by completed overlay flows, normalized by the amount of data that would have been sent if each of these flows had achieved its max-rate limit,

$$T = \frac{\sum_{i=1}^{k} \int a_i(t)dt}{\sum_{i=1}^{k} r_i \cdot d_i} \leq 1 \tag{3}$$

where k is the number of completed flows, a_i and r_i are the achieved throughput and the max-rate limit of the i'th flow, respectively, and d_i is the duration of the i'th flow. Notice that, given the limited network capacity resources, it may be infeasible to have T=100% for a given overlay load. Consequently, under the same offered load, a higher value of T reflects a more efficient overlay routing scheme.

To quantify the stability of a routing scheme we use the *path switching ratio* S. Suppose that an overlay flow i experienced u_i path update events during its lifetime, and that c_i among these updates were path changes. The ratio $\frac{c_i}{u_i} \in [0,1]$ reflects the relative frequency with which flow i switched between paths: if it is one the flow switched paths with every path update, while if it is zero the flow never switched paths. The *path switching ratio* S is the weighted average of the previous ratio across all completed overlay flows, with weights proportional to the flow durations,

$$S = \sum_{i=1}^{k} \left(\frac{u_i}{\sum_{j=1}^{k} u_j} \cdot \frac{c_i}{u_i} \right) = \frac{\sum_{i=1}^{k} c_i}{\sum_{i=1}^{k} u_i} \tag{4}$$

A higher value of S indicates that flows switch paths more frequently and so the network is less stable.

To quantify the safety margin of a routing scheme we use the *normalized average headroom* H. As we did for normalized throughput, we normalize the headroom of each flow by its max-rate limit. Instead of measuring the headroom of a flow as a continuous function of time however, we use Poisson sampling to estimate the time-average of the normalized per-flow headroom. Consider the j'th overlay flow at a sampling instant i, and let h_{ij} and r_{ij} be the headroom and max-rate limit of that flow, respectively. The flow's relative headroom is h_{ij}/r_{ij}. The weighted average of the relative headroom of all active flows at the i'th sampling instant, weighted by the max-rate limit of each flow, is

$$H_i = \sum_j \frac{r_{ij}}{\sum_{j'} r_{ij'}} \cdot \frac{h_{ij}}{r_{ij}} = \frac{\sum_j h_{ij}}{\sum_j r_{ij}}$$

Taking the corresponding weighted average across all sampling instants i, we get that the *normalized average headroom* is

$$H = \sum_i \frac{\sum_j r_{ij}}{\sum_{i'} \sum_{j'} r_{i'j'}} \cdot H_i = \frac{\sum_i \sum_j h_{ij}}{\sum_i \sum_j r_{ij}} \tag{5}$$

Note that H, as opposed to T, can be larger than 100%. In the simulations of Section 4, the average sampling period for the calculation of H is 0.5 seconds.

3.3 Simulation study

In this section, we first evaluate and compare the efficiency, stability, and safety margin of proactive and reactive overlay routing under various network conditions. Based on the

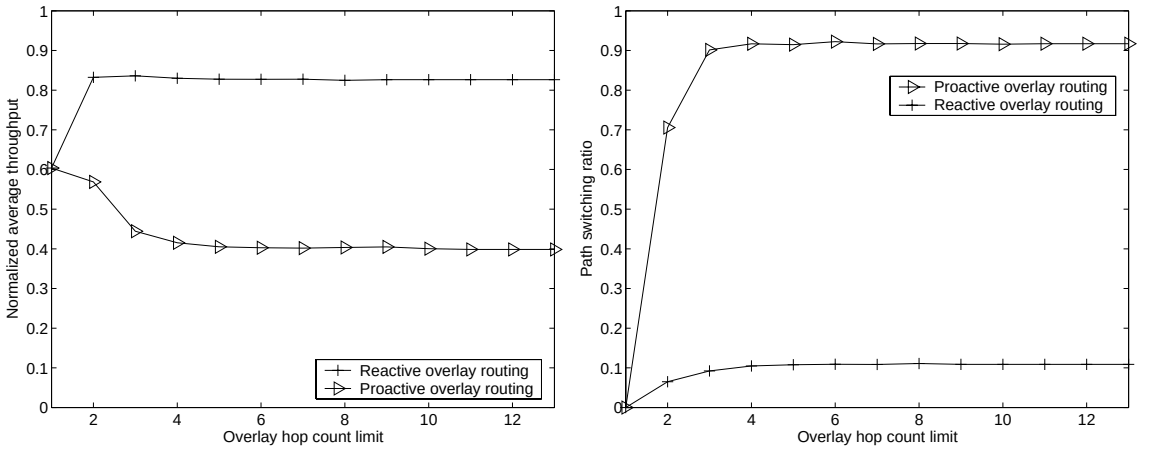
results of this comparison, we propose a hybrid algorithm that combines the best features of reactive and proactive routing. Finally, we examine the effect of several important factors on the performance of the hybrid algorithm.

3.3.1 Maximum overlay hop count

A major advantage of overlay routing is its ability to utilize several alternate paths instead of the single path that is provided by IP routing. The number of such alternate paths increases with the number of overlay nodes an end-to-end path can traverse. We refer to the *overlay hop count* as the number of hops (or overlay links) that an end-to-end path traverses. In practice, the overlay hop count would be bounded by a maximum value H_{max}. The practical necessity for this limit is related to source routing: the intermediate overlay nodes need to be encoded in the header of each packet, and there is a limited number of bits for doing so. H_{max}=1 means that the overlay path is the same with the native-layer path, while H_{max}=2 means that the overlay path can traverse at most one intermediate overlay node.

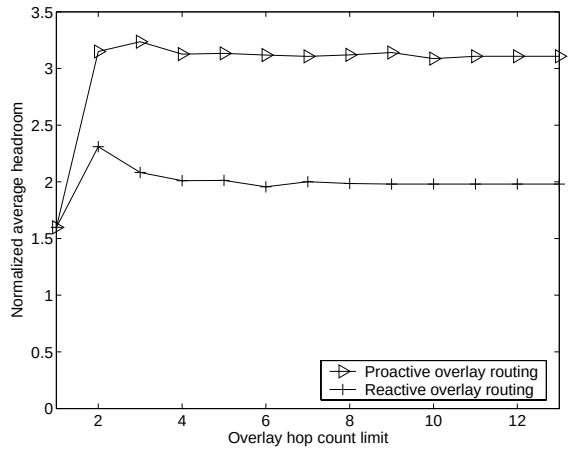
In this simulation, we increase the maximum overlay hop count H_{max} from 1 to 13, and compare the performance of reactive and proactive overlay routing. The performance of native routing is also shown, as $H_{max} = 1$. Figure 9(a) shows that the average throughput T of reactive routing improves significantly when we increase H_{max} from one to two hops. The increase for larger values of H_{max} is negligible, meaning that longer overlay paths are rarely needed to avoid congestion. This shows that using a single intermediate overlay node with reactive routing is sufficient to obtain most throughput gain compared to native routing, and that this gain can be substantial. On the other hand, proactive routing performs worse as we increase H_{max}. One reason for this behavior is shown in Figure 9(d), which shows the *average native hop count* as a function of H_{max}.[3] As we would expect, the chosen paths in the native network tend to be longer as we increase H_{max}. Also, the paths used by proactive routing are significantly longer than the paths used by reactive routing, because the former always attempts to choose the path with the maximum headroom. As a result,

[3]Another major reason is given in Section 4.2.

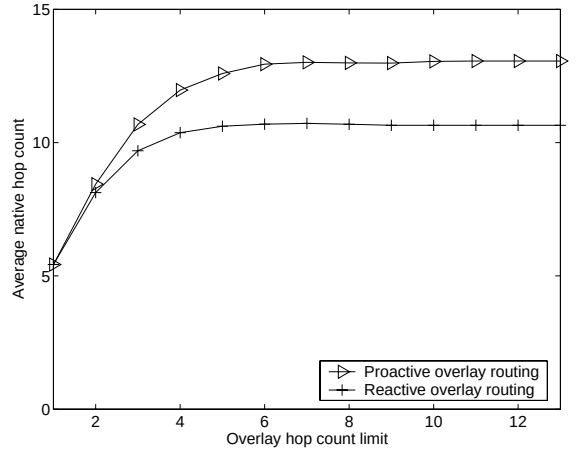


(a) Efficiency

(b) Stability



(c) Safety margin



(d) Native path length

Figure 9: Effect of maximum overlay hop count H_{max}.

the proactive algorithm uses more network resources than the reactive algorithm, decreasing the network's avail-bw and causing a lower value of T.

In terms of stability, increasing H_{max} causes the following two effects: 1) more alternate paths are considered by each flow and so there is a higher frequency of path switching, and 2) more native links are affected by the previous path changes, causing further variations in the avail-bw distribution and triggering even more path switching. Indeed, as Figure 9(b) shows, a higher value of H_{max} causes more frequent path switching. Note that proactive routing experiences significant instability, while reactive routing maintains a low path switching ratio across the range of of H_{max}.

Although proactive routing performs worse than reactive in terms of efficiency and stability, it does have the advantage of providing overlay flows with a higher average headroom, as shown in Figure 9(c). The increased headroom can act as a wider safety margin in the presence of traffic fluctuations and measurement errors. Note that the maximum headroom is obtained (by both algorithms) when H_{max}=2; longer overlay paths can cause larger consumption of network capacity.

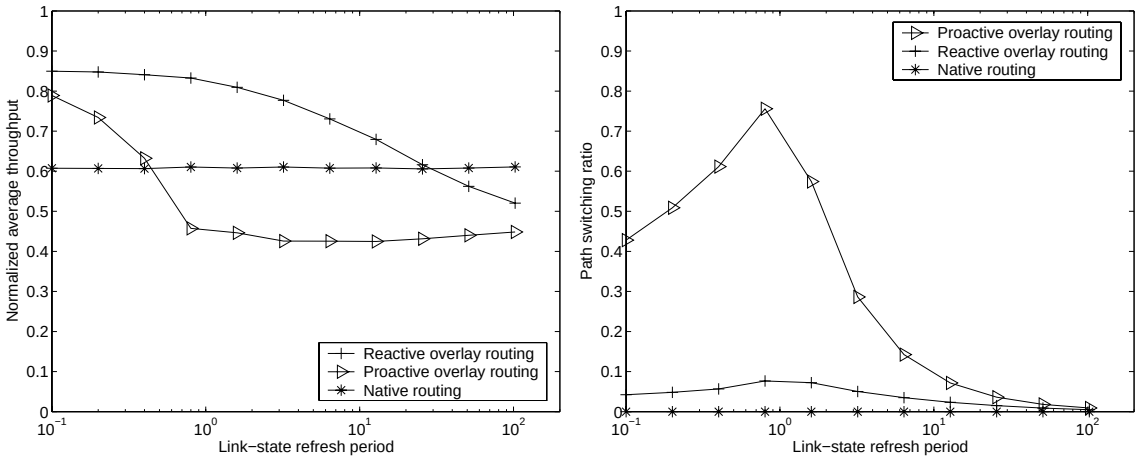
The previous results show that with at most one intermediate overlay node, reactive overlay routing can achieve significantly improved efficiency and headroom over native routing and maintain good stability. For proactive routing, limiting the maximum overlay hop count to two is even more critical in terms of efficiency and stability. Consequently, in the rest of this chapter we will set H_{max}=2 for both algorithms. The practical implication of this limit is that a single node identifier in the packet header would be enough to encode the overlay route.

3.3.2 Link-state refresh period

Recall that the link-state refresh period P_r is the time length between successive updates of the overlay avail-bw link-state. A higher value of P_r increases the staleness of overlay routing information, but also decreases the link-state dissemination overhead.

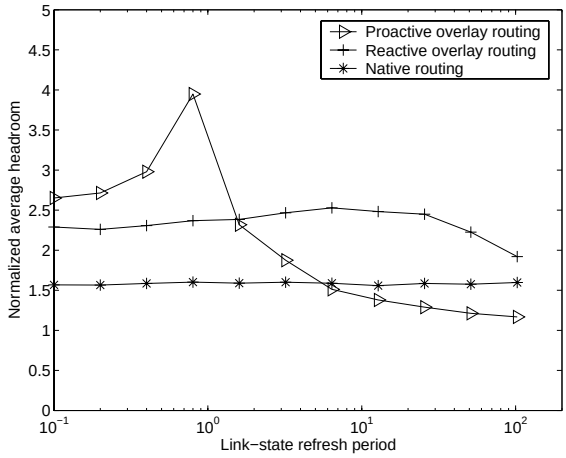
Figure 10 shows the performance of proactive, reactive, and native routing as we vary P_r from 100msec to 100sec. Note that P_r cannot be lower than the measurement delay D_m, which is set to 100msec in our simulations. Even though P_r would not be more than a few seconds in practice, we simulate a wider range for illustration purposes.

In terms of average throughput, Figure 10(a) shows that, as we would expect, the efficiency of both reactive and proactive routing drops as P_r increases. Interestingly, however, the reactive algorithm is much more robust to stale link-state than the proactive algorithm. The former can achieve better throughput than native routing as long as P_r is less than about 10 seconds, while the latter does worse than native routing if P_r exceeds 400msec. The reason for this major difference between reactive and proactive routing is that the latter relies much more heavily on avail-bw information, because it considers switching



(a) Efficiency

(b) Stability



(c) Headroom

Figure 10: Effect of link-state refresh period P_r.

even the satisfied flows. Consequently, a higher value of P_r, with its increased link-state staleness, causes a much more dramatic throughput loss for the proactive algorithm. The corresponding throughput loss for the reactive algorithm is negligible when P_r is between 100ms-1000ms, which is probably a reasonable range for P_r in terms of dissemination overhead.

In terms of stability, Figure 10(b) shows that the path switching ratio curves can be split into two regions. S increases as P_r approaches one second, which is the flow update period P_u. Then, for longer values of P_r, S decreases. The reason for this behavior is as follows. As P_r increases from 100ms to 1sec, the link-state becomes increasingly more stale. This increasing staleness means that a higher fraction of the path switching decisions were based

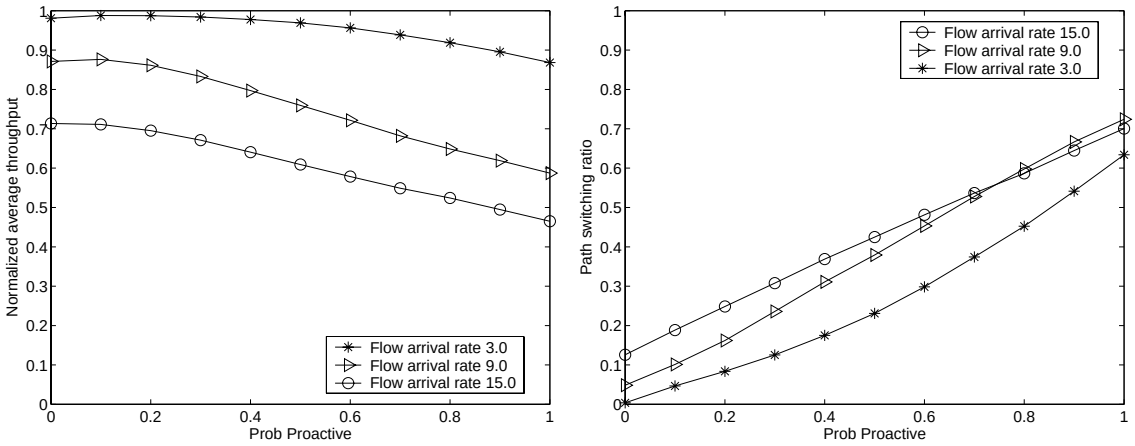
on inaccurate information and they were incorrect (meaning that the flow did not move to the maximum headroom path or it did not avoid congestion). Consequently, further path switching is then required to correct the previous routing decisions. On the other hand, when P_r becomes larger than P_u (one second) the link-state information changes less often than the frequency with which overlay flows examine whether to switch paths. Consequently, many of the path update events see the same link-state information, and so the corresponding flows decide that they should stay at the same path. This tends to decrease the metric S as P_r increases.

3.3.3 Hybrid routing and probability of proactive switching

The previous simulation results showed that the proactive algorithm performs worse than the reactive algorithm in terms of throughput and stability. Furthermore, those results showed two major reasons for the difference between the two algorithms: first, proactive routing tends to use longer native paths and thus consumes more network resources (especially when $H_{max} > 2$), and second, proactive routing is much more sensitive to stale link-state information. On the other hand, proactive routing performs better than reactive in terms of average headroom (when P_r is less than P_u), providing overlay flows with a wider safety margin.

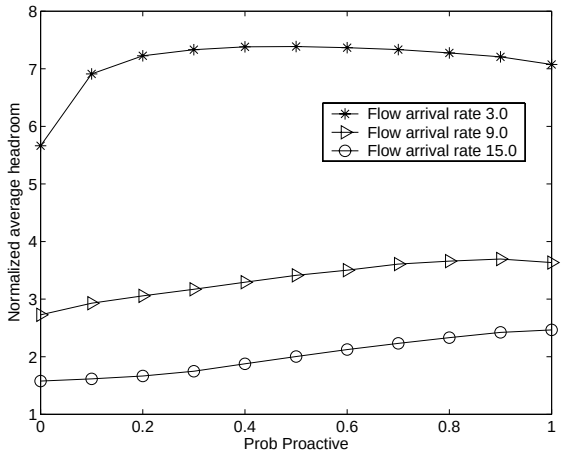
Given the previous trade-off, we propose a simple heuristic that combines the previous two algorithms in a probabilistic manner. We refer to this algorithm as *hybrid routing*. At each path update event, the hybrid algorithm performs proactive path switching with a probability p_p (referred to as *probability of proactive switching*); otherwise, the algorithm performs reactive path switching. The intuition behind this algorithm is to maintain the good throughput and stability properties of reactive routing (*i.e.,* to use a low p_p), but at the same time to occasionally switch the flow at a path with higher headroom, even if it is satisfied, in order to improve its safety margin.

Figure 11 shows the performance of hybrid routing for different values of p_p. The three curves in each graph represent cases of low (F_a=3.0), medium (F_a=9.0), and high (F_a=15.0) overlay traffic load. Note that p_p=0 corresponds to reactive routing and p_p=1 corresponds



(a) Efficiency

(b) Stability



(c) Safety margin

Figure 11: Effect of probability of proactive switching p_p.

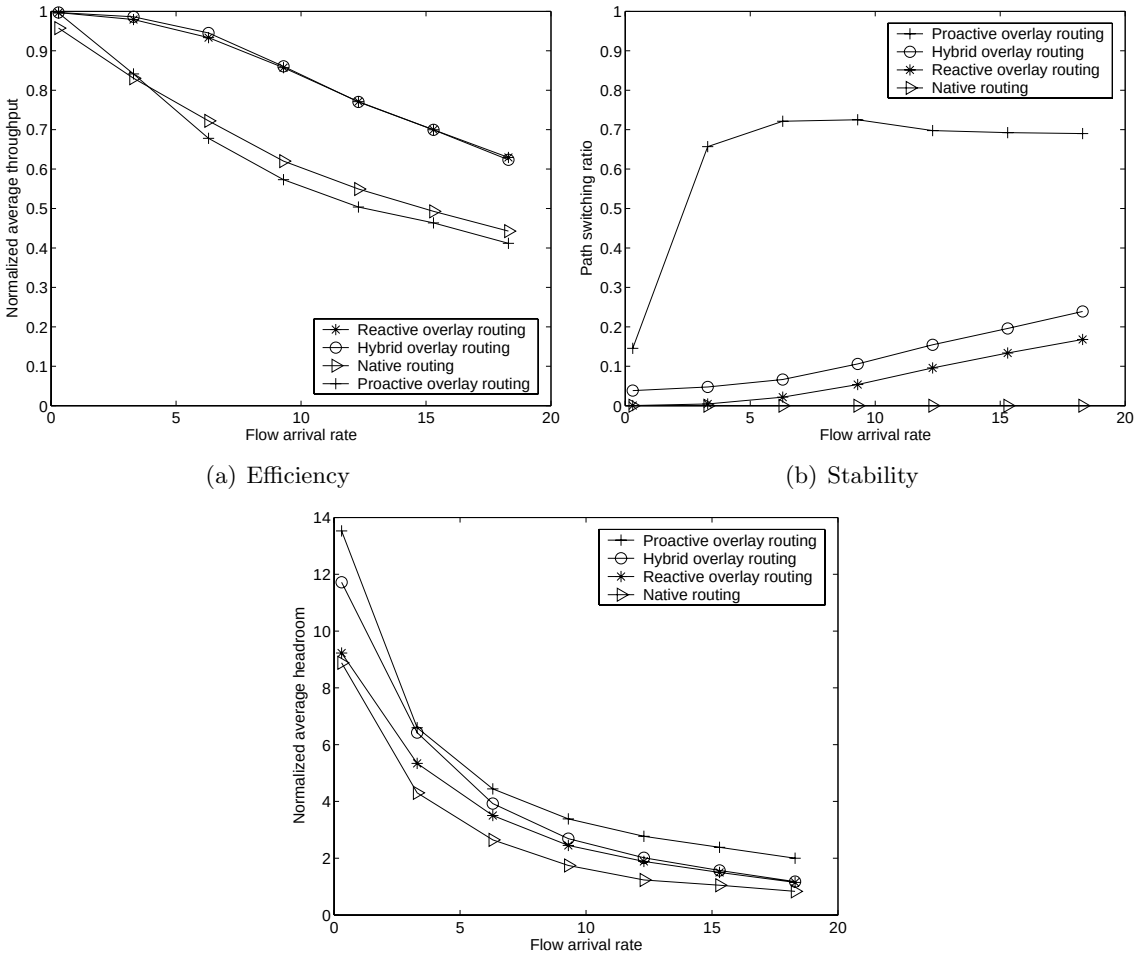
to proactive routing. In terms of efficiency, p_p=0.1 is almost as good as pure reactive routing. In terms of stability, the smaller p_p the lower S will be. The difference between p_p=0.1 and p_p=1 is substantial, however, and so the former value gives a major stability gain compared to proactive routing. In terms of headroom, on the other hand, the choice of p_p makes a significant difference only in low load conditions; in heavy load conditions the headroom is quite limited even with the proactive algorithm. In low load conditions, p_p=0.1 gives us most of the headroom gain of proactive routing.

To summarize, the simulation results indicate that the probability of proactive switching p_p can be close to 10%, resulting in almost the same efficiency and stability with the reactive algorithm, but also increased headroom especially in lower load conditions. In the rest of

this chapter, we set p_p=0.1.

3.3.4 Traffic load and flow arrival rate

The overlay traffic load is determined by the flow arrival rate, flow duration and max-rate limit. Increasing one of these parameters while keeping the others fixed increases the offered traffic load. In this experiment, we vary the flow arrival rate F_a and compare the performance of proactive, reactive, and hybrid overlay routing, as well as native shortest-path routing, under increasing load. These simulations provide further evidence that the hybrid algorithm combines the best features of reactive and proactive routing.



(a) Efficiency

(b) Stability

(c) Safety margin

Figure 12: Effect of overlay traffic load F_a.

Figure 12(a) shows the normalized throughput for the four routing algorithms. Under very light load, where native links are rarely saturated, both native routing and overlay

routing satisfy almost all flows. In higher loads, reactive routing manages to avoid saturated paths, as long as there is at least one non-congested path. Under heavy-load conditions, it still performs much better than native routing. The hybrid algorithm offers essentially the same throughput as reactive routing. Proactive routing, on the other hand, performs even worse than native routing, mostly due to its high sensitivity on stale link-state information.

In terms of stability, Figure 12(b) shows that the path switching ratio of reactive routing remains almost zero at low traffic loads and increases slowly as more flows become unsatisfied in heavy loads. Proactive routing, on the other hand, introduces significant instability, with about 70% of the flow update events causing path switching, even before the network becomes congested. The path switching ratio with hybrid routing is slightly higher than with reactive routing, but still much lower than proactive routing.

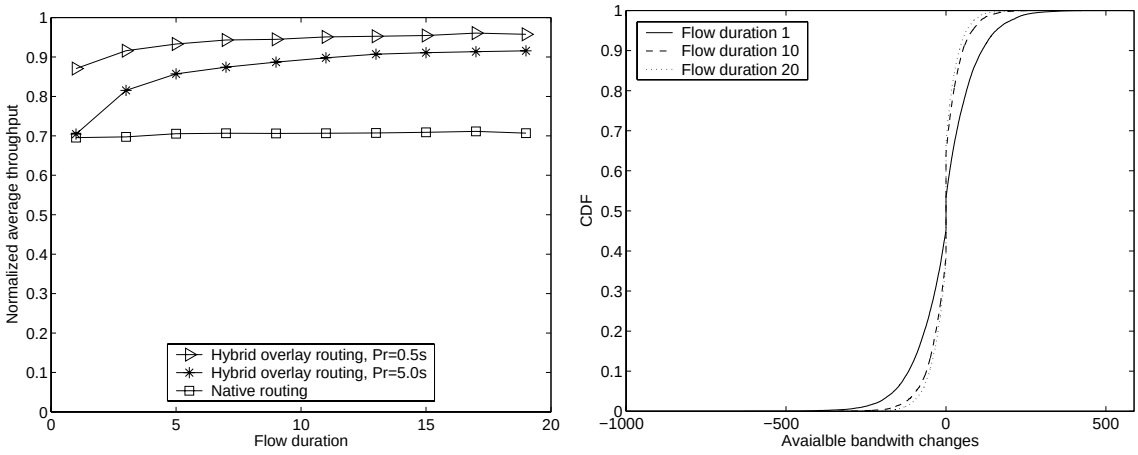
In terms of average headroom, Figure 12(c) shows that proactive routing achieves consistently higher average headroom than both reactive routing and native routing. The hybrid algorithm provides almost the same headroom with proactive routing in lower load conditions, when there is still significant headroom.

3.3.5 Effect of traffic variability

The performance of overlay routing depends on the variability of the underlying traffic, because the best path for the next update period is predicted based on the load of each path at the end of the last update period. Under the same aggregate load, we would expect that higher traffic variability will lead to worse overlay routing performance in terms of all three performance metrics. In this section, we examine two factors that affect the variability of the underlying traffic: the overlay flow duration F_d and the cross traffic variation period F_c.

Let us first focus on the average overlay flow duration F_d. In this simulation, we increase F_d from 1 to 20 seconds and reduce the flow arrival rate F_a proportionally so that the aggregate offered load remains constant. We also remove the non-overlay cross traffic, so that we focus exclusively on the variability introduced by overlay traffic. Figure 13(a) shows the throughput of hybrid routing under two refreshing periods: P_r=0.5sec and P_r=5.0sec;

the latter represents a rather extreme case of stale link-state. For reference, we also show the results with native shortest-path routing.



(a) Efficiency (b) CDF of avail-bw variations during 0.5s intervals

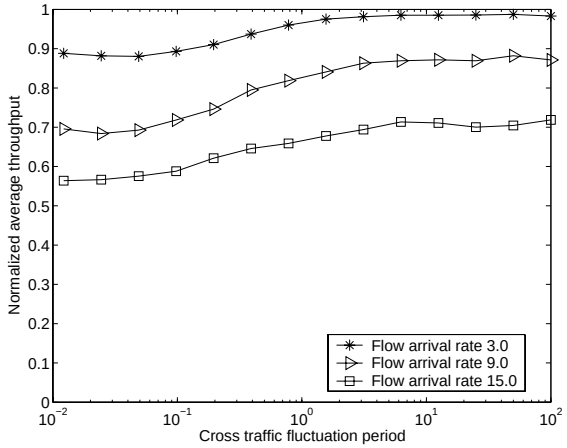
Figure 13: Effect of overlay flow duration F_d.

Since the offered load remains constant, the native routing can achieve the same throughput independent of the flow duration. In contrast, the throughput of hybrid routing increases with F_d. For short flows, more flow arrival/departure events take place within each measurement period, making the overlay traffic more variable and causing greater staleness in the link-state routing information. As flows become longer, the throughput increases because the resulting traffic variability decreases, and so even outdated link-state information is reasonably accurate. To further illustrate this point, Figure 13(b) shows the cumulative distribution functions (CDF) of the avail-bw variations across successive 0.5 second intervals for three different flow durations. Notice that shorter flows cause more significant traffic variability than longer flows.

Another observation from Figure 13(a) is that when the flow duration decreases, hybrid overlay routing needs a shorter refresh period P_r to maintain the same throughput. More generally, a shorter refresh period is required when the traffic variability increases. In selecting P_r, the objective should be that the refresh period is short enough so that the avail-bw does not change significantly during successive link-state updates. Similar observations hold for the path switching ratio and the average headroom (not shown here).

Let us now focus on the variability of the non-overlay cross traffic. As mentioned in

Section 3, the cross traffic has an average rate variation period F_c=20sec. In this simulation experiment, we vary F_c from 0.01sec to 100sec. Obviously a higher value of F_c would make the cross traffic less variable. Figure 14 shows the average throughput with hybrid



(a) Efficiency

Figure 14: Effect of cross traffic rate variation period F_c.

overlay routing under three load conditions. Note that the average throughput is not so sensitive to F_c, as long as the latter is larger than the flow update period P_u (one second). This is because when $P_u < F_c$, the avail-bw of overlay paths does not change significantly between successive path update events. With more frequent cross traffic variations, there is a reduction in the average throughput. The reduction is not major with the hybrid (or the reactive) algorithm however, because the latter is quite robust to stale link-state. Similar observations hold for the path switching ratio and the average headroom (not shown here).

3.3.6 Measurement errors

The literature on avail-bw measurement techniques reports estimation errors that vary from $\pm 10\%$ to $\pm 30\%$ [8, 34, 38, 67, 78]. It seems unlikely at this point that these measurement techniques can be improved so dramatically that the estimation error will become almost zero. Since the overlay routing algorithms that we study rely on avail-bw estimation, we need to examine the effect of avail-bw measurement errors on the performance of overlay routing.

Let us denote the real avail-bw value at an overlay link by v, the measured value by m,

and the error factor by $e \in [0, 1]$. We consider the following three error models:

Relative error: The measurement m is symmetrically and uniformly distributed in a range around v, with the error being proportional to v. That is, $m \in [(1 - e) \cdot v, (1 + e) \cdot v]$

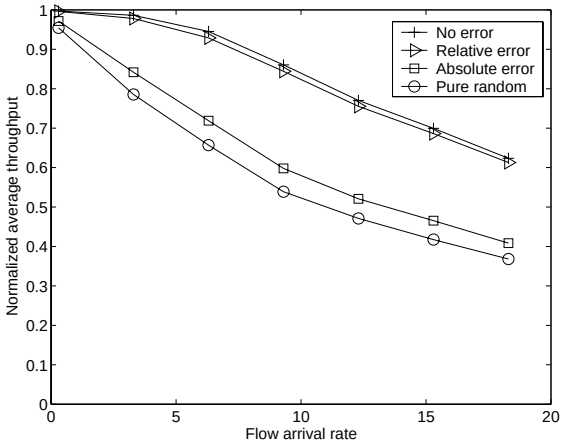
Absolute error: The measurement m is symmetrically and uniformly distributed in a range around v, with the error being proportional to the overlay link capacity C. That is, $m \in [\max(0, v - e \cdot C), v + e \cdot C]$.

Random estimate: The measurement m varies randomly anywhere between zero and the overlay link capacity C. C represents here an upper bound on the avail-bw of that link. In other words, instead of actually measuring avail-bw, the routing algorithms use a random estimate of the avail-bw in each overlay link, limited by the corresponding link capacity.

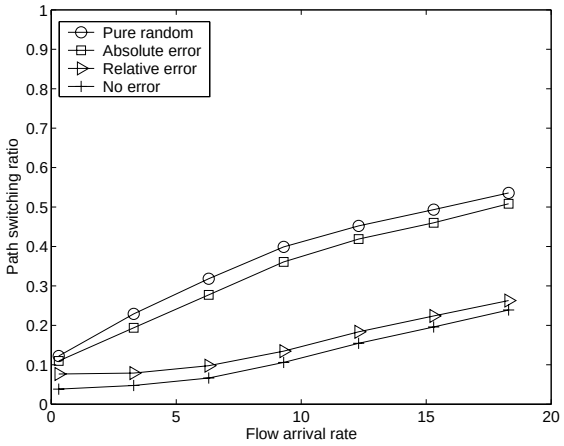
We note that the avail-bw estimation literature reports that the measurement errors typically follow the relative error model. The reason we examine the absolute error and random estimate models is to show that different error models can affect overlay routing in very different ways.

Figure 15(a) compares the throughput of hybrid overlay routing with no error, 100% relative error ($e = 1$), 100% absolute error, and random estimate. Surprisingly, we observe that *relative avail-bw estimation errors, even up to 100%, have very small impact on the performance of overlay routing.* On the other hand, absolute errors or random estimates cause a significant drop in the resulting average throughput. Similar trends appear in the stability and headroom results (see Figures 15(b) and 15(c)).

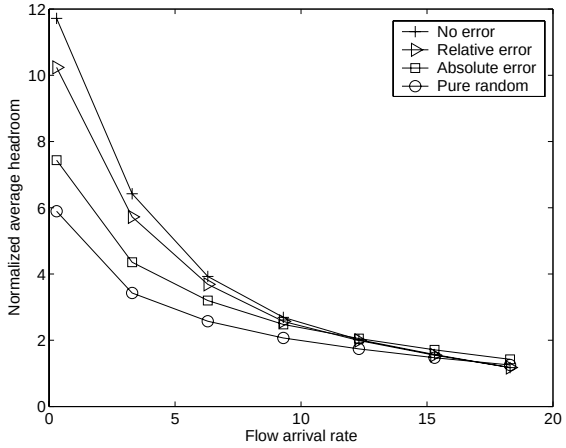
What is the reason that even large relative errors have no significant impact on the efficiency of overlay routing? The answer is actually simple if we consider that overlay routing decisions do not depend on the actual (absolute) avail-bw, but on the ranking of different paths in terms of avail-bw. After analyzing several simulations, we found that an overlay flow can be found in one of two main conditions. First, one of the paths that the flow can choose from has significantly higher avail-bw than the rest. Second, several of the paths that the flow can choose from have similar avail-bw. In the former, a relative error



(a) Efficiency



(b) Stability



(c) Safety margin

Figure 15: Effect of measurement error.

would affect all candidate paths, but the path with the maximum avail-bw would still be reported as the best with high probability. For example, if two paths have avail-bw 1000 and 100, a relative error of $\pm 100\%$ would result in a uniformly distributed value within [0,2000] and [0,200], respectively; the probability that the first path would be reported as better than the second is 95%. In the latter, the measurement error can affect the selection of the best path, but because the flow chooses among paths with similar avail-bw values the path selection does not matter significantly. Note that the previous reasoning would not hold true in the case of absolute errors or random estimate.

In summary, the previous experiments showed that relative errors in the avail-bw estimation process (which are common and probably unavoidable) will not have significant impact

on the performance of hybrid and reactive overlay routing. This implies that even relatively simple measurement techniques, which produce a "ballpark" estimate of the avail-bw, may be useful in overlay routing applications.

3.3.7 Native layer link sharing

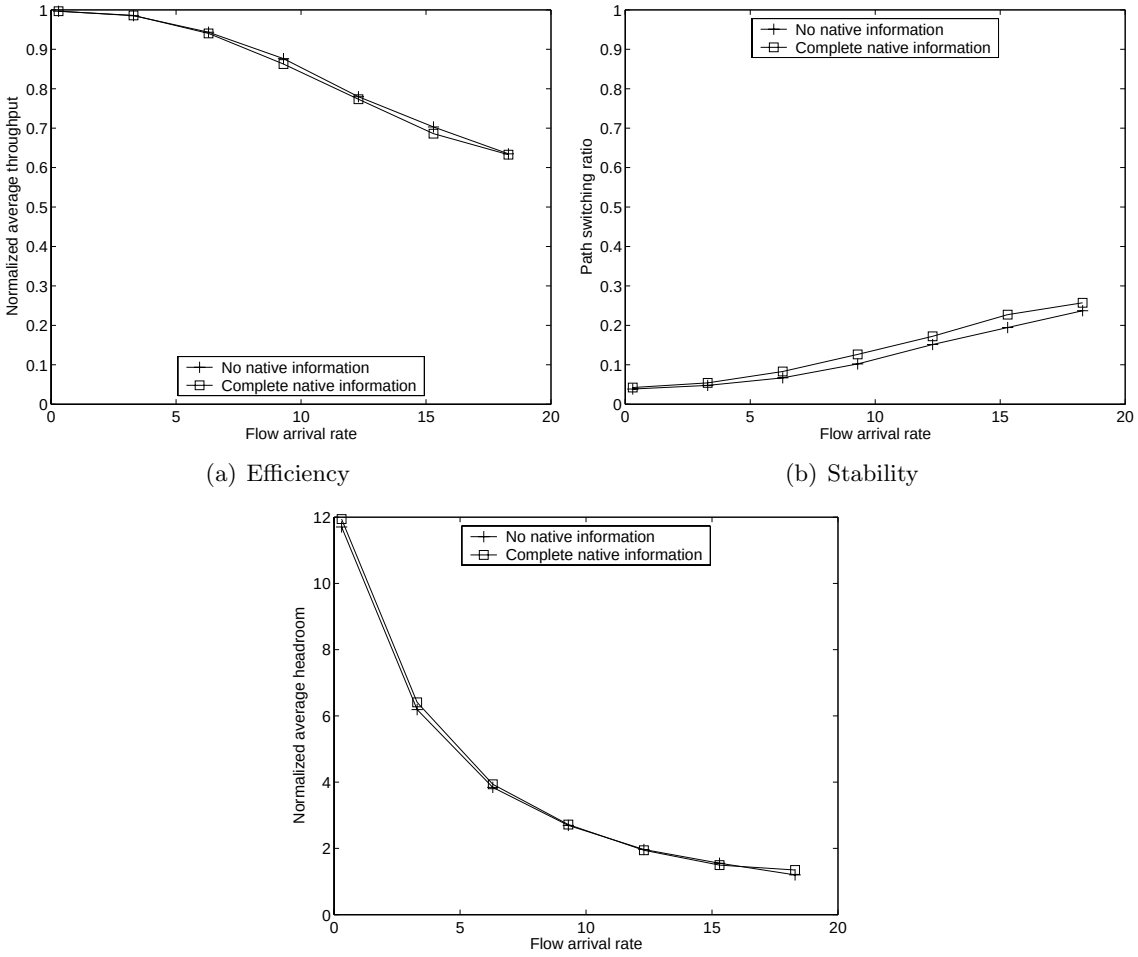
Two overlay links (or paths) may share one or more native links. Such "native sharing" effects may not be visible to the overlay network, which typically has information only for the overlay links and their avail-bw. Furthermore, as will be shown next, native sharing can cause errors in the selection of the maximum headroom path.

To see the effect of native sharing on overlay routing, we compare the following two different models, in terms of the amount of knowledge the overlay network has about the native network:

- *No-information:* The overlay network has absolutely no knowledge about the native topology or about the avail-bw of native links. This implies that when native sharing occurs, the overlay nodes may judge incorrectly which path provides the maximum headroom. On the other hand, this mode of operation would be the simplest to implement, as it does not require any sophisticated probing techniques for the detection of native sharing.

- *Complete-information:* This is an ideal case, in which the overlay nodes have accurate information about both the native layer topology and the avail-bw of each native link. This information would enable an overlay node to correctly determine the maximum headroom path even when native sharing takes place. Note that, at least so far, there are no measurement techniques that can estimate the avail-bw of each native link in a network path.

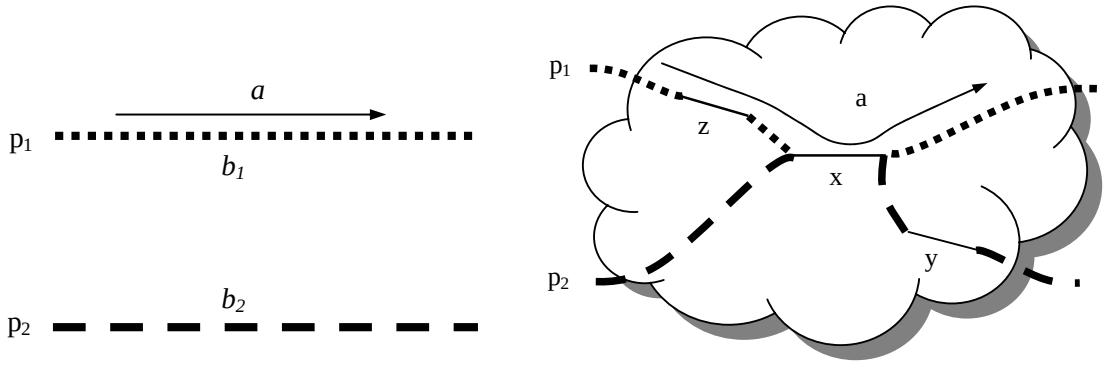
Figure 16 shows the performance of hybrid overlay routing, as a function of the offered load, with the previous two modes of operation. Perhaps surprisingly, notice that the "no-information" mode performs almost identically to the "complete-information" mode. This means that overlay routing can be equally effective even if it ignores the subtle effects of native sharing. What is the reason for this counter-intuitive effect?



(a) Efficiency (b) Stability

(c) Safety margin

Figure 16: Hybrid routing performance with and without information about the native network.



(a) Overlay topology (b) Native sharing

Figure 17: Native sharing between two overlay paths.

To explain, we will use the following model. Suppose that we have two overlay paths p_1 and p_2, as shown in Figure 17(a). The flow under consideration f is currently routed on p_1, and its throughput is a. The avail-bw in the two paths is b_1 and b_2, respectively.

Ignoring any potential native sharing ("no-information" model), the overlay routing algorithm will estimate the headroom of the two paths as follows:

$$h'(f, p_1) = b_1 + a \quad h'(f, p_2) = b_2$$

In the "complete-information" model, the headroom of the two paths will be denoted by $h''(f, p_1)$ and $h''(f, p_2)$, respectively. Based on the type of native sharing between p_1 and p_2, we have the following three cases:

1. The paths p_1 and p_2 do not share any native link: In this case, $h''(f, p_1) = b_1 + a$, and $h''(f, p_2) = b_2$. Therefore, the complete-information model provides the same headroom estimates as the no-information model, and so there is no difference in the performance of overlay routing between the two models.

2. The paths p_1 and p_2 share a native link x, but x is not the bottleneck of p_2, *i.e.,* the avail-bw in p_2 is limited by a native link y and it is not affected by flow f. In this case, we still have that $h''(f, p_1) = b_1 + a$, and $h''(f, p_2) = b_2$. Again, the two native information models would lead to the same routing decision.

3. The paths p_1 and p_2 share a native link x, and x is the bottleneck of path p_2 (shown in Figure 17(b)). This case can be further analyzed as two different sub-cases.

   (a) x is also the bottleneck of p_1. Then, $h''(f, p_1) = b_x + a$ and $h''(f, p_2) = \min(b_y, b_x + a)$, where y is the native link with the second lowest available bandwidth along path p_2, and b_x and b_y is the avail-bw in links x and y respectively. Therefore, $h''(f, p_1) \geq h''(f, p_2)$. In this case, the complete-information model would report that the maximum headroom path is p_1. Notice however that this is also what the no-information model would report, because $h'(f, p_1) = b_x + a > h'(f, p_2) = b_x$. Therefore, the overlay routing algorithm would choose the same path with both native information models.

(b) x is not the bottleneck of p_1. Then, $h''(f, p_1) = b_z + a$ and $h''(f, p_2) = \min(b_y, b_x + a)$, where z is the bottleneck of p_1. In the no-information model, the headroom of each path would be estimated as: $h'(f, p_1) = b_z + a$ and $h'(f, p_2) = b_x$. In this case, it is easy to verify that the no-information and complete-information models would disagree in their path selection only if the following inequality holds:

$$b_y > b_z + a > b_x > b_z \tag{6}$$

In this case, the complete-information model would cause a path change, while the no-information model would not.

The previous analysis shows that a naive overlay routing algorithm that ignores any native sharing effects would incorrectly choose the current path over another path with larger avail-bw only in a very specific scenario of native sharing. In that scenario, the two paths share at least one link x which is the bottleneck of p_2 but not of p_1, and the avail-bw of the three involved links x, y, and z satisfy (6). Apparently, this scenario does not happen often in our simulations, which explains why the simulation results for the no-information and complete-information models are so close. Note that the slightly higher path switching ratio of the complete-information model in in Figure 16(b) is due to the few cases which (6) is true.

In summary, native sharing effects can affect, in principle, the performance of overlay routing. Fortunately, however, an overlay routing algorithm that ignores native sharing would rarely choose a different path even if it had complete information about the native topology and the avail-bw of each native link.

3.4 Summary

This chapter presented a simulation study of dynamic overlay routing. Given that most previous work focused on delay-driven path selection, we focused instead on avail-bw based overlay routing algorithms leveraging the recently developed measurements techniques for end-to-end avail-bw. We considered two main approaches on overlay routing, proactive and

reactive, as well as a number of factors that can affect the performance of these routing algorithms.

The main conclusions of this study follow:

- Reactive overlay routing performs better in terms of efficiency than native or proactive overlay routing. The efficiency gain compared to native routing can be substantial, especially if the network is not very lightly loaded. Also, reactive routing is much much stable than proactive routing.

- Proactive overlay routing performs better in terms of headroom (safety margin) than native and reactive overlay routing.

- A single intermediate overlay node is sufficient for reactive routing to achieve its throughput and headroom gain over native routing. For proactive routing, limiting the maximum overlay hop count H_{max} to two is even more critical in terms of efficiency and stability.

- The reactive algorithm is quite robust to stale link-state information, and it performs better than native routing even when the link-state refresh period P_r is a few seconds. The proactive algorithm, on the other hand, is very sensitive to link-state staleness, and P_r should be as short as possible.

- A hybrid algorithm that acts reactively in about 90% of the time and proactively in about 10% of the time, can achieve a good compromise between high throughput, stability and safety margin, combining the best features of reactive and proactive routing.

- Overlay routing performs better with longer overlay flows, because the latter create lower traffic variability. Cross traffic variations can also decrease the performance of overlay routing, especially when these variations are significant in lower time scales than the path update period P_u.

- Relative errors in the avail-bw estimation process (which are common and probably

unavoidable) will have negligible impact on the efficiency of hybrid overlay routing. Absolute or random errors, on the other hand, can have a significant impact.

- Even though native sharing effects can affect the performance of hybrid overlay routing, ignoring native sharing performs almost equally well with having complete information about the native network.

CHAPTER IV

VIRTUAL NETWORK ASSIGNMENT

Recent proposals for network virtualization provide a promising way to overcome the Internet ossification. The key idea of network virtualization is to build a diversified Internet to support a variety of network services and architectures through a shared substrate. A major challenge in network virtualization is to assign substrate resources to virtual networks (VN) *efficiently* and *on-demand*. This chapter focuses on two versions of the VN assignment problem: VN assignment without reconfiguration (*VNA-I*) and VN assignment with reconfiguration (*VNA-II*). For the VNA-I problem, we develop a basic scheme as a building block for all other advanced algorithms. Subdividing heuristics and adaptive optimization strategies are then presented to further improve the performance. For the VNA-II problem, we develop a selective VN reconfiguration scheme that prioritizes the reconfiguration of the most critical VNs. Extensive simulation experiments demonstrate that the proposed algorithms can achieve good performance under a wide range of network conditions.

4.1 Network virtualization

Network virtualization provides a promising way to address the ossification of the Internet [57]. In such an approach, a substrate network provider (SNP) provides a common substrate to support a number of diversified virtual networks (DVN). These DVNs, implemented by *virtual routers* connected through *virtual links*, in turn would provide a range of different communication services to applications, using a variety of protocols (shown in Figure 18). In doing so, the burdens of providing an "everything-to-everybody" network architecture or requiring universal agreements on architecture and protocol changes are released. Furthermore, the diversified Internet would make it possible for an individual or organization to rapidly deploy a DVN with potentially global scope, without making any direct investment in physical infrastructure [84].

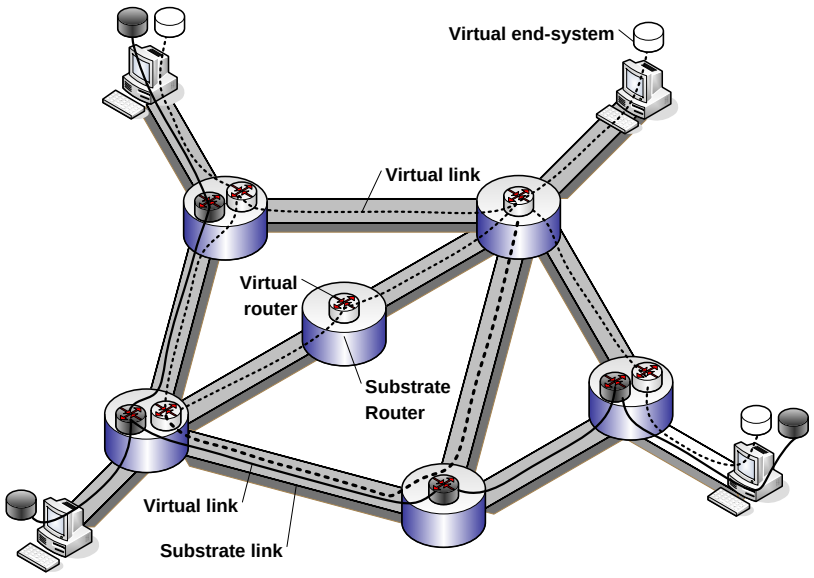


Figure 18: Network virtualization: Building a diversified Internet

One of the fundamental functions of network virtualization is the vna mapping or assigning of substrate network resources to individual virtual nodes and links. Specifically, each virtual node is assigned to a substrate node and each virtual link is assigned to a substrate path that connects the corresponding end nodes. In our envisioned scenario, DVN demands would arrive at different time instances and request to set up virtual networks (VNs) with different topologies and lifetimes. Allowing VNs to be assigned to the substrate network *efficiently* and *on-demand* is desirable for the following reasons:

- Increasing efficiency in the substrate resource utilization would allow the substrate network to accommodate more VNs with limited resources and reduce hot spots or congestion.

- On-demand assignment means that the assignment for each VN is determined based on the current network situation as the demand arrives. Given that DVN demands can arrive at any moment and the SNP would not have information regarding future arrivals, it is important for the SNP to be able to make the assignment decision in response to each individual demand as they arrive.

In this work, we are motivated by the above considerations to study the following on-demand VN assignment problem: Upon the arrival of a VN request, assign its topology to

the substrate network to achieve *low* and *balanced* load on both substrate nodes and links. A special case of this problem can be formulated as an unsplittable flow problem which is NP-hard. Therefore, the VN assignment problem is intrinsically difficult and heuristics will be used to solve the problem.

This chapter focuses on the algorithm design for two versions of the VN assignment problem: VN assignment without reconfiguration (*VNA-I*) and VN assignment with reconfiguration (*VNA-II*). For the VNA-I problem, where the VN assignment is fixed throughout the VN lifetime, we develop a basic scheme to achieve near optimal substrate node performance and use it as a building block for all other advanced algorithms. Subdividing heuristics and adaptive optimization strategies are then presented to further improve the performance. For the VNA-II problem, we develop a selective VN reconfiguration scheme that prioritizes the reconfiguration for the most critical VNs. In doing so, we can achieve most performance benefits of the reconfiguration without excessively high cost.

4.2 Network Model and Problem Description

4.2.1 Network model

We model the topology of the substrate network as a graph $G_S = (V_S, E_S)$, where V_S is the set of substrate nodes and E_S is the set of substrate links. The i'th VN arrives at time a_i and lasts for a certain lifetime. Its topology is modelled as a graph $G_V^i = (V_V^i, E_V^i)$ with the set of virtual nodes V_V^i and the set of virtual links E_V^i respectively [1]. The assignment of the i'th VN includes following two components:

- Node assignment: Each virtual node is assigned to a different substrate node. It is formalized as a mapping $f_N^i : V_V^i \rightarrow V_S$ from virtual nodes to substrate nodes such

[1] More generally, a set of vna mapping constraints (such as the substrate capacity of the substrate nodes/links, locations of access routers) may be attached to the VN assignment problem. In this thesis, we focus on the basic scenario, where there are no constraints attached. The solution of the basic problem can be easily extended to the constrained version by checking the constraints and considering only feasible assignments.

that

$$f_N^i(\hat{v}) \in V_S, \forall \hat{v} \in V_V^i$$

$$f_N^i(\hat{u}) = f_N^i(\hat{v}), \textit{iff}\, \hat{u} = \hat{v}$$

- Link assignment: Each virtual link is assigned to a substrate path between the corresponding substrate nodes. It is formalized as a mapping $f_L^i : E_V^i \rightarrow \mathcal{P}_S$ from virtual links to substrate paths such that

  $$f_L^i(\overline{\hat{u}\hat{v}}) \in P_S(f_N^i(\hat{u}), f_N^i(\hat{v})), \forall \overline{\hat{u}\hat{v}} \in E_V^i$$

  where $\mathcal{P}_S$ is the set of all substrate paths, $P_S(s,t)$ is the set of substrate paths from node s to node t. $\overline{\hat{u}\hat{v}}$ is the virtual link from virtual node $\hat{u}$ to $\hat{v}$.

When a VN arrives, the substrate network determines the above assignments and allocates resources on the selected substrate nodes and paths accordingly. The allocated substrate resources are released when the VN departs. Furthermore, if an existing VN changes its assignment, the substrate network will release previously allocated resources by the old assignment and allocate new resources based on the new assignment.

To quantify the resource usage of the substrate network, we use the notion of *stress*. The substrate node stress at time t, $S_N(t,v)$, is defined as the number of virtual nodes that are assigned to the substrate node $v \in V_S$. Similarly, the substrate link stress at time t, $S_L(t,e)$, is defined as the number of virtual links whose substrate path passes through the substrate link $e \in E_S$. Therefore, the arrival, departure and reconfiguration of the i'th VN determines the network state transition as follows:

- The i'th VN's arrival:

  $S_N(t,v) = S_N(t^-,v) + 1, \forall v \in \{f_N^i(\hat{v}) | \hat{v} \in V_V^i\}$

  $S_L(t,e) = S_L(t^-,e) + C(e), \forall e \in \{f_L^i(\hat{e}) | \hat{e} \in E_V^i\}$

- The i'th VN's departure:

  $S_N(t,v) = S_N(t^-,v) - 1, \forall v \in \{f_N^i(\hat{v}) | \hat{v} \in V_V^i\}$

$$S_L(t,e) = S_L(t^-,e) - C(e), \forall e \in \{f_L^i(\hat{e}) | \hat{e} \in E_V^i\}$$

- The i'th VN's reconfiguration:

  $S_N(t,v) = S_N(t^-,v) - 1, \forall v \in \{f_N^i(\hat{v}) | \hat{v} \in V_V^i\}$

  $S_N(t,v') = S_N(t^-,v') + 1, \forall v' \in \{\hat{f_N^i}(\hat{v}) | \hat{v} \in V_V^i\}$

  $S_L(t,e) = S_L(t^-,e) - C(e), \forall e \in \{f_L^i(\hat{e}) | \hat{e} \in E_V^i\}$

  $S_L(t,e) = S_L(t^-,e') + C(e'), \forall e' \in \{\hat{f_L^i}(\hat{e}) | \hat{e} \in E_V^i\}$

where t^- and t are the time instances immediately before and after the arrival, departure or reconfiguration event. $C(e)$ is the number of times that the substrate link e appears in all selected substrate paths for the i'th VN. (f_N^i, f_L^i) and $(\hat{f_N^i}, \hat{f_L^i})$ are the original and the reconfigured assignments for the i'th VN respectively.

4.2.2 Virtual network assignment problem description

Given a substrate topology G_s with the current stress level on each substrate link and node, upon the arrival of a VN request G_v^i, our goal is to assign substrate nodes and links to realize the requested virtual topology (as shown in Figure 19).

There are various ways to map a VN to the substrate network. The key issue is how to efficiently use substrate resources. Due to limitations on processing power, each substrate node could only be virtualized into a certain number of virtual nodes. Therefore, maintaining low stress across all substrate nodes is important for the efficient use of network resources.

Maintaining low link stress is also important in both provisioned substrate links (*i.e.,* substrate links with performance guarantees for its carried virtual links) and unprovisioned substrate links (*i.e.,* substrate links without performance guarantees). 1) For provisioned links, there is a provisioned capacity that could be allocated to virtual links. Reducing the stress will reduce the total bandwidth needed along that link. Furthermore, to provide guaranteed performance, special efforts are needed to isolate different virtual links within the same provisioned substrate link. Therefore, reducing the substrate link stress would reduce

the above isolation efforts. 2) For unprovisioned links, reducing the stress will reduce the effects of cross traffic interferences since less virtual links are sharing the substrate link.

Furthermore, having a balanced stress across the substrate network is essential to avoid hot spots and reduce congestion. Therefore, the objective of the VN assignment problem is to maintain *low* and *balanced* stress among all substrate links and substrate nodes.

To evaluate the stress balancing performance, we define the node stress ratio (R_N) as the ratio of the maximum node stress and the average node stress across the whole substrate network. Similarly, the link stress ratio (R_L) is the ratio of the maximum link stress and the average link stress, *i.e.,*

$$R_N(t) = \frac{\max_{v \in V_S} S_N(t, v)}{\big[\sum_{v \in V_S} S_N(t, v)\big]/|V_S|} \tag{7}$$

$$R_L(t) = \frac{\max_{e \in E_S} S_L(t, e)}{\big[\sum_{e \in E_S} S_L(t, e)\big]/|E_S|} \tag{8}$$

The stress ratio is a proper metric to evaluate the load balancing performance since a smaller stress ratio indicates that the stress is more balanced. If the node (or link) stress ratio is 1, the network achieves optimal load balancing since all substrate nodes (or links) have the same stress.

Although the link stress ratio reflects the load balancing performance, it may not be sufficient to reveal the true efficiency of the VN assignment. Figure 20 illustrates the problem of the stress ratio. The substrate network has a 3-node 2-link topology (shown as shadowed circles and thick links). There are 6 single-link VNs (represented by ellipses and links connecting them) that are needed to be assigned to the substrate network. Both Figure 20(a) and 20(b) achieve perfect link stress balancing since their $R_L = 1$. However, the latter is not efficient in using the link resources, since its links have higher stress. This indicates that assignments with different efficiency could have the same R_L. The key factor here is the maximum link stress. Therefore, instead of using the stress ratio, we use the maximum link/node stress as the performance metrics, *i.e.,* we want to minimize both the maximum node stress and maximum link stress.

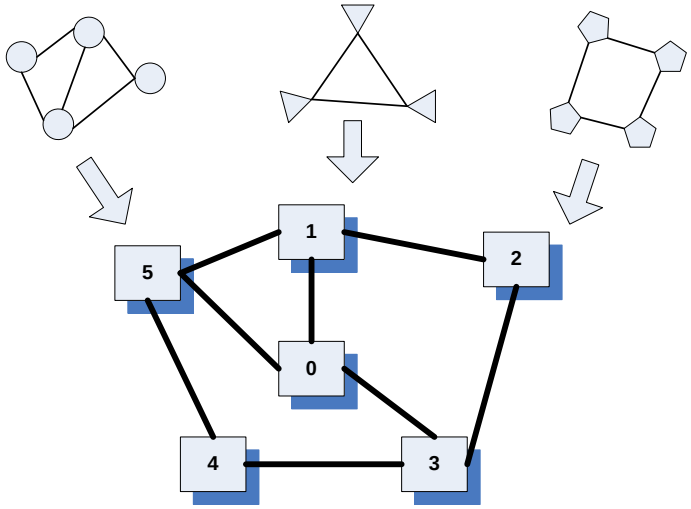
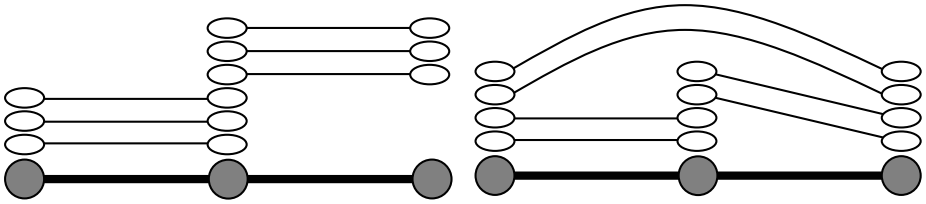


Figure 19: VN assignment: Allocate substrate resources to VN nodes and links.



(a) Optimal link stress (b) Optimal node stress

Figure 20: Different virtual network assignments: (a) achieves the optimal link stress and (b) achieves the optimal node stress

Due to the heterogeneity of the substrate topology, it may not be possible to achieve optimal stress for both substrate nodes and links. Again, using Figure 20 as an example, it is easy to verify that the assignment in Figure 20(a) minimizes the maximum link stress and its maximum link stress and maximum node stress are 3 and 6 respectively. Figure 20(b), on the other hand, minimizes the maximum node stress and the corresponding maximum link stress and maximum node stress are both 4. In fact, there is no such an assignment for these VNs that could achieve both the optimal node stress and optimal link stress. Therefore, instead of having a single term, the objective function of the VN assignment problem should be defined as the combination of the two:

$$\alpha \cdot \max_{v \in V_S} S_N(a_i, v) + \beta \cdot \max_{e \in E_S} S_L(a_i, e) \tag{9}$$

where a_i is the time instance immediately after the i'th VN arrival, $\alpha, \beta > 0$ are weights for the node stress objective and link stress objective respectively and are determined by the specific application scenario.

To summarize, the VN assignment problem for the i'th VN can be formalized as the following snapshot optimization problem upon the VN's arrival:

- Inputs:
  1. Substrate network information: The substrate topology G_S, snapshot of the current substrate node stresses $\{S_N(a_i{}^-, v) | \forall v \in V_S\}$ and link stresses $\{S_L(a_i{}^-, e) | \forall e \in E_S\}$, where $a_i{}^-$ is the time instant immediately before the i'th VN arrival.
  2. The requested VN topology G_V^i
- Find the vna mappings from the the i'th VN topology to the substrate topology: f_N^i and f_L^i
- Minimizing

$$\alpha \cdot \max_{v \in V_S} S_N(a_i, v) + \beta \cdot \max_{e \in E_S} S_L(a_i, e)$$

In this thesis, we consider two versions of the VN assignment problem: VN assignment without reconfiguration (VNA-I) and VN assignment with reconfiguration (VNA-II). The

former refers to the case where the VN assignment is fixed for the whole VN lifetime and the latter refers to case where the VN assignment is allowed to changed during the VN lifetime. In the next two sections, we will present algorithms for both versions of the problem respectively.

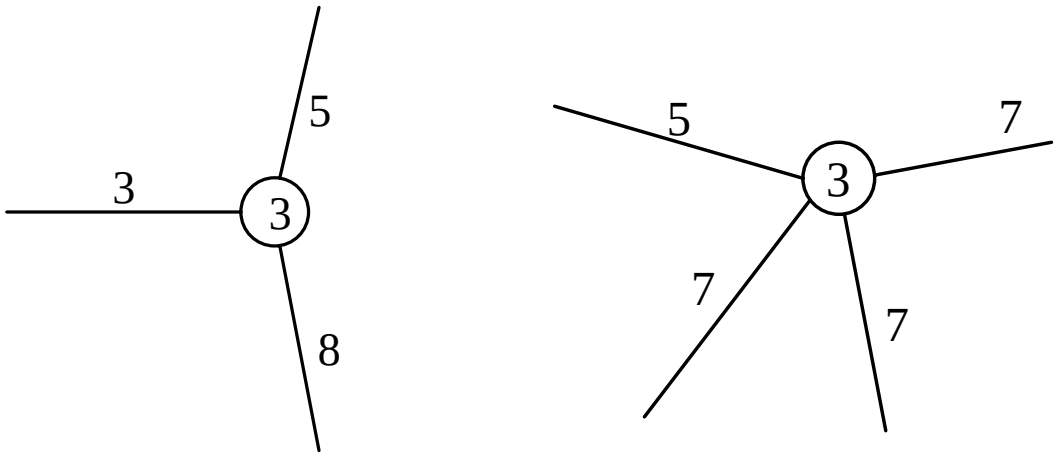
4.3 VNA-I: VN Assignment without Reconfiguration

For each VN request, if the vna mappings of the virtual nodes are given, then the VN assignment problem becomes an off-line load balancing routing problem where the source and destination are the ingress and egress nodes of each virtual link and each flow has a unit demand. This can be formulated as a well-known NP-hard unsplittable flow problem where the objective is to minimize the maximum link load [40,41,62]. The general VN assignment problem is even harder since the node vna mapping could be arbitrary and we want to optimize both node stress and link stress. Therefore, heuristic approaches are used to solve the problem. In this section, we will first describe a basic VN assignment scheme that acts as a building block for all other advanced algorithms. An improved VN assignment algorithm that subdivides the VN topology is then presented. Adaptive optimization strategies that achieve desirable tradeoffs between the node stress and link stress optimization objectives are also discussed.

4.3.1 A basic VN assignment algorithm

A naive way to perform VN assignment is to treat the node stress optimization and link stress optimization as two independent sub-problems and solve them sequentially. However, we will see from the simulation results presented later that this approach usually results in high link stress since the selected substrate nodes may be far away from each other. In this subsection, we propose a VN assignment algorithm that considers both node and link stresses throughout the VN assignment process.

The key idea of our algorithm is to select a cluster of substrate nodes that are not only lightly loaded but also likely to result in low substrate link stresses when they are connected. This is achieved through a coarse-to-fine approach: First, a cluster center in the substrate network is identified based on the stress level in its neighborhood. Substrate nodes are then



(a) (b)

Figure 21: Neighborhood resource availability, numbers represent the stress

selected sequentially based on their distances from the previously assigned virtual nodes.

4.3.1.1 Cluster center localization

Since any directly connected virtual link of a virtual node will go through one of the directly connected substrate links of the corresponding substrate node, substrate nodes with high link stresses on all their directly connected links should be avoided. To quantify the stress level of a substrate node and its connected substrate links, we define the following neighborhood resource availability (NR):

$$\mathrm{NR}(t,v) = [S_{\mathrm{nmax}}(t) - S_N(t,v)] \cdot \sum_{e \in L(v)} [S_{\mathrm{lmax}}(t) - S_L(t,e)] \tag{10}$$

where $S_{\mathrm{nmax}}(t) = \max_{v \in V_S} S_N(t,v)$ and $S_{\mathrm{lmax}}(t) = \max_{e \in E_S} S_L(t,e)$ are the maximum node stress and maximum link stress of the substrate network respectively. $L(v)$ is the set of substrate links directly connected to substrate node v.

The above definition of NR captures both the node stress and neighboring link stresses in the sense that a substrate node with a high NR means that both the node itself and some of its directly connected links are lightly loaded. The cluster center, which is the first selected substrate node, is therefore identified as the substrate node with the highest NR value. In the example shown in Figure 21, assume $S_{\mathrm{nmax}}(t) = 5$ and $S_{\mathrm{lmax}}(t) = 8$,

then the NR for the substrate node in Figure 21(a) and 21(a) are 16 and 12 respectively. Therefore, the former substrate node is preferred even though it has a lower node degree.

4.3.1.2 Substrate node selection

After identifying the cluster center, the next step is to select the rest of the substrate nodes for the VN request. Rather than simply selecting nodes with minimum stresses, our node selection algorithm also considers the link stress by weighting the node stress based on its distances to other selected substrate nodes. We adopt the definition of distance from the shortest-distance path algorithm [49], which can effectively balance the network load among links and efficiently utilize network resources. Based on this notion, the distance of a single substrate path p at time t is defined as:

$$d(t,p) = \sum_{e \in p} \frac{1}{S_{\text{lmax}}(t) + \delta_L - S_L(t,e)} \tag{11}$$

where $\delta_L \ll S_{\text{lmax}}(t)$ is a small positive constant to avoid dividing by zero in computing the distance. And the distance between two substrate nodes u and v is defined as the minimum distance of all paths between them, *i.e.,*

$$D(t,(u,v)) = \min_{p \in P_S(u,v)} d(t,p) \tag{12}$$

Since the VN topology could be arbitrary, there could be a virtual link between any pair of virtual nodes. Therefore, to minimize the overall distance, the weight a virtual node is set to be the summation of distances to all previously assigned virtual nodes. Combining the distance and node stress, we use the following *node potential* as the criteria for substrate node selection:

$$\pi(t,v) = \frac{\sum_{u \in V_A} D(t,(v,u))}{S_{\text{nmax}}(t) + \delta_N - S_N(t,v)} \tag{13}$$

where V_A is the set of selected substrate nodes for the same VN and $\delta_N \ll S_{\text{nmax}}(t)$ is a small positive constant to avoid dividing by zero. We choose $\delta_L = \delta_N = 1$ as the default value. The set of substrate nodes with the minimum potential are then selected.

After all substrate nodes are determined, we need to map the virtual nodes into the selected substrate nodes such that virtual nodes with higher degree are mapped to substrate nodes with higher NR. The intuition behind this is that virtual nodes with higher degree will setup more virtual links. All of these virtual links will go through some of the substrate link in the neighborhood of the corresponding substrate node. Therefore,the above matching tends to reduce the congestion and balance the link stress.

4.3.1.3 Substrate path selection

The last step of the basic VN assignment scheme is to connect the selected substrate nodes based on the virtual topology. This is a standard routing problem and we use the shortest-distance path algorithm [49] again to select the path with minimum distance (defined in Eq. (11)) between corresponding substrate nodes. Therefore, it is consistent with Eq. (12) in our node selection process. The detailed basic VN assignment algorithm is given in Algorithm 2. The basic VN assignment algorithm presented above can be used directly to

Algorithm 2 Basic VN assignment algorithm (Upon the i'th VN arrival)

INPUTS:
$G_s = (V_S, E_S)$: substrate topology;
$S(a_i^-, v), \forall v \in V_S$: current link stress;
$S(a_i^-, e), \forall e \in E_S$: current node stress;
$G_v^i = (V_V^i, E_V^i)$: VN topology;
OUTPUTS:
$f_N^i(\hat{v}), \forall \hat{v} \in V_V^i$ and $f_L^i(\hat{e}), \forall \hat{e} \in E_V^i$

$V_A = \emptyset$ {Note: V_A is the set of selected substrate nodes}
$V_A = V_A \cup \{\arg\ \max_{v \in V_S} \mathrm{NR}(a_i^-, v)\}$
for $i = 2, \ldots, |V_V^i|$ **do**
  $V_A = V_A \cup \{\arg\ \min_{v \in V_S - V_A} \pi(a_i^-, v)\}$
end for
Assign nodes in V_A to virtual nodes such that
$\mathrm{NR}(f_n^i(\hat{u})) \geq \mathrm{NR}(f_n^i(\hat{v}))$, *iff* $\mathrm{Degree}(\hat{u}) \geq \mathrm{Degree}(\hat{v})$
Find the shortest-distance path for all virtual links

allocate substrate resources for VN requests. However, since its node selection process does not take the VN topology into consideration, the basic algorithm may become inefficient when the VN has a sparse topology. Furthermore, the basic algorithm always compromises the node stress performance with the link stress performance regardless of the current

network situation. More intelligent approaches are discussed next to address these issues.

4.3.2 Subdividing the virtual network

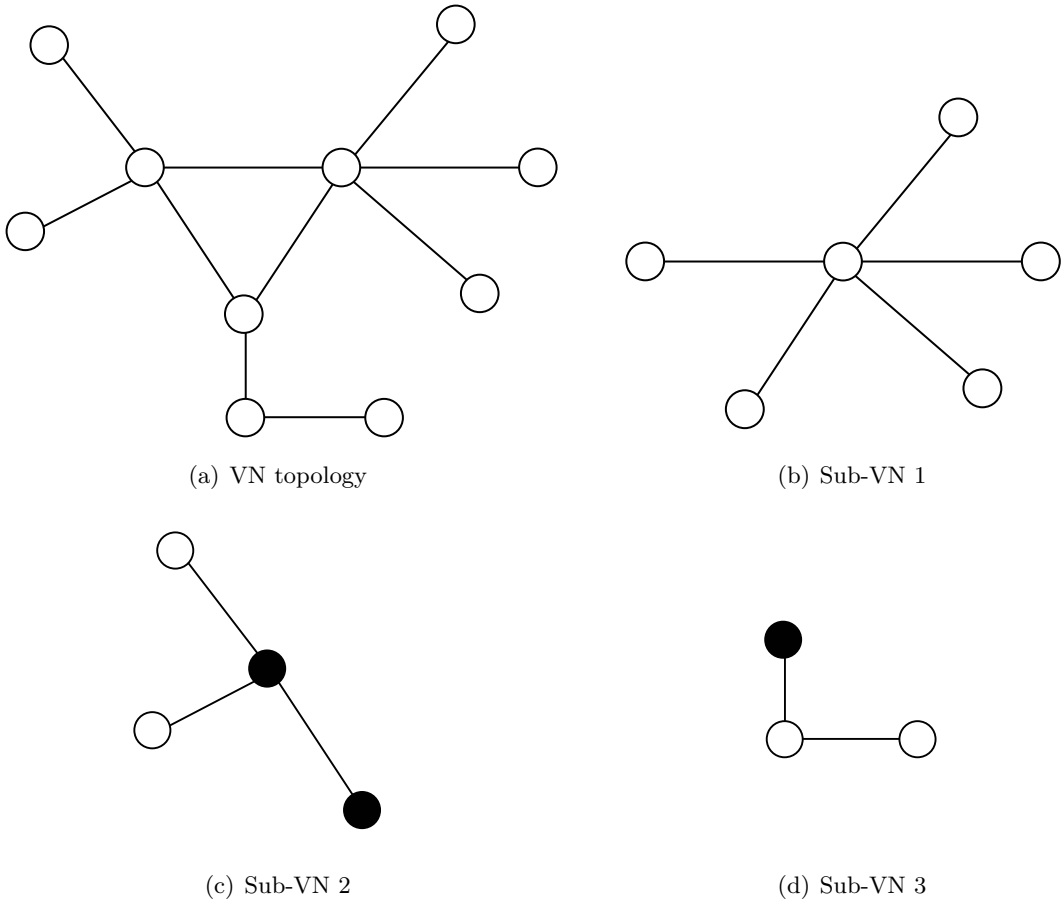
In the basic VN assignment algorithm, the requested VN is mapped to the substrate network as a whole unit. Small VNs have smaller footprints on the substrate network and are therefore easier to fit into isolated low stressed subregions in the substrate network. As an extreme case, if every VN contains only a single virtual link, then we have the most flexibility to allocate substrate network resources and can always achieve optimal link stress performance. As the VN topology becomes larger, it is more difficult to balance the load if we consider the whole VN at once. To utilize the flexibility of the small topology, we break up the VN into a number of connected sub-virtual networks ($subVN$). Each subVN is of a simple star topology and the connections between subVNs define the constraints in substrate node selection. Another advantage of subdividing is that by breaking down the big network into a number of small subVNs, computational requirements are reduced since less virtual nodes and links are considered in each subVN assignment process. Furthermore, since all subVNs have star topologies, the only possible links in the subVN are to/from the subVN center. Therefore, we can be more accurate in calculating the node potential in Eq. (13), since $D(t,(u,v))$ can be summed over exact virtual links instead of assuming virtual links to every previously assigned virtual nodes.

The subdividing algorithm works in a recursive way: starting from the original VN topology $G_V = (V_V, E_V)$, the algorithm finds the virtual node with the highest degree as the center of the subVN. All virtual nodes directly connected to the selected center and the corresponding virtual links form a subVN and are removed from the original VN. This process continues on the residual topology $G_R = (V_R, E_R)$ until all the virtual links have been assigned to a certain subVN. The detailed algorithm is given in Algorithm 3. Figure 22 shows an example of the subdividing process. For the original VN shown in Figure 22(a), the subdividing algorithm sequentially generates three subVNs as shown in Figure 22(b), 22(c) and 22(d) respectively.

Algorithm 3 Subdividing algorithm

INPUTS: $G_V = (V_V, E_V)$
OUTPUTS: $G_{V(i)} = (V_{V(i)}, E_{V(i)}), i = 1, 2, \cdots$

$G_R = G_V, i = 0, G_{V(i)} = \emptyset$
while $G_R \neq \emptyset$ **do**
  $\hat{v} = \arg\ \max_{\hat{v}' \in V_R} \text{Degree}(\hat{v}')\}$
  $V_{V(i)} = V_{V(i)} \bigcup \{\hat{v}\}$
  for $\forall \hat{u} \in V_R$ **do**
    if $\overline{\hat{v}\hat{u}} \in E_R$ **then**
      $V_{V(i)} = V_{V(i)} \bigcup \{\hat{u}\}$, $E_{V(i)} = E_{V(i)} \bigcup \{\overline{\hat{v}\hat{u}}\}$
    end if
  end for
  $G_R = G_R - G_{V(i)}$, $i = i + 1$
end while



(a) VN topology (b) Sub-VN 1

(c) Sub-VN 2 (d) Sub-VN 3

Figure 22: Subdividing the virtual network, constrained nodes are marked in black.

The above process divides a VN topology into a number of subVNs with some inter-subVN constraints. Each subVN is then assigned to the substrate network based on the basic algorithm described in Section 4.3.1 and the attached inter-subVN constraints are handled as follows:

- Unconstrained subVN: In this case, there is no constraint on the subVN assignment so the subVN can be assigned to any location in the substrate network. Assume the requested VN topology is connected, then the only unconstrained subVN is the first one generated by the subdividing algorithm. The basic scheme can therefore be directly applied to this subVN. An unconstrained subVN is shown in Figure 22(b).

- Constrained subVN: In this case, the vna mappings of some virtual nodes in the subVN have already been determined by the previous assigned subVNs. Therefore, only nodes that are not constrained need to be assigned. Assume there are m unconstrained VN nodes, based on whether the subVN center is constrained or not, it can be further divided into the following two cases:

  - Center constrained: If the subVN center is constrained, then the node selection is based on the relative distance to the center. Specifically, the algorithm selects m substrate nodes with the least potential. A center constrained subVN is shown in Figure 22(c).

  - Center unconstrained: In this case, we need to determine the subVN center first by selecting the substrate node with the maximum NR. After that, we can use the same scheme as the center constrained case. A center unconstrained subVN is shown with Figure 22(d).

Note that since all subVNs have star topologies, the only possible virtual links are those connecting to the subVN center. Therefore, when calculating the potential in Eq. (13), V_A should be modified to contain only the subVN center.

4.3.3 Adaptive optimization strategies

We have shown in Section 4.2.2 that there are inherent tradeoffs between optimization objectives of node stress and link stress. To optimize the node stress, a simple way is to select substrate nodes with the least stress. However, this naive approach totally ignores the link stress in node selection and may result in poor link stress performance. This is because the selected nodes may be scattered far away from each other in the substrate network and choosing these nodes may result in excessively long substrate paths. The basic VN assignment scheme described in Section 4.3.1 also aims at minimizing the maximum node stress and could achieve near optimal node stress and much better link stress performance, as will be shown later in the simulation results. Therefore, it can be viewed as a node optimization strategy (*node-opt*). On the other hand, if the objective is to optimize link stress only, then the node selection and link vna mapping should be solely based on the substrate link stress. Specifically, we use the same algorithm in Algorithm 2 but with a modified definition of the NR and the node potential as follows:

$$\hat{\mathrm{NR}}(t,v) = \sum_{e \in L(v)} [S_{\mathrm{lmax}}(t) - S(t,e)] \tag{14}$$

$$\hat{\pi}(t,v) = \sum_{u \in V_A} D(t,(v,u)) \tag{15}$$

This modified algorithm aims to optimize link stress performance and therefore is called the link optimization strategy (*link-opt*).

Based on the application context, either the node-opt or link-opt strategy can be used to perform VN assignment. As we discussed earlier, due to the heterogeneity of substrate topology, it may not be possible to achieve both the optimal node performance and optimal link performance. Therefore, instead of using either the node-opt or link-opt for all VN assignments, we use an adaptive strategy that chooses between the node-opt and link-opt for any given VN request adaptively.

The node stress ratio and link stress ratio defined in Eq. (7) and (8) provide a straightforward way to identify the relative performance of link and node stress balancing. Specifically, if $R_L(t) > R_N(t)$, the substrate link stress is more unbalanced. Otherwise, the substrate

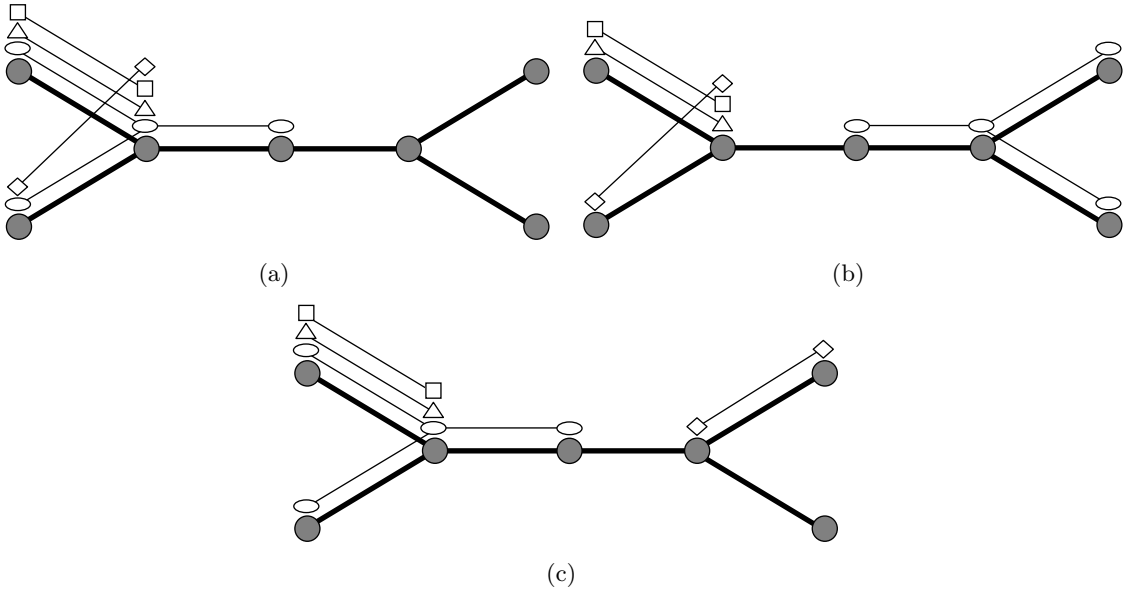


Figure 23: Effects of different reconfiguration orders

node stress is more unbalanced. Upon the arrival of a new VN request, the adaptive scheme determines if the node stress or link stress is more unbalanced and performs the node-opt or link-opt strategy accordingly. More generally, we can set an *adaptive threshold* $\eta \in [0, +\infty)$, and use the adaptive scheme shown in Algorithm 4. The adaptive strategy is especially effective if the objective of the VN assignment is in the form of Eq. (9). In that case, the value of η could be determined based on values of α and β. For example, we can set $\eta = \alpha/\beta$.

Algorithm 4 Adaptive optimization strategy (Upon the i'th VN arrival)

if $R_L(a_i^-) > \eta \cdot R_N(a_i^-)$ **then**
  Perform link-opt
else
  Perform node-opt
end if

4.4 VNA-II: VN Assignment with Reconfiguration

In the dynamic process of VN assignment, network conditions change over time due to the arrival and departure of VNs. As a result, VN assignment without reconfiguration may lead to inefficient resource utilization where some part of the substrate network can become excessively loaded while others are under-utilized. Similar to the case of rerouting [87], reconfiguring existing VNs can help to improve performance under dynamic situations.

However, periodically reconfiguring all existing VNs is not desirable for a number of reasons.

First, VN reconfiguration is much more expensive than rerouting. The cost of reconfiguration includes both the computational cost and the service disruption cost. Computational cost refers to the expenses involved in recomputing the VN assignment and therefore is proportional to the frequency of VN reconfiguration. Service disruption cost is incurred because the normal operation of a VN is affected when it is switched from one assignment to another. Unlike flow rerouting where at most one path change happens per rerouting event, the reconfiguration of a single VN may result in more substantial changes involving both *node switching* and multiple *path switching*. Node switching happens when the assignment of a virtual node is changed from one substrate node to another. Path switching happens when the assignment of a virtual link is changed from one substrate path to another. The disruption of the reconfiguration is therefore significant if all VNs are reconfigured periodically. Based on the above discussion, we quantify the reconfiguration cost as follows:

$$C = w_1 \cdot R_{\text{recfg}} + w_2 \cdot R_{\text{node}} + w_3 \cdot R_{\text{path}} \tag{16}$$

where R_{recfg} is the reconfiguration rate, defined as the number of reconfiguration events per time unit. R_{node} and R_{path} are the node and path switching rates, defined as the number of substrate node and path switches per time unit respectively. $w_1, w_2, w_3 > 0$ are weights of each cost component and are determined by the specific application scenario.

Furthermore, the reconfiguration order of different VNs will affect the VN assignment performance. If there is a limit on the number of reconfigurations, reconfiguring VNs that are on maximum stressed links/nodes will be more likely to reduce the maximum stress than reconfiguring other VNs. Therefore, VNs that are on the maximum stressed links/nodes are *critical* to the performance. Even if there is no limit on the number of reconfigurations, reconfiguring non-critical VNs before critical ones may take up resources that could otherwise be used by critical VNs to improve the performance.

Figure 23 illustrates the effects of reconfiguring different VNs. Assume that the network is originally in a state shown in Figure 23(a), where there are 4 VNs on the substrate network (three 2-node VNs and one 3-node VN). The 2-node diamond VN is non-critical

since it is not carried by any maximum stressed link. In contrast, the other two 2-node VNs and the 3-node VN are all critical since they use the maximum stressed link. If the 3-node VN reconfigures first, then the network enters the state shown in Figure 23(b). The maximum link stress is reduced from 3 to 2. On the other hand, if the 2-node diamond VN reconfigures first, then the network state is shown in Figure 23(c) and the maximum link stress is unchanged. Obviously, the former reconfiguration is more desirable.

Based on this notion of criticality, we developed a selective reconfiguration scheme to perform efficient reconfiguration by giving higher priority to critical VNs. The key idea is to reconfigure only a subset of the existing VNs that are most critical in reducing the maximum stress instead of reconfiguring all existing VNs. The selective reconfiguration algorithm has two components: a global marking process and per VN reconfiguration processes.

4.4.0.1 Global marking

The marking process periodically (with the marking period τ_M) compares the node stress ratio R_N and link stress ratio R_L (Eq. (7) and (8)) to determine which one is more unbalanced. The algorithm then identifies a set of critical substrate nodes (or links) accordingly as follows:

$$V_C(t_M) = \{v | S(t_M, v) \geq (1-\theta) \cdot S_{\text{nmax}}(t_M), v \in V_S\}$$
$$E_C(t_M) = \{e | S(t_M, e) \geq (1-\theta) \cdot S_{\text{lmax}}(t_M), e \in E_S\}$$

where t_M is the time instance of the marking event. $\theta \in [0, 1]$ is the *reconfiguration threshold* that controls the stress level to be considered as critical. For example, $\theta = 0$ means that only the maximum stressed nodes/links are considered critical and $\theta = 1$ means that all links/nodes are considered critical. Next, all VNs that are currently using the critical nodes (or links) are marked for reconfiguration until the next marking event. The detailed marking algorithm is given in Algorithm 5.

Algorithm 5 Global marking algorithm

if $R_L(t_M) > R_N(t_M)$ **then**
  Identify critical substrate links $E_C(t_M)$
  Mark all VNs on critical links: $\{i | \exists e \in E_V^i, f_L^i(e) \bigcap E_C(t_M) \neq \emptyset\}$
else
  Identify critical substrate nodes $V_c(t_M)$
  Mark all VNs on critical nodes: $\{i | \exists v \in V_V^i, f_N^i(v) \in V_C(t_M)\}$
end if

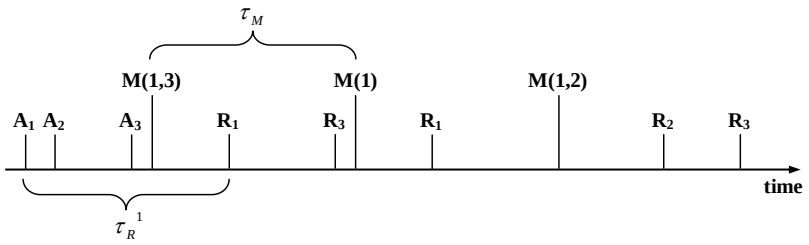


Figure 24: Global marking and per VN reconfiguration events with related time scales. A, M and R are VN arrival, marking and reconfiguration events respectively. Numbers besides A, R and τ_R index the VN. Numbers besides M show the indexes of marked VNs. τ_M and τ_R are the marking period and reconfiguration period respectively.

4.4.0.2 Per VN reconfiguration

The above global marking process only determines which VNs are allowed to reconfigure. The actual reconfiguration process happens at each individual VN. Upon the VN arrival, the VN assignment algorithm presented in Section 4.3 is performed to allocate substrate resources. Furthermore, each existing VN also periodically checks (with *reconfiguration period* τ_R^i for the i'th VN) if it is selected by the marking process. If it is the case, then the VN reapplies the VN assignment algorithm to compute the new assignment. Otherwise, the VN keeps its current assignment. The previous marking and reconfiguration events and the corresponding time scales are illustrated in Figure 24.

The above algorithm reduces the reconfiguration cost by selecting only a subset of VNs for reconfiguration. By allowing only critical VNs to reconfigure, it also optimizes the reconfiguration order since non-critical VNs are less likely to be reconfigured. Furthermore, since the marking process happens periodically, the algorithm can adapt to dynamic situations by selecting different sets of critical VNs accordingly.

4.5 Performance Evaluation

In this section, we first evaluate the performance of VN assignment algorithms without reconfiguration. Next, we investigate the performance of our selective VN reconfiguration algorithm under different parameter settings. Finally, we examine the VN assignment performance over a wide range of network settings.

4.5.1 Simulation settings and performance metrics

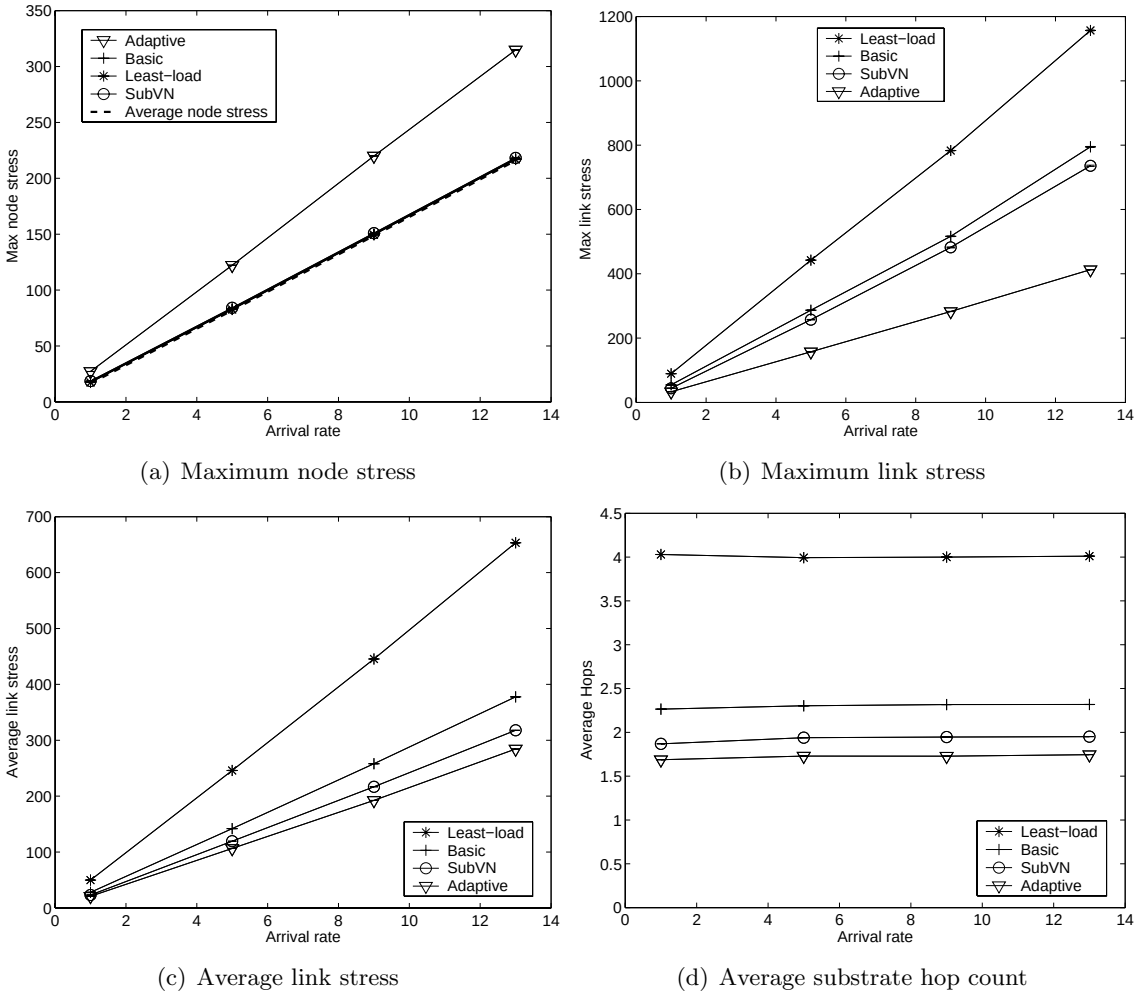
We have implemented a discrete event simulator for VN assignment. The substrate network is a 100-node 316-link random topology generated by the GT-ITM tool [90]. We assume that VN requests arrive in a Poisson process with an average rate λ VNs per time unit, and the lifetime of VNs follows an exponential distribution with an average of $\mu = 275$ time units. Unless otherwise specified, the VN topologies are random and the VN size is uniformly distributed from 2 to 10. The average VN connectivity (*i.e.,* probability that there is a virtual link between a pair of virtual nodes) is fixed to be 50%. To evaluate the performance of a VN assignment algorithm, we use Poisson sampling, with an average inter-arrival time of 1 time unit, to estimate the time-averaged maximum node stress (N_{max}) and maximum link stress (L_{max}) as follows:

$$N_{max} = \frac{\sum_i \max_{v \in V_S} S(t^{(i)}, v)}{M}$$

$$L_{max} = \frac{\sum_i \max_{e \in E_S} S(t^{(i)}, e)}{M}$$

where $t^{(i)}$ is the time instance of the i'th sample and M is the total number of samples.

Each data point in the simulation results is obtained by running the simulation until 80,000 VN requests have been serviced. Furthermore, to reduce the effects of transient simulation results, we start to collect data after the first 40,000 VN requests. The results in this section are shown with their 95% confidence intervals. Note that the confidence intervals are very small, sometimes hardly visible on the plots due to the relatively large



(a) Maximum node stress

(b) Maximum link stress

(c) Average link stress

(d) Average substrate hop count

Figure 25: VN assignment performance under increasing offered load

number of samples.

4.5.2 VN assignment performance without reconfiguration

4.5.2.1 Effects of the offered load

In the first experiment, we increase the VN arrival rate λ while fixing the average VN lifetime and show the performance of the following static VN assignment algorithms:

- *Basic*: The basic VN assignment algorithm presented in Section 4.3.1
- *SubVN*: The basic VN assignment with the enhancement of subdividing the VN topology
- *Adaptive*: The SubVN based scheme with the adaptive optimization strategy where

the adaptive threshold $\eta = 1$

For comparison purposes, we also show results of the least-loaded algorithm (*least-load*) that treats link stress optimization and node stress optimization separately by selecting the least loaded substrate nodes as the virtual nodes and connecting them using the shortest distance path algorithm.

Since there is no blocking in the system (*i.e.,* VN requests are never rejected for service), both L_{max} and N_{max} grow linearly with λ (as shown in Figure 25). Note that under a fixed λ, the arrival rate of VN nodes is fixed. Therefore, the average substrate node stress is the same for any VN assignment algorithm. Figure 25(a) shows that the least-load, basic and subVN based schemes can all achieve similar N_{max} which is very close to the average node stress (shown in dashed line in Figure 25(a)). Since N_{max} could not be lower than the average node stress, this result indicates that all of the above three algorithms achieve near optimal node stress performance. The adaptive scheme has higher N_{max} since it trades off node stress performance for link stress performance.

The link stress performance in Figure 25(b) shows that the least-load scheme performs poorly in terms of L_{max}. The basic VN assignment scheme effectively reduces L_{max} by more than 30%. The subVN based scheme further reduces L_{max} by subdividing the VN topology. The adaptive scheme has the best link stress performance and its L_{max} is about 65% lower than the least-load scheme.

Results of the average link stress and average substrate hop count for virtual links are given in Figure 25(c) and Figure 25(d) respectively. They clearly show that our VN assignment algorithms reduce the resources usage by using shorter substrate paths. The least-load scheme, in contrast, does not consider the link stress in node selection so the selected substrate nodes could be far away from each other. Therefore, the least-load algorithm has much higher average link stress and average substrate hop count. We also observe in Figure 25(d) that the average substrate hop count is fixed under different VN arrival rate. The reason is that all of our VN assignment schemes are based on the relative orders of node/link stress, NR, or node potential, and these relative orders do not change with the offered load.

From this experiment, we can see that the basic VN assignment algorithm is consistently better than the least-load algorithm such that it can achieve near optimal node stress performance with much lower and more balanced link stresses. The adaptive VN assignment algorithm effectively balances the link/node stress objectives and significantly reduces the maximum link stress from the basic VN assignment algorithm. Furthermore, the virtual network built by our VN assignment algorithms has better quality than the least-load algorithm since the average virtual link length in substrate hop counts is significantly lower.

4.5.2.2 Benefits of the subVN scheme

The next experiment focuses on the subVN based scheme. Figure 26 compares L_{max} of both the basic scheme and the subVN scheme when the average VN connectivity is increased from 0.2 to 1.0 and the VN arrival rate is fixed at 10 per time unit. Both curves appear to be linear to the average VN connectivity since the latter linearly determines the arrival rate of virtual links. The subVN scheme achieves consistently lower L_{max} than the basic scheme but the differences between these two reduces from about 33% to almost zero as the VN connectivity increases from 0.2 to 1. Recall that the key improvement of the subVN scheme over the basic scheme is to perform node selection based on actual VN topology instead of assuming full VN connectivity. Obviously, the above difference becomes less significant as the actual VN connectivity increases. This experiment indicates that the subVN algorithm is more effective for VNs with sparse connectivity. For the densely connected VNs, the basic scheme could be used instead to simplify the VN assignment process without losing much performance. For the rest of the chapter, we use the subVN scheme by default.

4.5.2.3 Effectiveness of adaptive optimization strategies

To demonstrate the effectiveness of adaptive optimization strategies in VN assignment, we use an R_N vs. R_L graph: During each sampling instance in the simulation, we plot a point in the R_N vs. R_L graph using the corresponding values of (R_N, R_L) as the coordinates. Therefore the location of the (R_N, R_L) point indicates its relative performance of node stress balancing and link stress balancing. Specifically, all points located above the line of $R_L = R_N$ achieve better node stress balancing than link stress balancing, while points

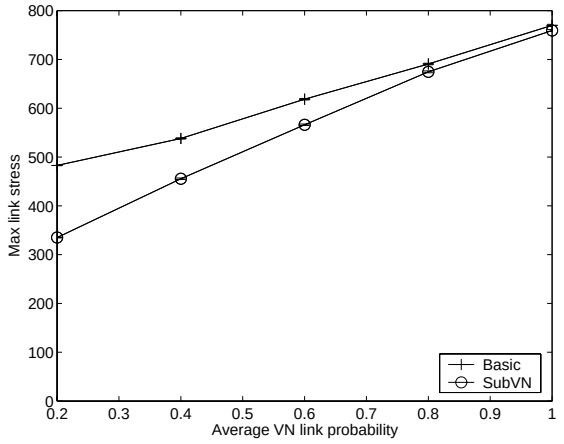


Figure 26: Effects of VN connectivity

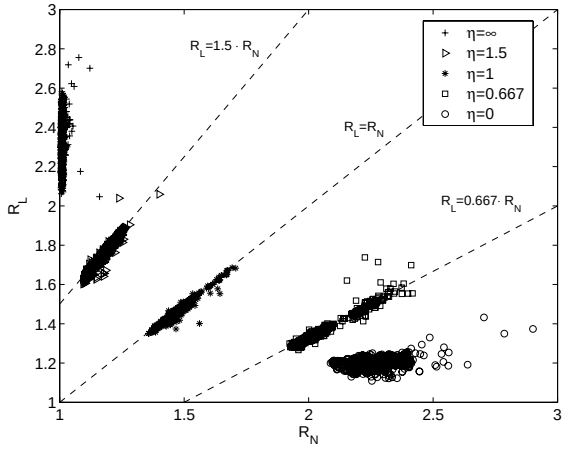
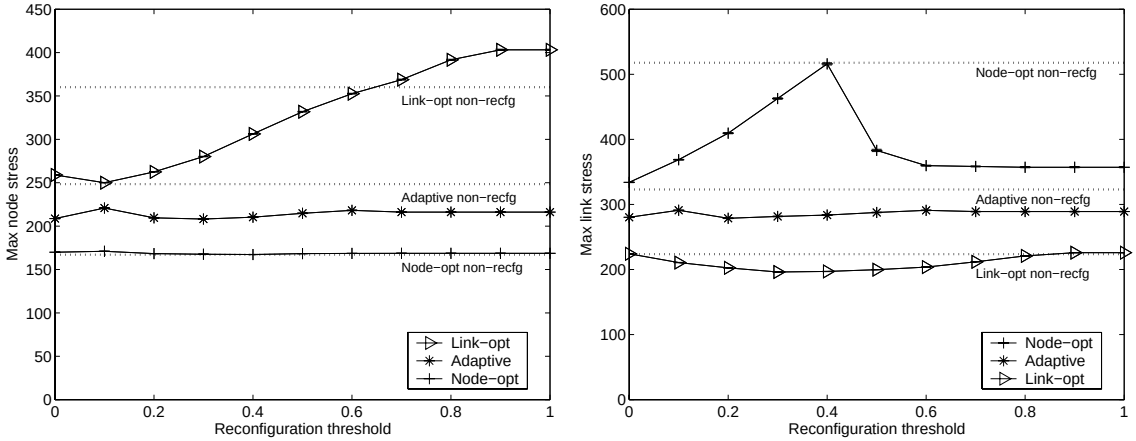


Figure 27: Tradeoffs between link stress and node stress performances

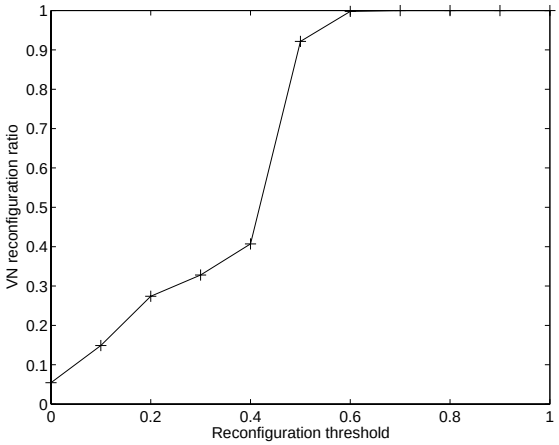
located below the line of $R_L = R_N$ have better link stress balancing performance. In this experiment, we fix the VN arrival rate at 15 per time unit and vary the adaptive threshold η to be $0, 0.667, 1, 1.5, \infty$ respectively and plot (R_N, R_L) points obtained from all sampling instances in Figure 27. Note that $\eta = 0$ and $\eta = \infty$ represent the pure link optimization (link-opt) and pure node optimization (node-opt) strategy respectively.

Figure 27 shows that VN assignment algorithms with different values of η form distinctive clusters in the R_N vs. R_L graph. Points of the node-opt strategy ($\eta = \infty$) are all located near the line of $R_N = 1$, indicating that the node-opt strategy achieves the optimal node stress balancing. However, it has very unbalanced link stress than all other cases since its points have the highest R_L values. In contrast, the link-opt strategy ($\eta = 0$) has the

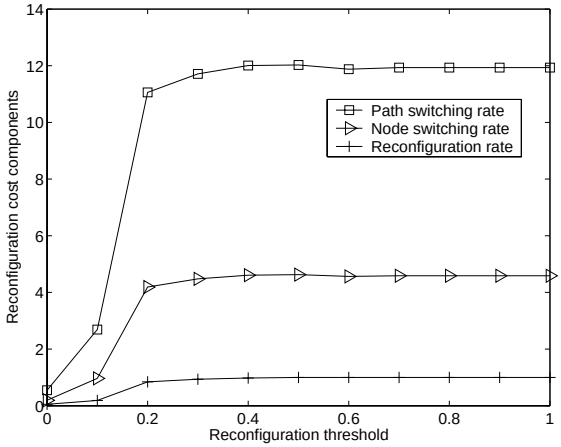


(a) Maximum node stress

(b) Maximum link stress



(c) Reconfiguration ratio (node-opt)



(d) Reconfiguration costs

Figure 28: Performance vs. reconfiguration threshold

lowest R_L values and therefore has the best link stress balancing performance. All intermediate values of adaptive threshold η result in different tradeoffs between node and link stress performance: As η increases from 0 to ∞, the VN assignment algorithm puts more emphasis on node stress optimization and less on link stress optimization.

Furthermore, points of $\eta = 0.667, 1, 1.5$ are closely located along their corresponding reference lines of $R_L = 0.667R_N$, $R_L = R_N$ and $R_L = 1.5R_N$ respectively, except for a small number of outlying points that occur during the transients. This indicates that the adaptive threshold η is able to effectively control the exact tradeoff between node and link stress balancing. For the rest of this section, we will only show the results of three representative optimization strategies: link-opt ($\eta = 0$), node-opt($\eta = \infty$) and adaptive ($\eta = 1$).

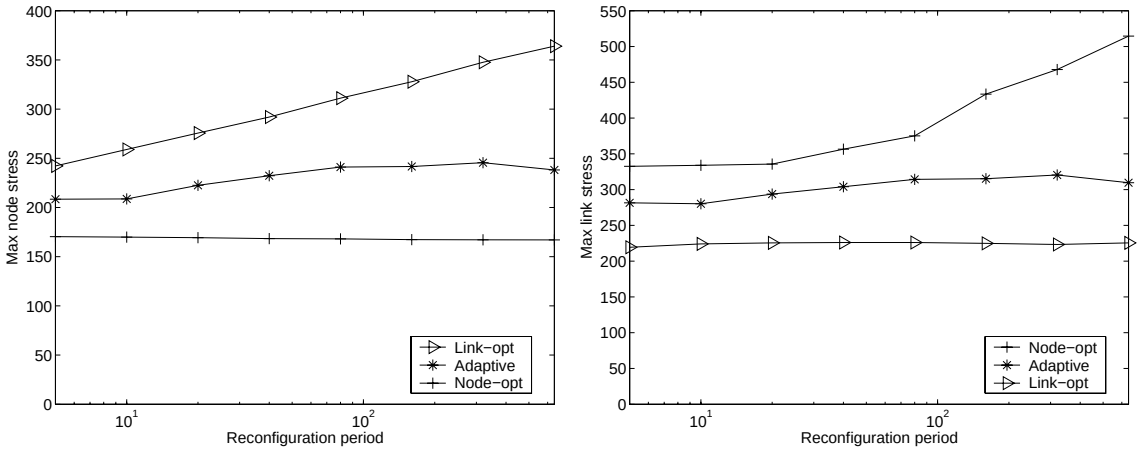
4.5.3 VN assignment performance with reconfiguration

Experiments in this subsection examine the performance of the selective VN reconfiguration algorithm. We choose the same value for both the marking period τ_M and per VN reconfiguration period τ_R^i so that critical VNs could have a chance to reconfigure promptly. Note that although their periods are the same, the marking and reconfiguring events are not synchronized since the latter is determined by the arrival time of each individual VN.

4.5.3.1 Effects of reconfiguration threshold

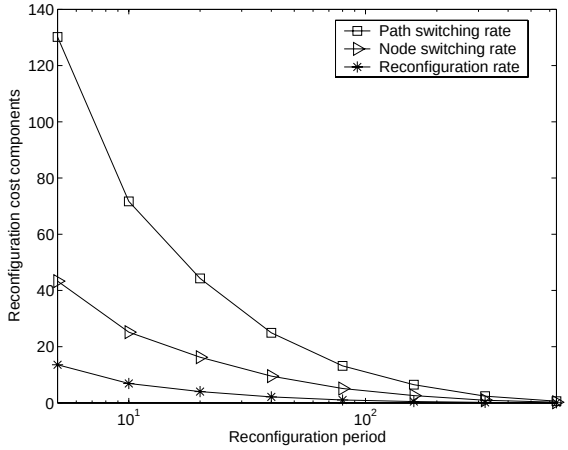
We first examine the effects of the reconfiguration threshold θ. Figure 28(a) and 28(b) show the performance the VN reconfiguration with node-opt, adaptive strategies as θ is increased from 0 to 1. Note that $\theta = 0$ means only VNs using the highest stressed substrate nodes/links are allowed to reconfigure while $\theta = 1$ means all VNs are allowed to reconfigure. Results for the corresponding non-reconfiguration schemes are also shown in dotted lines.

Our first observation is that VN reconfiguration generally achieves better L_{max} and N_{max} performance than the corresponding non-reconfiguration schemes. Figure 28(a) shows that N_{max} of the link-opt reconfiguration scheme is significantly lower than its non-reconfiguration counterpart when θ is small. When θ increases from 0 to 0.1, N_{max} of the link-opt scheme



(a) Maximum node stress

(b) Maximum link stress



(c) Reconfiguration cost

Figure 29: Performance vs. reconfiguration period

decreases slightly since more critical VNs are allowed to reconfigure. When θ is further increased, more non-critical VNs are reconfigured and take up substrate resources that could otherwise be used by critical VNs to reduce N_{max}. Therefore, N_{max} eventually increases as θ increase. The adaptive strategy demonstrates similar patterns but with a much smaller fluctuation magnitude since its node stresses are more balanced. Reconfiguration of the node-opt scheme has little effects on N_{max} performance since the node-opt scheme always achieves near optimal node stress performance.

In terms of the L_{max}, Figure 28(b) shows a more complicated picture for the node-opt scheme. Its L_{max} increases with θ when $\theta \leq 0.4$ since more non-critical VNs are competing with critical VNs. However, L_{max} has a sudden drop as θ increase from 0.4 to 0.5. To

understand this phenomena, Figure 28(c) gives the average ratio of VNs that perform the reconfiguration in the node-opt strategy. It shows that the percentage increases from 45% to more than 90% when θ increase from 0.4 to 0.5. This indicates that there are about 50% of the VNs that are only located on most lightly loaded substrate links and reconfiguring them would make more room to accommodate critical VNs to reduce L_{max}. Therefore, when $\theta \geq 0.5$, almost all VNs are allowed to reconfigure and L_{max} is reduced.

Figure 28(d) shows three reconfiguration cost components (defined in Eq (16)) and all of them increase with the reconfiguration threshold θ. The changes are especially dramatic for small θ values. Therefore, a small θ should be used to keep reconfiguration cost low. Fortunately, as we have seen in both Figure 28(a) and and 28(b), having a small θ could achieve most of the benefits of dynamic reconfiguration as long as we only critical VNs to reconfigure. The appropriate value of θ should be determined by the substrate topology and the resulting stress distribution. We set $\theta = 0$ in the rest of the chapter since it gives the best overall performance for our simulation topology.

4.5.3.2 Effects of reconfiguration period

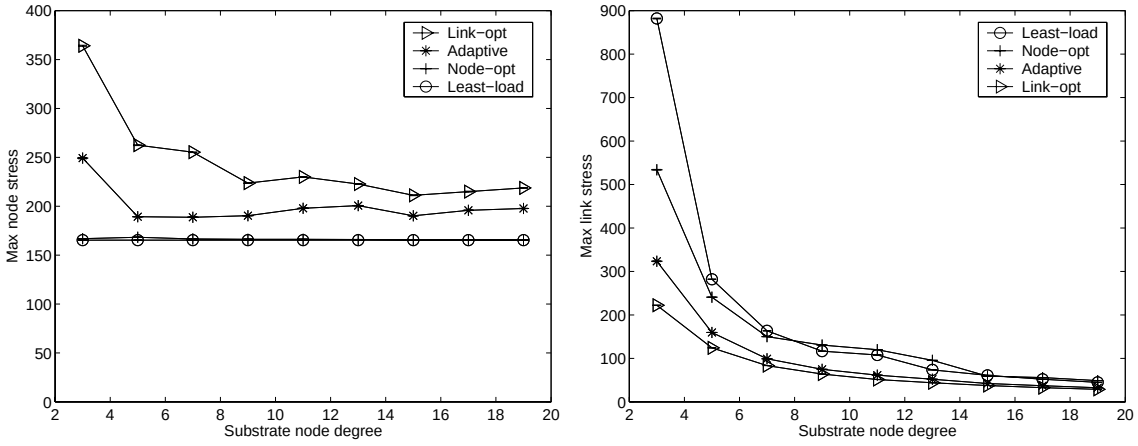
Another important parameter of the VN reconfiguration algorithm is the reconfiguration period τ_R, which determines the frequency of reconfiguration. Figure 29 gives the performance of the reconfiguration algorithm when τ_R is increased from 5 to 640 time units. Figure 29(a) shows that N_{max} of both the link-opt and adaptive strategies increases with τ_R. The node-opt strategy is not sensitive to τ_R since it can always achieve near optimal node stress performance. Figure 29(b) shows that L_{max} of both the node-opt and adaptive strategies increases with τ_R. Therefore, having a small τ_R will achieve better overall performances. In terms of reconfiguration cost, however, having very small reconfiguration period is not desirable since it would result in excessively high costs as indicated by Figure 29(c), which shows the reconfiguration cost components for the adaptive strategy. The cost curves of the other two optimization strategies are similar. Considering the performance and cost together, $\tau_R = 10$ is a good choice since it can achieve most of the benefits of reconfiguration without excessively high cost.

4.5.4 Effects of different network settings

The final set of the experiments examines the performance of VN assignment algorithms under a wide range of network settings. Due to space limitation, we only show results for VN assignment without reconfiguration.

4.5.4.1 Effects of substrate network connectivity

Figure 30 shows the VN assignment performance when the average substrate node degree is increased from 3 to 19. We observe that both N_{max} and L_{max} decrease as the substrate network connectivity increases and the changes in L_{max} are more dramatic. Furthermore, the absolute performance differences among different VN assignment schemes are reduced as the substrate network becomes more densely connected.

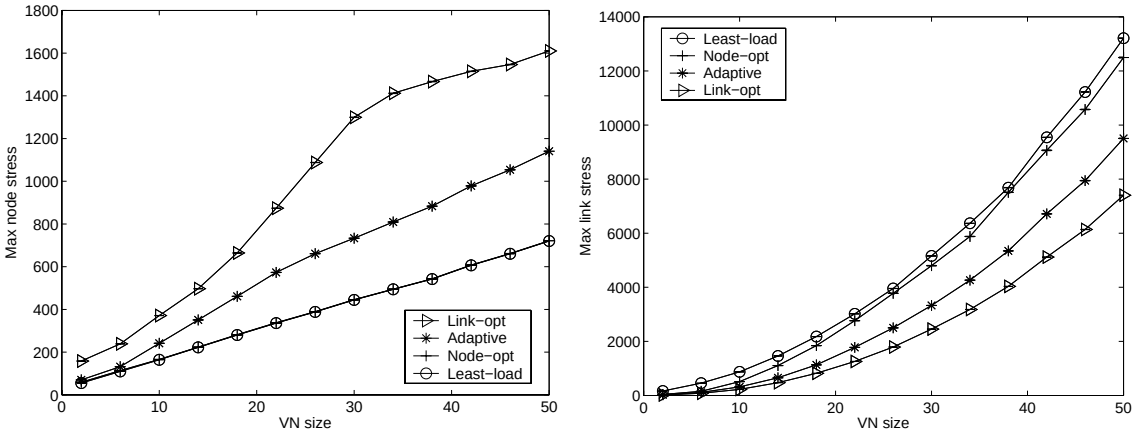


(a) Maximum node stress

(b) Maximum link stress

Figure 30: Performance vs. substrate connectivity

In terms of N_{max}, increasing the substrate connectivity means more substrate nodes become close to each other and, therefore, the node selection scheme will not concentrate only on a small number of substrate nodes. As a result, the node stress becomes more balanced. In terms of L_{max}, when the substrate network is more connected, shorter paths can be found between substrate nodes and the effects of node selection on link stress balancing become less significant. Therefore, when the substrate network is densely connected, even the naive least-load algorithm achieves acceptable link stress performance. As a summary, this experiment indicates that the proposed VN assignment scheme is more effective when



(a) Maximum node stress (b) Maximum link stress

Figure 31: Performance vs. VN size

the substrate network has sparse connectivity.

4.5.4.2 Effects of VN size

This experiment studies the performance of VN assignment algorithms under different VN sizes. Figure 31 shows the results when the maximum VN size increases from 2 to 50. Note that the substrate network is fixed at 100 nodes, increasing VN sizes makes the node stress balancing easier since larger VNs are less sensitive to their node selection. As an extreme example, if the VN size is the same as the total number of substrate nodes, then all algorithms achieve perfect node stress balancing. This explains the reduced N_{max} increasing rate for both the link-opt and adaptive strategies at large VN sizes in Figure 31(a).

In terms of L_{max}, both the adaptive and link-opt strategies have significantly better performance than the least-load scheme. The node-opt strategy also performs better than the least-load scheme, but the differences are small especially for larger VNs. Again, this is because for larger VNs, the effects of node selection are less important, and therefore the node-opt strategy is more similar to the least-load scheme.

4.5.4.3 Multi-domain substrate network topology

To see the performance of the VN assignment algorithms under multi-domain substrate topology, we used a 124-node 510-link transit-stub substrate topology generated by the GT-ITM tool instead of the random topology. The substrate topology has a single 4-node

transit domain and each transit node is attached by three 10-node stub domains. Figure 32 shows that L_{max} of the least-load scheme in this scenario is almost 8 times higher than the basic VN assignment algorithm. The differences are significant as compared to the corresponding results on random substrate topology (Figure 25(b)). This is because the substrate nodes with the lowest stress may be scattered in different domains and the least-load algorithm would select them for a single VN request. As a result, the transit-stub links become highly loaded because they are the only links that connect different domains. The proposed VN assignment scheme, in contrast, avoids hot spots by allocating resources for a single VN within the same domain.

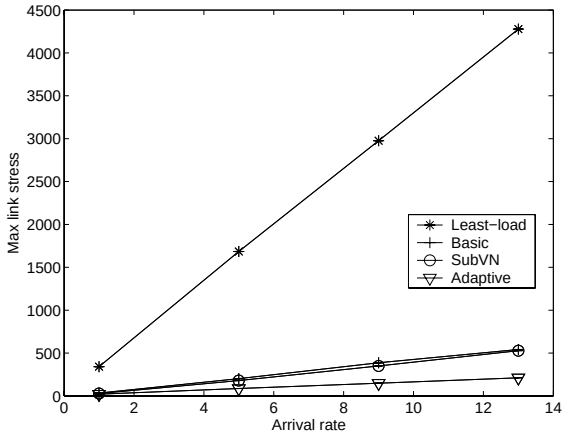


Figure 32: Maximum link stress (transit-stub substrate topology)

4.6 Summary

Network virtualization is a promising way to de-ossify the current Internet by providing a shared platform for a variety of new network services and architectures. A major challenge in building the diversified Internet is to perform efficient and on-demand VN assignment.

In this chapter, we developed a basic scheme for VN assignment without reconfiguration and use it as a building block for all other advanced algorithms. Subdividing heuristics and adaptive optimization strategies are presented to further improve the performance. We also developed a selective VN reconfiguration scheme that prioritizes the reconfiguration for the most critical VNs.

We evaluate the performance of the proposed VN assignment algorithms through extensive experiments and our findings are summarized as follows:

- The basic VN assignment algorithm performs consistently better than the least-load algorithm.
- The benefits of subdividing the VN topology are more significant when the VN topology is sparse.
- The advantage of proposed VN assignment algorithms is more prominent when the substrate network is sparsely connected.
- The proposed VN assignment algorithms can effectively avoid hot spots or congestion in the substrate network (such as the transit-stub links).
- For VN reconfiguration, reconfiguring only a subset of the existing VNs that are most critical achieves most of the benefits of dynamic reconfiguration while keeping a low cost. Small reconfiguration threshold and reconfiguration period should be used but their exact values should be determined by the specific substrate topology to achieve the desired tradeoff between performance and cost.

CHAPTER V

AUTOMATIC OVERLAY NETWORK CONFIGURATION IN PLANETLAB WITH NETFINDER

PlanetLab has been widely used in the networking community to test and deploy user-defined overlays. Serving as a meta testbed to support multiple overlay networks, PlanetLab has significantly lowered the barriers to build new overlays. However, PlanetLab users always face the problem of selecting a set of nodes and interconnecting them to form the desired overlay network. Unfortunately, such a task is usually carried out manually by individual users and sometimes in an ad-hoc manner. In this chapter, we develop NetFinder, an automatic overlay network configuration tool to efficiently allocate PlanetLab resources to individual overlays. NetFinder continuously monitors the resource utilization of PlanetLab and accepts a user-defined overlay topology as input and selects the set of PlanetLab nodes and their interconnection for the user overlay. Experimental results indicate that overlay networks constructed by NetFinder have more stable and significantly higher bandwidth than alternative schemes and near optimal available CPU.

5.1 Overview

PlanetLab has been widely used in the networking community to test and deploy new network technologies [58]. It can serve as a testbed for overlay networks. Research groups are able to request a PlanetLab slice in which they can experiment with a variety of wide-area networks and services, such as file sharing, content distribution networks, routing and multicast overlays, QoS overlays, and network measurement tools. One problem faced by PlanetLab users is selecting a set of PlanetLab nodes and interconnecting them to form the desired overlay network.

PlanetLab also serves as a meta testbed on which multiple, more narrowly-defined virtual testbeds can be deployed. For example, the "Internet-in-a-Slice"($IIAS$) service aims at

recreating the Internet's data plane and control plane in a PlanetLab slice [36]. Network researchers can use this infrastructure to experiment with modifications and extensions to the Internet's protocol suite. Currently, IIAS does not provide resource discovery and overlay assignment services and the overlay configuration has to be given explicitly.

Having an automated overlay network assignment service for PlanetLab is important for two reasons. From the PlanetLab operator's perspective, efficient assignments would result in better resource utilization so that the PlanetLab could accommodate more user overlays with limited resources and avoid hot spots or congestion. From the overlay user's perspective, having good assignment service would provide overlay users better choices in building their overlays by avoiding congested links/nodes.

Unfortunately, such a task is usually carried out manually by individual users and sometimes in an ad-hoc manner. Manual configuration is time consuming (especially for networks) and prone to human error. It is not efficient even for small overlay networks since the underlying PlanetLab network has hundreds of nodes and its state fluctuates over time due to failures, congestion and load variation. In this work, we are motivated by the above considerations and focus on the on-demand overlay network assignment problem: Upon the arrival of a user-defined overlay topology and associated overlay resource requirements, find the set of PlanetLab nodes and their interconnections to satisfy the user request.

The set of physical resources provided by PlanetLab to support multiple overlays can be divided into two categories: node resources and link/path resources. Node resources such as memory, disk space, number of active slices, and available CPU are all related to a particular PlanetLab node. They enable individual overlays to compute, modify and store their data. In contrast, link/path resources enable an overlay network to send data from one node to another. Common metrics for link/path resources include available bandwidth, delay and loss rate, which indicate how well the user can send data within its overlay.

A recent study reported that available resources vary significantly in PlanetLab, suggesting that wise placement of application instances can be beneficial [54]. Among different node resource metrics, available CPU is the most important one since most users will suffer if the CPU resources are poorly utilized. Unlike the link performance metrics such as delay,

loss rate or TCP throughput, available bandwidth directly indicates the extra amount of traffic that can be carried by the link before it becomes congested. Based on these considerations, we focus on the CPU usage and available bandwidth. However, our service model can be easily extended to support other metrics.

There are two challenges in providing an automatic overlay network assignment service: the lack of PlanetLab bandwidth monitoring services and the difficulties in developing a good overlay network assignment algorithm. Despite the growing literature and tools for end-to-end bandwidth measurement, many tools have practical limitations and do not work well in PlanetLab. Even for tools that work within PlanetLab such as Pathload [38] and Iperf [3], running pair-wise measurements between hundreds of Planetlab nodes is a challenging task. Another challenge comes from the dual objective of the overlay assignment. Specifically, upon the arrival of a overlay request, we want to achieve load balancing on both PlanetLab nodes and links. A special case of the this problem can be formulated as an unsplittable flow problem which is NP-hard. Therefore, the overlay assignment problem is intrinsically difficult and heuristics will be used to solve the problem.

In this chapter, we develop NetFinder, an automatic overlay network configuration tool in PlanetLab. This tool continuously collects information about the resource utilization of PlanetLab and accepts a user-defined overlay topology as input. NetFinder then performs overlay network assignment by selecting the set of PlanetLab nodes and their interconnection for the desired user overlay.

5.2 Network architecture

PlanetLab is composed of a number of nodes connected to the Internet at different locations. Therefore, there is an end-to-end native Internet path connecting each pair of PlanetLab nodes. From the view point of PlanetLab users, the only resources available are those at individual Planetlab nodes and along paths between PlanetLab nodes. Network situation on all other links and nodes are transparent to PlanetLab users. Therefore, the PlanetLab network itself can be viewed as a full-mesh overlay on top of the native Internet.

The PlanetLab network in turn serves as a shared substrate to support multiple user-defined overlays. Each user would treat the underlying PlanetLab network just like any physical network.

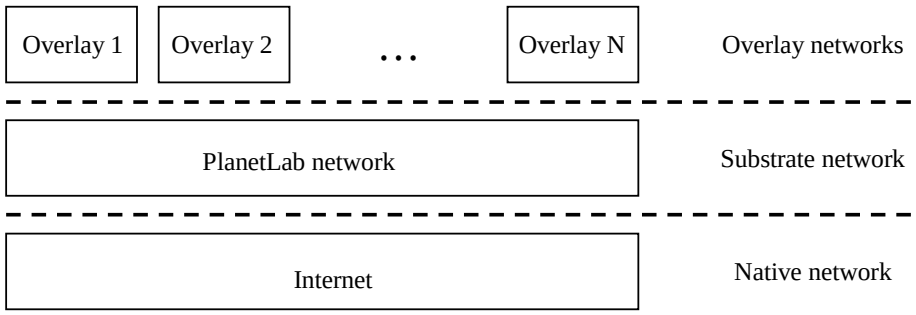


Figure 33: PlanetLab network layered architecture

Based on the above consideration, we model the PlanetLab overlay network configuration using three layers of network: a native Internet, a PlanetLab substrate network and a number of user-defined overlay networks. The lowest layer is the native Internet composed of native routers and links. On top of that, there is a common PlanetLab substrate network, composed of PlanetLab nodes and the interconnection between them. Finally, a number of user-defined overlays are built on top of the PlanetLab substrate such that each overlay node is a substrate node and each overlay link is a substrate path, which could, in turn, be a concatenation of a set of Internet paths. In this work, we assume that all PlanetLab nodes are already deployed and their interconnection are determined by native IP routing mechanisms, and focus only on issues between the substrate layer and user overlays.

The flowchart of NetFinder is illustrated in Figure 34. In the background, the system periodically collects the PlanetLab resource utilization information including both available bandwidth data and available CPU data from external sources. NetFinder then accepts user-defined overlay requests. When the user requests include constraints such as the minimum CPU or minimum bandwidth requirements, infeasible parts of the PlanetLab network are removed through preprocessing. Finally, the overlay assignment algorithm is used to calculate the overlay configuration.

The rest of this section discusses each component of the system in detail.

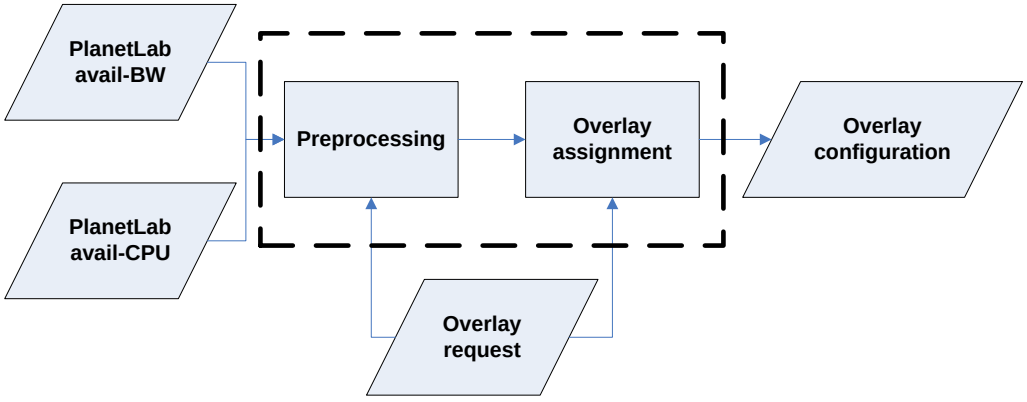


Figure 34: System diagram

5.2.1 Overlay network request format

To setup an overlay network in the PlanetLab, a user submits a request specifies the topology of the overlay network and a set of resource constraints. An example to setup a 4 node 8 link overlay network is shown in Table 2. The request also indicates that each overlay link should have minimum available bandwidth of 10Mbps and each selected PlanetLab node should have at least 20% free available CPU resource.

Table 2: Overlay request format

```
# NUM_NODES, MIN_BW MIN_CPU
  4, 10 20

# LINKS: SRC DEST
  0 1
  1 0
  0 2
  2 0
  0 3
  3 0
  1 2
  2 1
```

5.2.2 PlanetLab substrate performance monitoring

NetFinder collects measurement results from other reporting services. It then preprocesses and stores the resulting data into its own database. The current version of NetFinder obtains node CPU usage information by periodically polling the CoMon daemon on each PlanetLab node. This probing also serves as a failure detection mechanism, where NetFinder determines that a node is down if the probe times out.

The bandwidth measurement in PlanetLab is a challenging problem due to the scale of the network and practical limitations of various tools. NetFinder uses the bandwidth measurement result from the S^3 project, which reports the all-pair available bandwidth measurement results of 400^+ PlanetLab nodes every four hours. NetFinder then processes the raw measurement results by removing all inactive nodes and links and keeping only relevant bandwidth and CPU data.

Although NetFinder relies on the PlanetLab measurement results, it is not restricted by any specific tools or reporting services. We choose S^3 results since it is the only all-pair bandwidth measurement service available for the large scale PlanetLab network. However, NetFinder has the flexibility to adapt to measurement results of new techniques when they become available in the future. The only thing that needs to be done is to convert the format from the raw measurement data to NetFinder's internal data format.

5.2.3 Overlay network assignment

NetFinder's overlay network assignment is based on our recent work on network virtualization. The problem of assigning PlanetLab nodes/paths to the overlay network is similar to the virtual network (VN) assignment problem [91]. However, to be able to work with PlanetLab, we need the following modifications and enhancements:

- Using bandwidth as the Link-state metrics instead of substrate link stress: Unlike the scenario of VN assignment in [91], where stress is used to model the utilization of the substrate node and substrate link, the meaning of stress in the PlanetLab context is not obvious.

- Constraints in overlay network assignment: The overlay assignment algorithm should consider a number of constraints in making the assignment decision. Basic constraints include bandwidth constraints (*e.g.,* the available bandwidth of the overlay link should exceed 10Mbps) and CPU constraints.

Based on the above consideration, we modify our original VN assignment algorithm as follows:

5.2.3.1 Preprocessing

After obtaining the measurement results, given an overlay request with bandwidth or CPU constraints. The algorithm first removes all nodes and links that do not have enough resources.

5.2.3.2 Cluster center localization

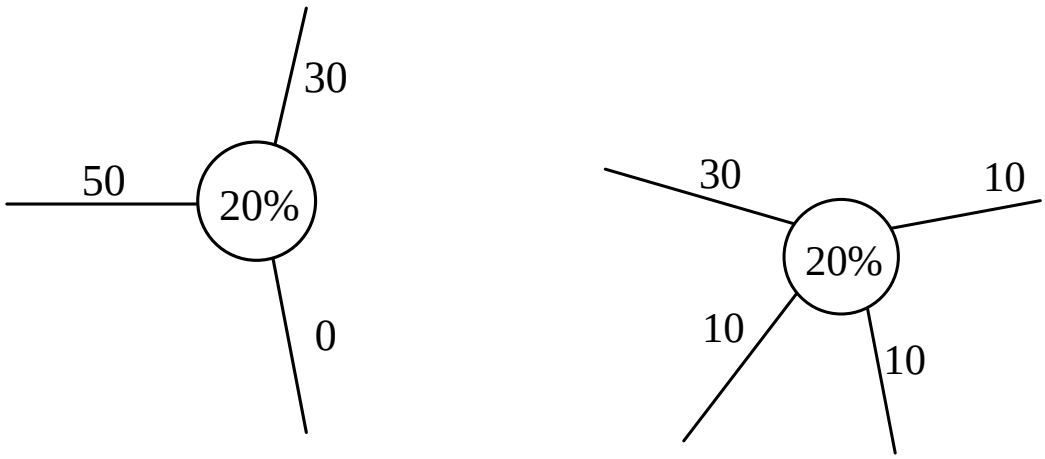
In the PlanetLab overlay configuration, we use the more realistic metrics such as CPU usage and available bandwidth instead of the abstract ideas of node/link stress to represent the resource availability in the substrate network.

Specifically, we modify the definition of neighborhood resource availability (NR) in [91] as follows:

$$\mathrm{NR}(t,v) = C(t,v) \cdot \sum_{e \in L(v)} B(t,e) \tag{17}$$

where $C(t,v)$ is the available CPU at substrate node v, $B(t,e)$ is the available bandwidth on substrate link e, and $L(v)$ is the set of substrate links directly connected to substrate node v.

The above definition of NR captures both the node resources and neighboring link resources in the sense that a substrate node with a high NR means that both the node itself and some of its directly connected links are lightly loaded. The cluster center, which is the first selected substrate node, is, therefore, identified as the substrate node with the highest NR value. In the example shown in Figure 35, the NR for the substrate node in Figure 35(a) and 35(b) are 16 and 12 respectively. Therefore, the former substrate node is preferred even though it has a lower node degree.



(a) (b)

Figure 35: Neighborhood resource availability, numbers beside links and nodes represent the available bandwidth and CPU respectively

5.2.3.3 Substrate node selection

After identifying the cluster center, the next step is to select the rest of the substrate nodes for the overlay request. We redefine the distance of a single substrate path p at time t as follows:

$$d(t,p) = \sum_{e \in p} \frac{1}{B(t,e) + \delta_L} \tag{18}$$

where δ_L is a small positive constant to avoid dividing by zero in computing the distance. And the distance between two substrate nodes u and v is defined as the minimum distance of all paths between them, *i.e.,*

$$D(t,(u,v)) = \min_{p \in P_S(u,v)} d(t,p) \tag{19}$$

Since the overlay topology could be arbitrary, there could be an overlay link between any pair of overlay nodes. Therefore, to minimize the overall distance, the weight of an overlay node is set to be the summation of distances to all previously assigned overlay nodes. Combining the distance and node stress, we use the following *node potential* as the criteria for substrate node selection:

$$\pi(t,v) = \frac{\sum_{u \in V_A} D(t,(v,u))}{C(t,v) + \delta_N} \tag{20}$$

where V_A is the set of selected substrate nodes for the same overlay and δ_N is a small positive constant to avoid dividing by zero. The set of substrate nodes with the minimum potential are then selected.

After all substrate nodes are determined, we need to map the overlay nodes into the selected substrate nodes such that overlay nodes with higher degree are mapped to substrate nodes with higher NR. The intuition behind this is that overlay nodes with higher degree will setup more overlay links. All of these overlay links will go through some of the substrate link in the neighborhood of the corresponding substrate node. Therefore,the above matching tends to reduce the congestion and balance the link load.

5.2.3.4 Substrate path selection

The last step of the overlay assignment scheme is to connect the selected substrate nodes based on the overlay topology. This is a standard routing problem and we use the shortest-distance path algorithm [49] again to select the path with minimum distance (defined in Eq. (18)) between corresponding substrate nodes. Therefore, it is consistent with Eq. (19) in our node selection process. The detailed overlay assignment algorithm is given in Algorithm 6.

Algorithm 6 Overlay assignment algorithm (Upon the i'th overlay arrival at time a_i)

INPUTS:
$G_s = (V_S, E_S)$: substrate topology;
$C(a_i^-, v), \forall v \in V_S$: current available CPU;
$B(a_i^-, e), \forall e \in E_S$: current available bandwidth;
$G_v^i = (V_V^i, E_V^i)$: Overlay topology;
OUTPUTS:
$f_N^i(\hat{v}), \forall \hat{v} \in V_V^i$ and $f_L^i(\hat{e}), \forall \hat{e} \in E_V^i$

$V_A = \emptyset$ {Note: V_A is the set of selected substrate nodes}
$V_A = V_A \cup \{\arg \max_{v \in V_S} \mathrm{NR}(a_i^-, v)\}$
for $i = 2, \ldots, |V_V^i|$ **do**
$V_A = V_A \cup \{\arg \min_{v \in V_S - V_A} \pi(a_i^-, v)\}$
end for
Assign nodes in V_A to overlay nodes such that
$\mathrm{NR}(f_n^i(\hat{u})) \geq \mathrm{NR}(f_n^i(\hat{v}))$, *iff* $\mathrm{Degree}(\hat{u}) \geq \mathrm{Degree}(\hat{v})$
Find the shortest-distance path for all overlay links

5.3 Implementation

We implement the above service to include all 412 nodes in the S^3 data base. Latest NetFinder results detects 352 alive nodes and 68,966 substrate links with bandwidth measurement results.

The high level service model of NetFinder is shown in Figure 36, which is composed of three entities: a PlanetLab network (composed of nodes and paths connecting them) served as the substrate for all user-defined overlays, a central database responsible for collecting measurement data of PlanetLab and answering overlay setup requests, and users submitting their requests to setup overlay networks. Events in the centralized model evolve as follows: 1) Each PlanetLab node perform periodic performance measurement of its own as well as paths leading to the neighboring PlanetLab nodes, this information is stored at each PlanetLab node as local data [1]. 2) The centralized database periodically contacts each PlanetLab node to get a copy of the local measurement data and merge them into a global view of the current PlanetLab network status. 3) The software at PlanetLab user sends a request for the current substrate states. 4) Central database returns the data to the user software. 5) The user software calculates the assignment and generates suitable outputs. 6) IIAS script is launched to start the overlay network based on the output from step 5) [2].

5.4 Performance evaluation

To evaluate the performance of NetFinder from its user's perspective, we use the following metrics: 1) average overlay node available CPU, defined as the average available CPU of the selected PlanetLab nodes, and 2) average overlay link available bandwidth, defined as the available bandwidth along the path connecting the corresponding PlanetLab nodes. Since each overlay node is assigned to a PlanetLab node, the available CPU on the selected PlanetLab node represents the available computational resources for the corresponding overlay node. Therefore, these metrics are more relevant to individual user.

In the first experiment, we took a snapshot of the PlanetLab network and run NetFinder

[1] This step is not needed when NetFinder obtains performance data from third-party sources, such as S^3.
[2] This step is needed when we use IIAS to directly startup the overlay.

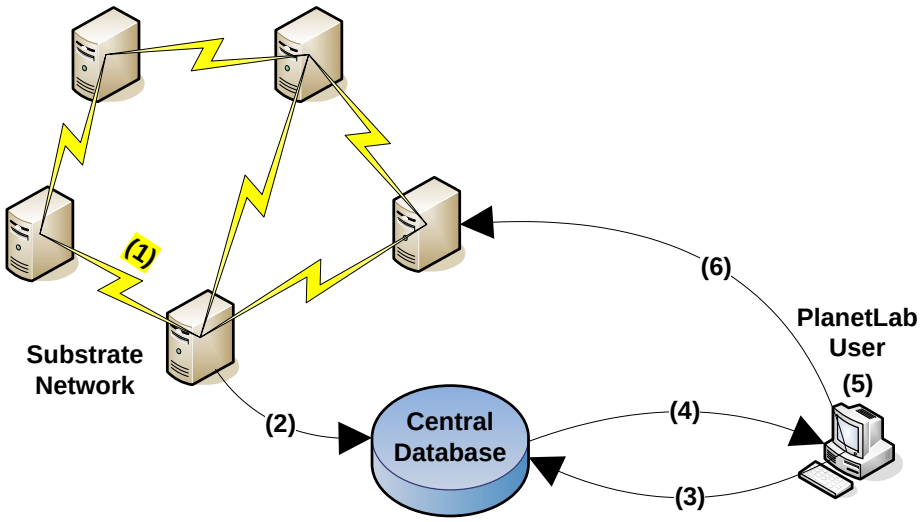


Figure 36: Centralized NetFinder service model

to assign 5,000 randomly generated overlay networks (with evenly distributed size from 2 to 40 and average link probability 0.5). Each overlay network is assumed to be given the identical PlanetLab snapshot. Therefore, this experiment evaluates the average performance of NetFinder on different overlay topologies. For comparison purposes, we also show the results for two alternative overlay assignment schemes: 1) Least load scheme and 2) random selection. The least-load scheme represents the scenario where node selection tools such as CoMon or SWORD is used to find the set of nodes with maximum available CPU. The selected nodes are then connected to form the overlay network. The random selection scheme represents the case where a user totally ignores the performance of PlanetLab and selects nodes randomly.

Figure 37 shows the cumulative distribution function (CDF) of single hop overlay path available bandwidth (*i.e.,* the direct overlay link available bandwidht). Comparing the results of NetFinder with least-load and random selection heuristics, we can clearly see that NetFinder significantly improves the available bandwidth. Specifically, more than 90% of the overlay links have the available bandwidth higher than $60Mbps$. In contrast, more than 50% overlay links assigned by the least-load scheme have available bandwidth lower than $60Mbps$. The random selection scheme performs slightly better than the least-load scheme since it spreads the load among all PlanetLab nodes. However, it is still significantly worse

than NetFinder.

A potential use of the overlay network is bandwidth based overlay routing, where data are sent from the source to the destination through the overlay path with the maximum available bandwidth. To show how assigned overlay networks perform under this scenario, we collected the widest overlay path between each pair of overlay nodes and show their CDF in Figure 38. The result has a similar pattern as the direct overlay link available bandwidth. But the different between these schemes are even more significant.

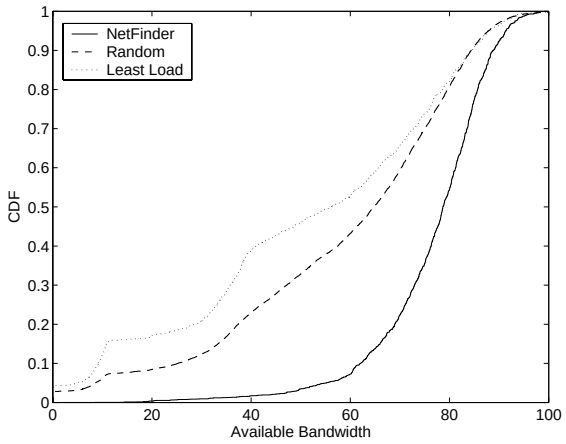


Figure 37: CDF of single hop overlay path available bandwidth

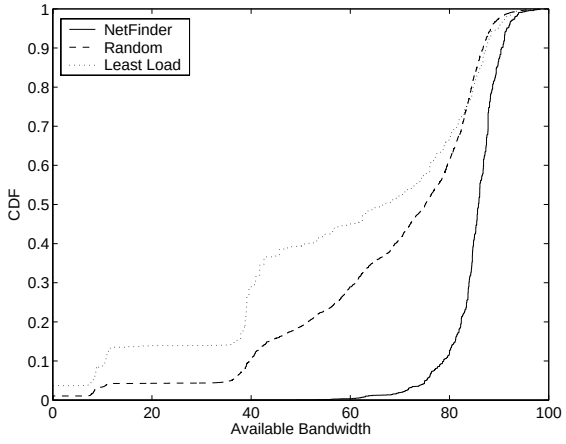


Figure 38: CDF of widest overlay path available bandwidth

Figure 39 shows the CDF of available CPU on overlay nodes. As expected, the least-load scheme has the best node CPU performance since it performs node selection exclusively on the node performance optimization [91]. However, the difference between NetFinder and

the optimal least-load scheme is small. This indicates that NetFinder can achieve near optimal available CPU performance.

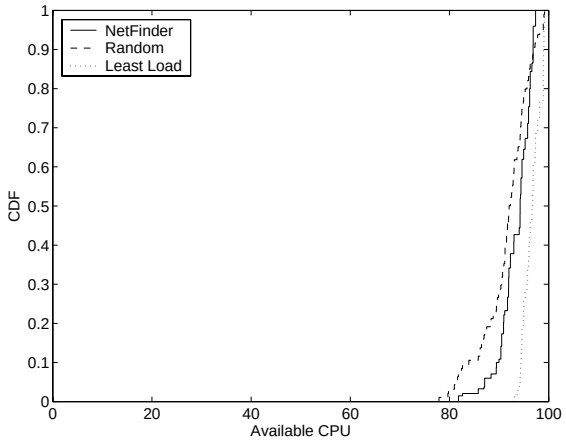


Figure 39: CDF of overlay node available CPU

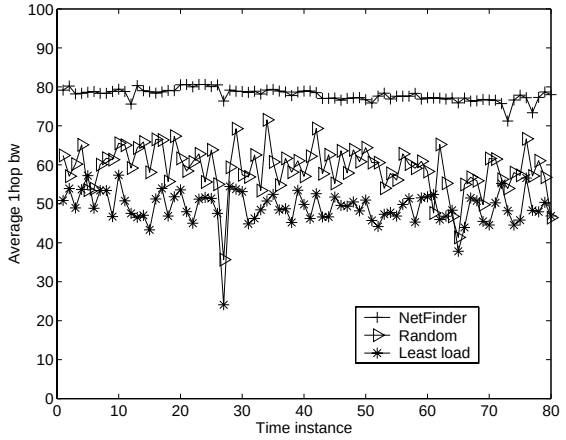


Figure 40: Average available bandwidth of single hop overlay path

The second experiment studies the performance of assigning a single overlay network at different time instances. We collect the PlanetLab measurement data during the week of February 13-17, 2006 and plot the results of assigning the same 20 node 176 link overlay network. Figure 40, 41 and 42 show the average available bandwidth of all 1-hop overlay paths (*i.e.,* direct overlay path without going through any intermediate overlay node), 2-hop overlay paths (*i.e.,* indirect overlay paths that go through exactly one intermediate overlay node), and widest overlay paths (*i.e.,* overlay paths with the highest available bandwidth) respectively. All these results show that NetFinder can allocate significantly higher available

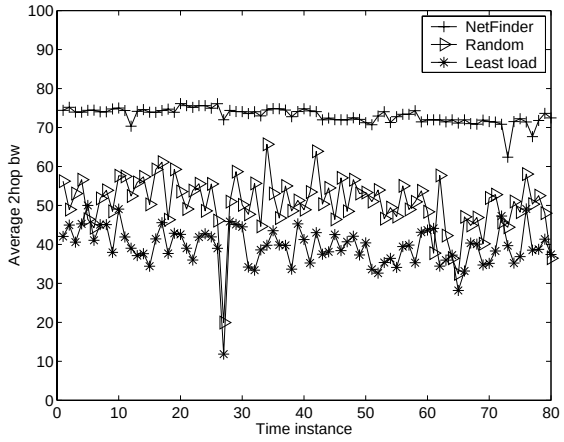


Figure 41: Average available bandwidth of 2-hop overlay path

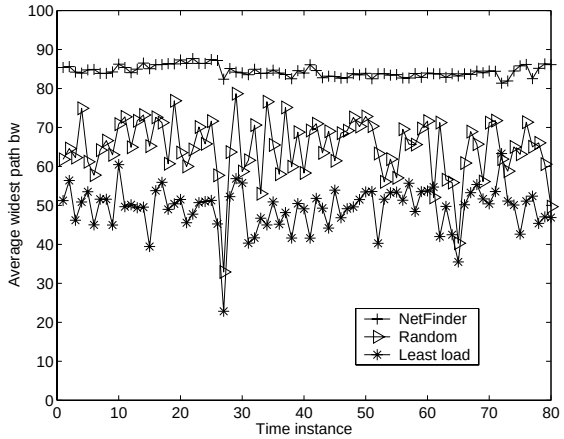


Figure 42: Average available bandwidth of the widest overlay path

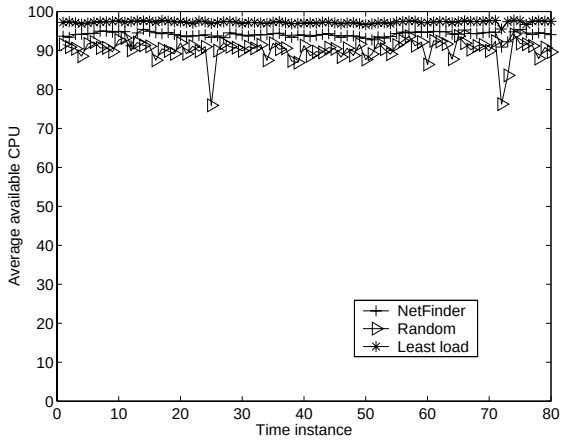


Figure 43: Average available CPU of overlay nodes

bandwidth than the other two schemes. Furthermore, the performance improvements are consistent throughout all time instances. Another interesting observation is that selecting the PlanetLab nodes at random and totally ignoring the network states, performs better than the least-load scheme in terms of available bandwidth. The reason is that the low loaded areas in the substrate network tend to be isolated, and therefore, they may be separated by some heavily loaded links. In terms of available CPU, the Least load scheme which is optimal outperforms NetFinder. However, the difference is marginal.

Another observation in this experiment is that the performance curves of NerFinder are much smoother than the other two schemes. This indicates that NetFinder is capable of achieving consistent performances in dynamic network situations. As a summary of this experiment, the results indicate that the NetFinder service can achieve significantly higher available bandwidth than the both least-load and random schemes.

5.5 Summary

PlanetLab has been widely used in the networking community to test and deploy user-defined overlays. One problem always faced by PlanetLab users is how to select a set of PlanetLab nodes to form a user-defined overlay. In this chapter, we develop NetFinder, an automatic overlay network configuration tool for PlanetLab. This tool continuously collects information about the resource utilization of PlanetLab and accepts a user-defined overlay topology as input. NetFinder then selects the set of PlanetLab nodes and their interconnection for the user overlay. Performance evaluation using realistic PlanetLab data demonstrates the advantage of NetFinder by comparing it with alternative heuristics or existing tools. The results indicate that NetFinder is highly effective in locating the available PlanetLab resources and avoiding overloaded nodes and links.

CHAPTER VI

MULTI-HOMED OVERLAY NETWORKS - A HYBRID ARCHITECTURE FOR INTELLIGENT ROUTING

Multihoming and overlay routing are used, mostly separately, to bypass Internet outages, congested links and long routes. In this chapter, we examine a scenario in which multihoming and overlay routing are jointly used. Specifically, we assume that an Overlay Service Provider (OSP) aims to offer its customers the combined benefits of multihoming and overlay routing, in terms of improved performance, availability and reduced cost, through a network of multihomed overlay routers. We focus on the corresponding design problem, *i.e.,* where to place the overlay routers and how to select the upstream ISPs for each router, with the objective to maximize the profit of the OSP. We examine, with realistic network performance and pricing data, whether the OSP can provide a network service that is profitable, better (in terms of round-trip time), and less expensive than the competing native ISPs. Perhaps surprisingly, we find out that the OSP can meet all three objectives at the same time. We also show that the MON design process is crucial. For example, operating more than 10 overlay nodes or routing traffic through the minimum-delay overlay path, rarely leads to profitability in our simulations.

6.1 Overview

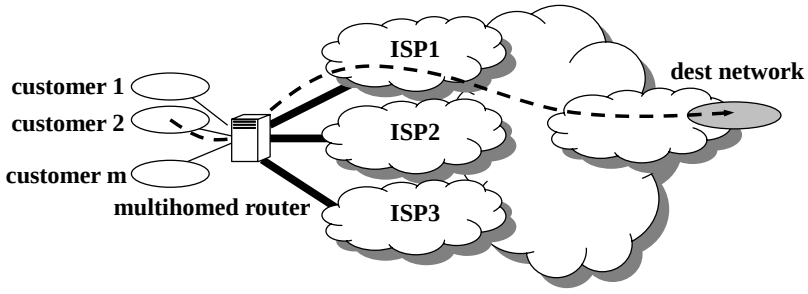
The most basic form of Internet access is *singlehoming*, where a stub network uses a single upstream ISP to reach all destinations. It has been shown that singlehoming can lead to poor availability and performance [9]. The single route from the source network to a destination network/prefix may not be always available, while routing policies and traffic engineering practices can (and often do) impose a heavy performance penalty on the resulting end-to-end performance [6].

To achieve improved reliability and performance, *multihoming* has become the mainstream service model for major content providers (see Figure 44(a)). In the more advanced form of this model, known as *intelligent route control*, the multihomed source network selects the upstream ISP for every significant destination prefix based on performance and cost considerations [1, 30].

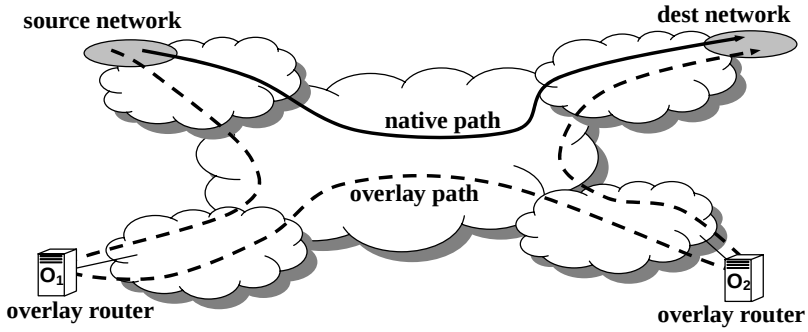
Another approach to improve end-to-end availability and performance is *overlay routing* (see Figure 44(b)). Here, the traffic between two networks is sent through one or more intermediate overlay nodes that are connected through IP tunnels [9, 48]. The advantage of overlay routing is that it typically provides a greater number of diverse paths to reach a destination network compared to the typical case of multihoming to 2-4 upstream ISPs [63]. On the other hand, overlay routing requires the deployment of a distributed routing/forwarding overlay infrastructure. A comparison of multihoming and overlay routing has been conducted in [7].

Given the previous two approaches, it is interesting to consider a scenario in which both models are used. Specifically, we envision a new type of Internet provider referred to as *Overlay Service Provider (OSP)* (to distinguish from an ISP) that attempts to offer its customers the combined benefits of multihoming and overlay routing in terms of improved performance, availability and reduced cost. The OSP operates a *Multihomed Overlay Network (MON)*, with each MON node being a multihomed router. MON nodes are placed at "key" Internet locations, mostly Internet Exchange Points (IXPs), and the OSP purchases Internet connectivity for each MON node from several locally present ISPs. An OSP customer can connect directly to a MON node if the former is collocated at the same IXP with that MON node. Major content providers are usually collocated at major IXPs to avoid the cost of leased lines. On the other hand, the OSP is responsible to route a customer's traffic with greater availability and higher performance than the customer's current *native* ISP. Note that this is similar to the InterNAP service and business model [1].

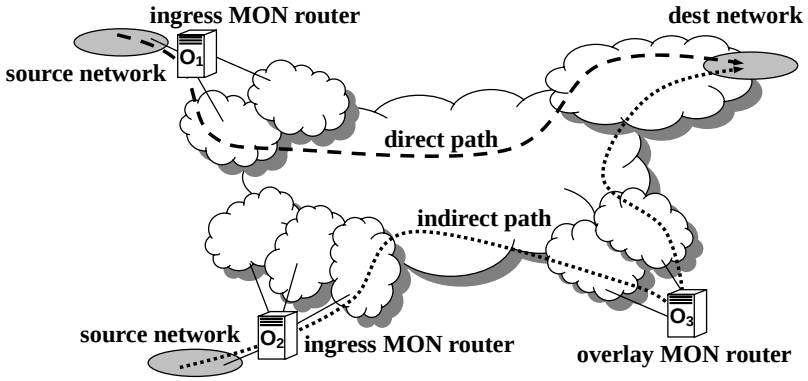
Furthermore, we envision that the OSP performs overlay routing, utilizing MON nodes as overlay routers. Based on the findings of [92], we limit the number of intermediate MON nodes in an overlay path to one. It is rarely the case that more intermediate nodes are



(a) Multihoming



(b) Overlay routing



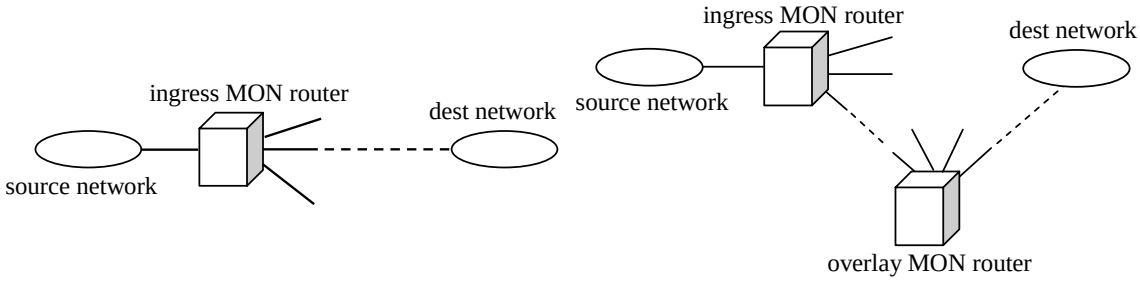
(c) MON architecture

Figure 44: Multihoming, overlay routing, and the Multihomed Overlay Network (MON) architecture.

needed to improve performance or availability significantly. Figure 44(c) shows that the MON network can utilize multihoming to form *direct paths* and overlay routing to form *indirect paths*. If a MON node is multihomed to K ISPs, there are K direct MON paths to choose from for each flow. With N MON nodes, the number of indirect MON paths for each flow increases to $K^2(N-1)$.

It is interesting that an OSP is "an Internet provider that does not own a network", in the sense that the OSP does not operate any long-distance links or a backbone. Its

infrastructure is located at the network edges, and its long-distance communications are conducted from the underlying native-layer ISPs. In fact, early ISPs were often built in the same way, leasing long-distance trunks from telecommunication providers and placing IP routers at aggregation points at the network edge.



(a) Direct MON path (b) Indirect MON path

Figure 45: Direct and indirect MON paths.

In this chapter, we focus on the *MON design problem*, *i.e.,* where to place MON nodes and how to select the upstream ISPs for each node. We aim to examine, with realistic network performance and pricing data, whether an OSP can combine multihoming and overlay routing to provide a network service that is, first, profitable, second, better in terms of performance than the competing native ISPs, and third, less expensive than the competing native ISPs.[1] Perhaps surprisingly, we find out that the OSP can meet all three objectives. We also show, however, that the MON design process is crucial. For example, operating more than 10 MON nodes or routing traffic through the minimum-delay MON path, rarely leads to profitability in our simulations.

In more detail, we formulate an optimization problem where the OSP aims to maximize its profit by placing up to N MON nodes and connecting each node with up to K locally present ISPs. The OSP revenues come from subscribed customers while the costs are due to leased upstream capacity and node deployment. The optimization is constrained because a potential customer will only subscribe to the OSP if the latter can offer better performance than the competing native ISP, at least for a large fraction of the customer's traffic. As in any network design problem, we focus on large timescales, namely weeks or months. The

[1]In terms of availability, we assume that the OSP can take advantage of its multihoming and overlay routing capabilities to provide higher availability than singlehoming or just multihoming.

reason is that both the deployment of MON nodes and the contractual agreements with native ISPs are hard to change in shorter timescales. Consequently, the performance metric we consider is the propagation delay between MON nodes. Other metrics, such as loss rate or available bandwidth, vary significantly in shorter timescales and so they would not be appropriate as inputs to a network design problem.

6.2 Model and problem formulation

The MON design problem involves ISPs and the performance of the native network, the location and traffic matrix of potential customers, and the OSP routing strategy, pricing function and node deployment costs. In this section, we present a model for these components of the problem and formulate a MON design optimization framework. We also prove that the optimal MON design problem is NP-hard.

6.2.1 ISPs and the native network

Consider a geographical area that the OSP aims to cover. There are L possible locations where the OSP can place MON nodes (*e.g.,* IXP locations or network access points). The set of ISPs that are present at location $l \in L$ is denoted by I_l, while the union of all such sets is I. We use the term *POP* $p = (l, i)$ to identify the access point to ISP i at location l. $\mathcal{LOC}(p)$ and $\mathcal{ISP}(p)$ are the location and ISP that correspond to POP p, respectively. The set of all POPs is denoted by P.

We represent the native-layer performance with the matrix $T_{|P|\times|P|}$, where the entry $\tau_{p,q}$ represents the propagation Round-Trip Time (RTT) from POP p to q. We expect that most of the elements in this matrix remain practically constant for weeks, except during periods of interdomain routing instability. The matrix T can be measured directly, as long as the OSP can conduct simple measurements (*e.g.,* ping) between any pairs of POPs. If that is not possible, the matrix T can be estimated using the technique presented in Section 6.3.

6.2.2 MON representation

MON consists of up to N nodes, with each node placed at a different location of L. Each MON node is multihomed to at most K locally present ISPs. We say that *a MON node*

is present at POP p if the node is located at $\mathcal{LOC}(p)$ and connected to ISP $\mathcal{ISP}(p)$. The entire MON network can be represented with the *POP selection vector* $\mathcal{MON}$,

$$\mathcal{MON}(p) = \begin{cases} 1 & \text{if a MON node is present at POP } p \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

Given a POP selection vector, we can identify the locations of all MON nodes with:

$$\mathcal{NODE}(l) = \begin{cases} 1 & \text{if } \sum_{\mathcal{LOC}(p)=l} \mathcal{MON}(p) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

6.2.3 Customers and OSP-preferred flows

We denote the set of potential OSP customers as U. A customer u is present at location $\mathcal{LOC}(u)$ and, before subscribing to the OSP, is connected to one or more locally present ISPs. To attract a customer, the OSP needs to provide better performance (lower RTT) to most of that customer's traffic. Specifically, the workload of customer u is a set of *flows* $F(u)$. A flow $f = (s_f, d_f, r_f, \tau_f)$ is defined as a large traffic aggregate from one of u's source POPs towards a destination POP, where s_f and d_f are the flow's source and destination POPs, respectively, r_f is the flow's average rate, and τ_f is the RTT in the flow's native path. The set of all flows is F. The flow f is *OSP-preferred* if the OSP can route f, through a direct or indirect path, with RTT $\hat{\tau}_f < \tau_f$. The corresponding MON path is referred to as *OSP-preferred* path for flow f. Customer u *subscribes* to the OSP if at least a fraction H (say 70-80%) of its traffic is in OSP-preferred flows.

6.2.4 OSP routing strategy

As mentioned earlier (and shown in Figure 45), an OSP can utilize either one of the direct paths from the ingress MON node to the destination, or one of the indirect paths through an intermediate MON node. The OSP uses a certain *routing strategy* to select the path to each destination. One routing strategy is that the OSP always selects the path, direct or indirect, with the minimum native RTT.

In most of this chapter (except for Section 6.5.4) we consider a more economical strategy,

referred to as *Direct-Routing-First (DRF)*. With DRF, the OSP first attempts to route a flow f through the direct path with the minimum RTT. If that path does not result in lower RTT than τ_f, and there exists an OSP-preferred indirect path, the OSP selects the indirect path with the minimum RTT. The reasoning behind DRF is that indirect paths are more costly because the OSP has to pay for upstream capacity at two MON nodes (rather than at a single node for the case of direct paths). So, with DRF, the OSP assigns higher priority to direct paths than to indirect paths.

If $\mathcal{PATH}(f)$ is the sequence of POPs for an OSP-preferred flow f using a certain routing strategy, then the following function identifies whether flow f passes through the MON node at POP p,

$$\mathcal{RTE}(p, f) = \begin{cases} 1 & \text{if } p \in \mathcal{PATH}(f) \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

Given a POP selection vector and an OSP routing strategy, as well as the set of flows $F(u)$, it is easy to determine whether each flow of u can be OSP-preferred, and thus whether customer u would subscribe to the OSP or not,

$$\mathcal{SUB}(u) = \begin{cases} 1 & \text{if customer } u \text{ subscribes to OSP} \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

6.2.5 OSP revenues and costs

The OSP generates revenue from subscribed customers. Let $\hat{P}(r)$ be the OSP pricing function, where r is the total traffic rate from a customer, say in the period of a month. Then the total OSP revenue in that period is:

$$R(\mathcal{MON}) = \sum_{u \in U} \mathcal{SUB}(u) \cdot \hat{P}(\sum_{f \in F(u)} r_f) \tag{25}$$

The OSP has two types of costs: first, a recurring cost for each deployed node (*i.e.,* monthly fee at IXPs) and second, the cost of upstream capacity from native ISPs. In general, different ISPs have different capacity pricing functions, and these functions may

vary across different locations. Therefore, we calculate the OSP capacity cost on a per-POP basis. Specifically, the required upstream capacity at POP p is:

$$c(p) = \sum_{u \in U} \mathcal{SUB}(u) \cdot \sum_{f \in F(u)} \mathcal{RTE}(p, f) \cdot r_f \tag{26}$$

and so the total capacity cost is:

$$C_{CP}(\mathcal{MON}) = \sum_{p} \mathcal{MON}(p) \cdot P_p\left(c(p)\right) \tag{27}$$

where $P_p(\cdot)$ is the pricing function used by the ISP at POP p.

The total node deployment cost can be modeled as:

$$C_{ND}(\mathcal{MON}) = \sum_{l \in L} \mathcal{NODE}(l) \cdot d(l) \tag{28}$$

where $d(l)$ is the cost of deploying a MON node at location l.

The OSP pricing function $\hat{P}(r)$, as well as the ISP pricing functions $P_p(r)$, are assumed to be non-decreasing and concave, which is the typical case in practice [28]. The concavity is important because it implies *economies of scale*, *i.e.,* the price per Mbps decreases as the purchased capacity r increases. The OSP can exploit this property of the capacity market to offer less expensive services than the competing native ISPs by aggregating the traffic from many customers. Specifically, suppose that the pricing ratio R_p between the OSP and ISP pricing functions is constant,

$$R_p = \frac{\hat{P}(r)}{P_p(r)} \tag{29}$$

If $R_p < 1$, the OSP is less expensive than the ISP at POP p. The OSP can still be profitable, however, because the aggregation of traffic from several customers means that the OSP can purchase upstream capacity at a lower unit price than what it charges to its customers.

6.2.6 Problem statement

We can now state the MON design problem as the following constrained nonlinear optimization problem. Given the following inputs:

Native network information: Set of locations L and ISPs I; Delay matrix T; ISP pricing functions $P_p(\cdot)$ for all $p \in P$;

OSP information: OSP routing strategy; OSP pricing function $\hat{P}(\cdot)$; MON node deployment cost $d(l)$ for all $l \in L$;

Customer information: Set of customers U with their flow descriptors $F(u)$ for all $u \in U$; Subscription threshold H;

Determine the POP selection vector $\mathcal{MON}$ to maximize the profit:

$$\Pi(\mathcal{MON}) = R(\mathcal{MON}) - C_{CP}(\mathcal{MON}) - C_{ND}(\mathcal{MON}) \tag{30}$$

under the following constraints:

1) At most N MON nodes: $\sum_{l \in L} \mathcal{NODE}(l) \leq N$

2) Maximum multihoming degree K: For all $l \in L$, $\sum_{p \in P, \mathcal{LOC}(p)=l} \mathcal{MON}(p) \leq K$

6.2.7 NP-hardness

We next give a sketch of the NP-hardness proof based on a reduction from the set covering problem. Consider a set of sets S, where $\bigcup_{s_i \in S} s_i = X$. The set covering problem is to find a minimum-size set $C \subseteq S$ such that $\bigcup_{c_i \in C} c_i = X$. We construct an instance of the MON design problem in which there is a set of ISPs S available at a single location l and the MON design constraints are N=1 and K=$|S|$. Let X be the set of customers at location l, with one flow per customer, and suppose that all flows can be OSP-preferred, *i.e.,* there is at least one ISP in S that can make each flow OSP-preferred. Assume that the OSP has a constant pricing function $\hat{P}(r) = p_{osp}$, and all ISPs have the constant pricing functions $P_p(r) = p_{isp}$, with $p_{osp} \gg p_{isp}$ and $p_{osp} \gg d(l)$. Under these constraints, the OSP should deploy a single MON node at location l, and determine the minimum-size set of ISPs that maximizes its profit. We can determine (in polynomial time with $|S|$ and $|X|$) the set of flows s_i that become OSP-preferred when the MON node is connected to the i'th ISP. Because $p_{osp} \gg p_{isp}$ and $p_{osp} \gg d(l)$, the OSP can maximize its profit by connecting to just enough ISPs so that *all* flows are OSP-preferred. In other words, the OSP needs to find a minimum-size set of ISPs $C \subseteq S$ such that $\bigcup_{c_i \in C} c_i = X$. So, any instance of the set covering problem can be reduced in polynomial time to the previous instance of the

MON design problem. Therefore, given that the set covering problem is NP-hard, the MON design problem is also NP-hard.

6.3 Estimating the native RTT matrix

In this section, we describe a methodology for estimating the native RTT matrix T. Recall that this matrix consists of the POP-to-POP pairwise RTTs in the native network, and it is a required input for one of the four heuristics of the next section. Also recall that the RTT we aim to estimate is the propagation delay RTT, which mostly depends on the physical distance between two POPs and the routing at the underlying native network; queuing delays, in particular, are not included in these RTTs.

We distinguish between intradomain RTTs, where both POPs are in the same Autonomous System (AS), and interdomain RTTs, otherwise. As shown next, an intradomain RTT can be modeled as proportional to the physical driving distance between the two POPs, at least for the networks we measured. An interdomain RTT, on the other hand, further increases with the number of AS's in the route between the two POPs. So, in the interdomain case, the ratio between RTT and driving distance depends on the number of AS hops.

6.3.1 Intradomain case

To develop an accurate model for intradomain RTTs, we examined the correlation among various distance metrics and the measured RTTs between POPs of various network providers in the US. The highest correlation resulted from the "highway driving distance", as reported from Google-map. This is probably because most backbone optical fiber is laid along highways or railroads. In the following, we analyze RTT pairwise measurements between 15 ping servers (located at POPs) in the US Level3 network. For each source/destination POP pair, we conducted 1,000 consecutive ping measurements, reporting the minimum RTT as the best estimate of the propagation delay RTT. Figure 46 shows that the minimum measured RTT varies almost linearly with the physical driving distance, with a correlation coefficient of about 95%. Therefore, the OSP could model the intradomain RTT between

two POPs p and q of the Level3 network as:

$$\tau_{intra}(p,q) = 0.02349 \cdot L(p,q) \tag{31}$$

where $\tau_{intra}(p,q)$ and $L(p,q)$ are the intradomain RTT (in milliseconds) and the driving distance (in miles) from p to q, respectively. The same procedure should be followed for each ISP, because the proportionality constant between RTT and distance may be different across ISPs.

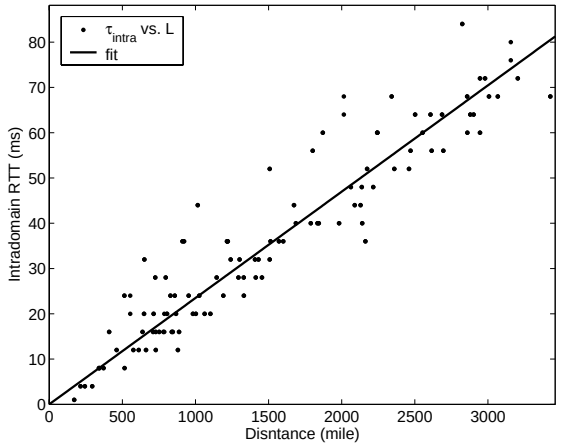
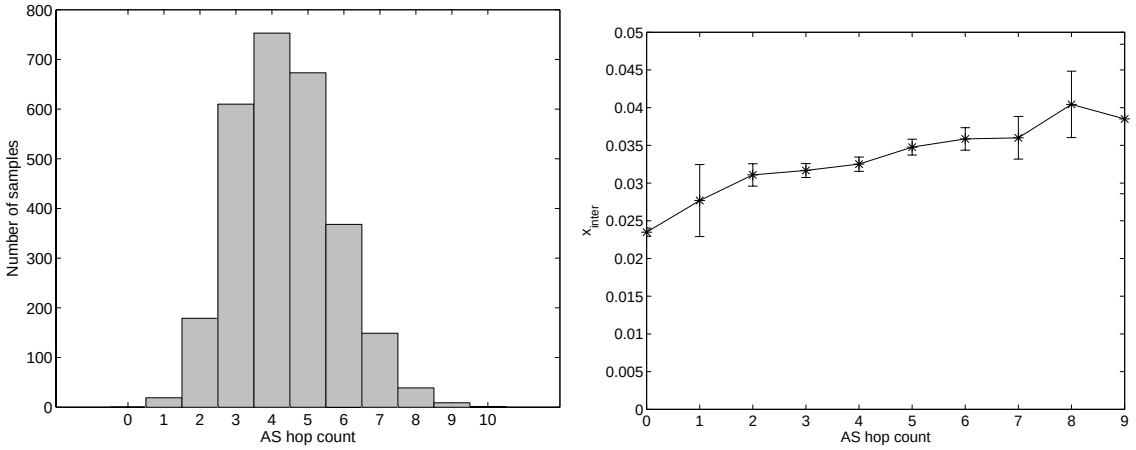


Figure 46: Intradomain RTT versus driving distance in the US Level3 network



(a) AS hop-count histogram

(b) Ratio between RTT and driving distance for various AS hop-counts

Figure 47: Relation between interdomain RTT and AS hop-count

6.3.2 Interdomain case

We observed that in the interdomain case, the RTTs are not so highly correlated with the physical distance between POPs. The reason is that interdomain routes are commonly dictated by policy constraints, and so they often follow suboptimal paths. We observed, however, that there is a high correlation between RTT and driving distance when we consider routes with the same number of AS hops (*i.e.,* the same number of traversed networks). Consequently, we can model the interdomain RTT between two POPs p and q as:

$$\tau_{inter}(p,q) = x_{inter}(h) \cdot L(p,q) \tag{32}$$

where $x_{inter}(h)$ is a constant that depends on the number of AS hops h in the native route between POPs p and q.

To estimate $x_{inter}(h)$ for various values of h, we measured 3,136 interdomain paths from 32 PlanetLab US nodes to 98 web servers (of museums and newspapers) with two servers per continental state. For each path, we conducted 1,000 ping measurements (to estimate the propagation delay RTT) and a traceroute measurement. The latter was used to estimate the AS hop-count of the path, using the IP-to-AS mapping database of [51]. Figure 47(a) shows the histogram of AS hop-counts in the measured paths. Note that most paths have 2-7 AS hops, including the destination AS (but not the source AS).

Figure 47(b) shows the 95% confidence interval of $x_{inter}(h)$ as a function of h. Note that h=0 corresponds to the intradomain case. The results show clearly an increasing trend in $x_{inter}(h)$ as h increases. Furthermore, the confidence intervals are narrow, indicating that the RTT can be modeled as proportional to the driving distance for a given AS hop-count, except the single hop and 7-9 hop cases (for which we do not have enough measurements though). In summary, the OSP can estimate the RTT between two interdomain POPs as long as it can determine the number of AS's between the two POPs (*e.g.,* through BGP routing feeds) and also construct a measurement-based graph such as Figure 47(b).

Table 3: Input requirements for each heuristic

Heuristic	Customer info	Traffic info	Performance info
RAND			
CUST	√		
TRFC	√	√	
PERF	√	√	√

6.4 MON design heuristics

The MON design problem involves two major tasks: 1) select up to N locations for placing MON nodes; 2) select up to K upstream ISPs for each deployed MON node. In this section, we present four heuristics for the MON design problem. The heuristics differ in terms of their inputs, ranging from a simple random heuristic (RAND) that does not require any customer or performance data, to the most complex heuristic (PERF) that utilizes information for the location of customers, the traffic matrix of each customer, and the native delay matrix (see Table 3). The heuristics assume that the node deployment cost is the same at all locations, and the ISPs at the same location have identical pricing functions. The latter is a reasonable assumption for a stable and competitive ISP market.

6.4.1 Random (RAND)

This heuristic represents a naive approach in which we select N random locations, and connecting node at location l to a random set of $\min(K, |I_l|)$ locally present ISPs.

6.4.2 Customer-driven (CUST)

In RAND, different locations and ISPs have equal probabilities of being selected. Obviously, this strategy will not perform well when customers are not uniformly distributed. CUST utilizes information about the location of each customer and places N MON nodes at the locations with the maximum number of customers. Placing MON nodes at those locations enables more customers to connect to the OSP and therefore it increases revenues. At each selected location l, CUST then selects the $\min(K, |I_l|)$ locally present ISPs with the maximum coverage, *i.e.,* the ISPs that are present at the largest number of locations. The

intuition here is that larger-coverage ISPs can typically reach traffic destinations through fewer AS's, and so they are more likely to provide lower RTTs.

6.4.3 Traffic-driven (TRFC)

Although CUST utilizes the profile of customers locations, it does not consider the traffic that each customer generates, nor the distribution of traffic destinations. The traffic-driven heuristic uses the aggregated traffic rate that originates from all potential customers at a location. TRFC places N MON nodes at the locations with the largest volume of aggregated customer traffic. By placing MON nodes at locations where "traffic-heavy" customers are located, more traffic can subscribe to the OSP generating more revenue than CUST. At each selected location l, TRFC then selects the $\min(K, |I_l|)$ locally present ISPs that receive the maximum traffic rate from customers. The intuition here is that an ISP that can deliver traffic directly to its destination will probably result in lower delays than an ISP that delivers traffic through other AS's.

6.4.4 Performance-driven (PERF)

Note that the CUST and TRFC heuristics do not utilize any native delay information. If there are OSP-preferred direct paths then these two heuristics perform quite well in identifying a good set of locations, because the DRF routing strategy does not need to consider indirect paths in that case. If, however, many customer flows only have indirect OSP-preferred paths, then the previous two heuristics cannot choose good locations for placing intermediate MON nodes. This motivates the performance-driven heuristic (PERF). PERF requires an estimate of the native delay matrix T. The key idea in PERF is to select locations and upstream ISPs that will turn as many flows to OSP-preferred as possible.

The location selection process is performed iteratively. PERF keeps track of the set $\bar{L}$ of locations that are not yet selected and the set $\bar{F}$ of flows that are not yet OSP-preferred. Initially, $\bar{L} = L$ and $\bar{F} = F$. During each iteration, PERF associates a weight $W_L(l)$ with each location l to represent the amount of traffic that can become OSP-preferred if l is selected. At the beginning of an iteration, all weights are set to zero. PERF then processes the flows in $\bar{F}$ sequentially. For each flow $f \in \bar{F}$, PERF first finds all OSP-preferred MON

paths for that flow based on the currently chosen locations, assuming that these locations are multihomed to all locally present ISPs (the assumption will be refined during the ISP selection phase of the algorithm). Then, PERF updates the weight $W_L(l)$ as follows: If l is the ingress location of a direct path for flow f, $W_L(l)$ is increased by the rate r_f. If l is either the ingress or the intermediate location of an indirect path for f, $W_L(l)$ is increased by $r_f/2$. After all flows have been processed, the location with the highest weight is selected and removed from $\bar{L}$, and the flows that are now OSP-preferred are removed from $\bar{F}$. This process repeats until we either select N locations or all flows are OSP-preferred. After the MON node locations have been selected, the OSP-preferred flows are routed based on the given OSP routing strategy. The set of flows assigned to location l is denoted by $F_A(l)$.

Next, PERF enters the ISP selection phase for each selected location l. For each locally present ISP i, we calculate the weigh $W_I(i)$ as the traffic rate of all flows in $F_A(l)$ that use ISP i in their OSP-preferred path. The ISP with the highest weigh is selected and the process is repeated until either K ISPs are selected or all flows in $F_A(l)$ are OSP-preferred. The pseudo code for the PERF heuristic is shown in Algorithm 7.

Algorithm 7 Performance-driven heuristic

Location selection:
  Initialize $\bar{L} = L$, $\bar{F} = F$;
  Repeat until N locations are selected or $\bar{F} = \varnothing$;
  Begin
    Initialize weights $W_L(l) = 0$, $\forall l \in \bar{L}$;
    For each flow $f \in \bar{F}$, find all OSP-preferred paths, and update $W_L(l)$ as follows:
      If l is on a direct OSP-preferred path:
        $W_L(l) = W_L(l) + r_f$;
      If l is on an indirect OSP-preferred path:
        $W_L(l) = W_L(l) + r_f/2$;
    Select location $\hat{l} = \arg\max_l W_L(l)$;
    Update $\bar{L}$=$\bar{L}$-$\{\hat{l}\}$ and remove OSP-preferred flows from $\bar{F}$;
  End
  Assign OSP-preferred flows to selected location l, calculate $F_A(l)$;
ISP selection (for each selected location l):
  Initialize $\bar{I} = I_l$, $\bar{F} = F_A(l)$;
  Repeat until K ISPs are selected or $\bar{F} = \varnothing$;
  Begin
    Calculate the weight $W_I(i)$ as the total traffic in $F_A(l)$ that can be OSP-preferred if ISP i is selected;
    Select ISP $\hat{i} = \arg\max_i W_I(i)$;
    Update $\bar{I}$=$\bar{I}$-$\{\hat{i}\}$ and remove the OSP-preferred flows from $\bar{F}$;
  End

6.5 Performance evaluation

We conducted extensive simulations to compare the MON design heuristics, study the OSP profitability and performance depending on several key parameters (number of MON nodes, degree of multihoming, node deployment cost, OSP/ISP pricing ratio), and examine various OSP routing strategies.

6.5.1 Simulation setup

We randomly chose 51 cities in the continental US as the set of POP locations L. The population of these locations varies from 29,000 to 8,100,000.[2] These locations are served, overall, by 100 ISPs. The average number of ISPs per location is 10, and the number of ISPs per location is proportional to the logarithm of that locations's population.

Unless otherwise specified, we assume a *population-based customer distribution* (CUST-POPUL), *i.e.,* the number of customers at each location is proportional to that locations's population. Furthermore, 70% of the customers are multihomed to 2-4 ISPs (as long as there are enough ISPs at that location). For a multihomed customer, each flow originates from a single ISP at that location; the assignment of flows to ISPs is random. We model only the 10 largest flows of each customer; recall that a "flow" in this context is a traffic aggregate from a customer to a destination POP. The flow destinations are uniformly distributed across all POPs. The average flow rate follows the *gravity model* (RATE-GRVTY), *i.e.,* the rate between a pair of POPs is proportional to the product of the population at the two locations [68]. The flow rates are normalized by a constant factor so that the average flow rate is 1Mbps. We set the customer subscription threshold to H=70%. Unless otherwise specified, the maximum multihoming degree is K=2, and the total number of potential customers is 500.

An important input to the MON design problem is the ISP pricing function $P_p(r)$ at each POP p and the OSP/ISP pricing ratio $R_p = \hat{P}(r)/P_p(r)$. It has been observed that, at least during the last 10 years or so, ISP capacity pricing shows economies of scale, *i.e.,* the price per Mbps drops almost logarithmically with the purchased capacity r [28]. Based

[2] We have also experimented with the 50 largest US cities and the results are very similar.

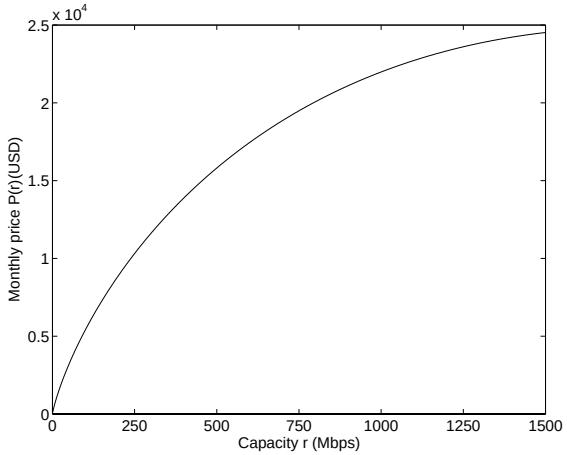


Figure 48: Internet capacity pricing function

on data from [2], and using the previous logarithmic relationship, we model the ISP pricing function as:

$$P_p(r) = [118 - 13.9 \cdot ln(r)] \cdot r \tag{33}$$

where r is measured in Mbps and the price is a monthly fee in USD (see Figure 48). Note that ISP pricing is sometimes done at discrete capacity values, and so the pricing function can be a discrete step-like function. Here, we assume a usage-based pricing model where both ISPs and the OSP charge based on the routed traffic volume. For simplicity, we also assume that the pricing function is the same at all POPs, and that the pricing ratio R_p is constant with r, the same at all POPs. If these assumptions do not hold in practice, the MON design process can still use the proposed optimization framework but the evaluation will be more cumbersome. Unless otherwise specified, the deployment cost per MON node at any location l is $d(l)$=\$5,000, based on [4], and the pricing ratio R_p=0.8.

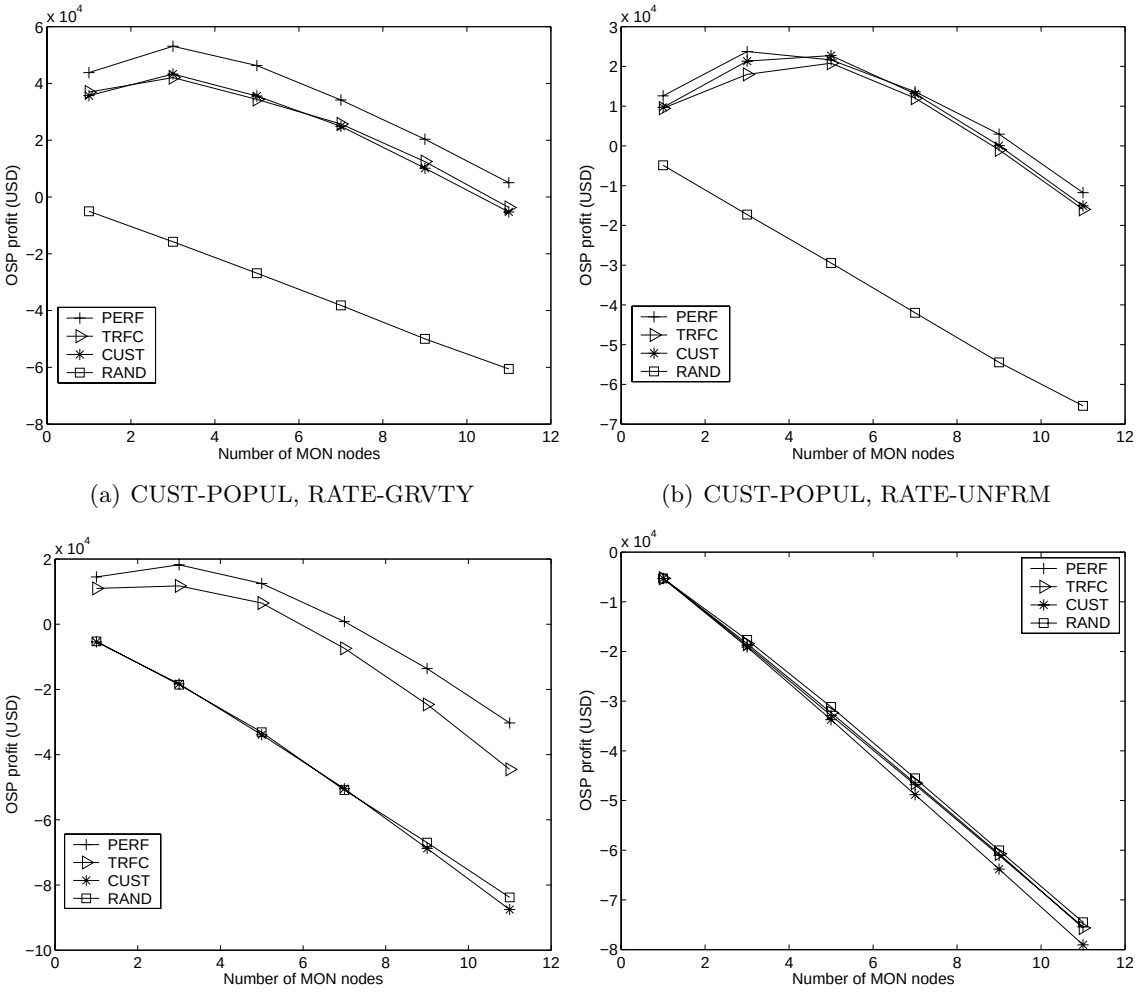
6.5.2 Comparison of MON design heuristics

We first compare the four MON design heuristics in terms of OSP profitability as we increase the number of MON nodes. Together with the default models, CUST-POPUL and RATE-GRVTY, we also examine models of uniform customer distribution across all locations (CUST-UNFRM), and uniform traffic rate distribution across all flows (RATE-UNFRM).

The average flow rate is adjusted in each model to maintain similar maximum traffic load at the MON nodes.

Figure 49 shows that, as expected, heuristics that utilize more information about customers, traffic, or the native network perform better. Specifically, PERF outperforms all other heuristics, while RAND performs so poorly that it never leads to positive profit. In the more realistic combination of models, CUST-POPUL and RATE-GRVTY, PERF performs significantly better than other heuristics, providing a maximum monthly profit of about \$50,000 instead of \$40,000 with TRFC or CUST (Figure 49(a)). Clearly, the OSP would have a strong motivation to optimize the MON design process, even if that requires the collection of more input data. In the following sections we show results only for PERF.

With the CUST-POPUL and RATE-UNFRM models (Figure 49(b)), the OSP has lower profit, compared to the RATE-GRVTY model, because traffic tends to be more dispersed. So it becomes harder to aggregate large traffic volumes destined to the same POP and route them through direct paths. Furthermore, TRFC and PERF do not perform much better than CUST because the additional information they have about the traffic is not so useful with the RATE-UNFRM model. With the CUST-UNFRM and RATE-GRVTY models (Figure 49(c)), CUST performs equally bad with RAND, because in this case placing nodes where most customers are located is no different than placing nodes randomly. TRFC and PERF perform better, but the profits are still significantly lower than the CUST-POPUL model because the OSP cannot place nodes in just a small number of locations where most customers are. Finally, the CUST-UNFRM and RATE-UNFRM models (Figure 49(d)) do not lead to OSP profitability, with any of the four heuristics, under the default OSP/ISP pricing ratio $R_p = 0.8$. Achieving substantial traffic aggregation with just a few MON nodes is much harder in this case. At the same time, the OSP has to pay the node deployment costs and so it does not end up with profit.



(a) CUST-POPUL, RATE-GRVTY

(b) CUST-POPUL, RATE-UNFRM

(c) CUST-UNFRM, RATE-GRVTY

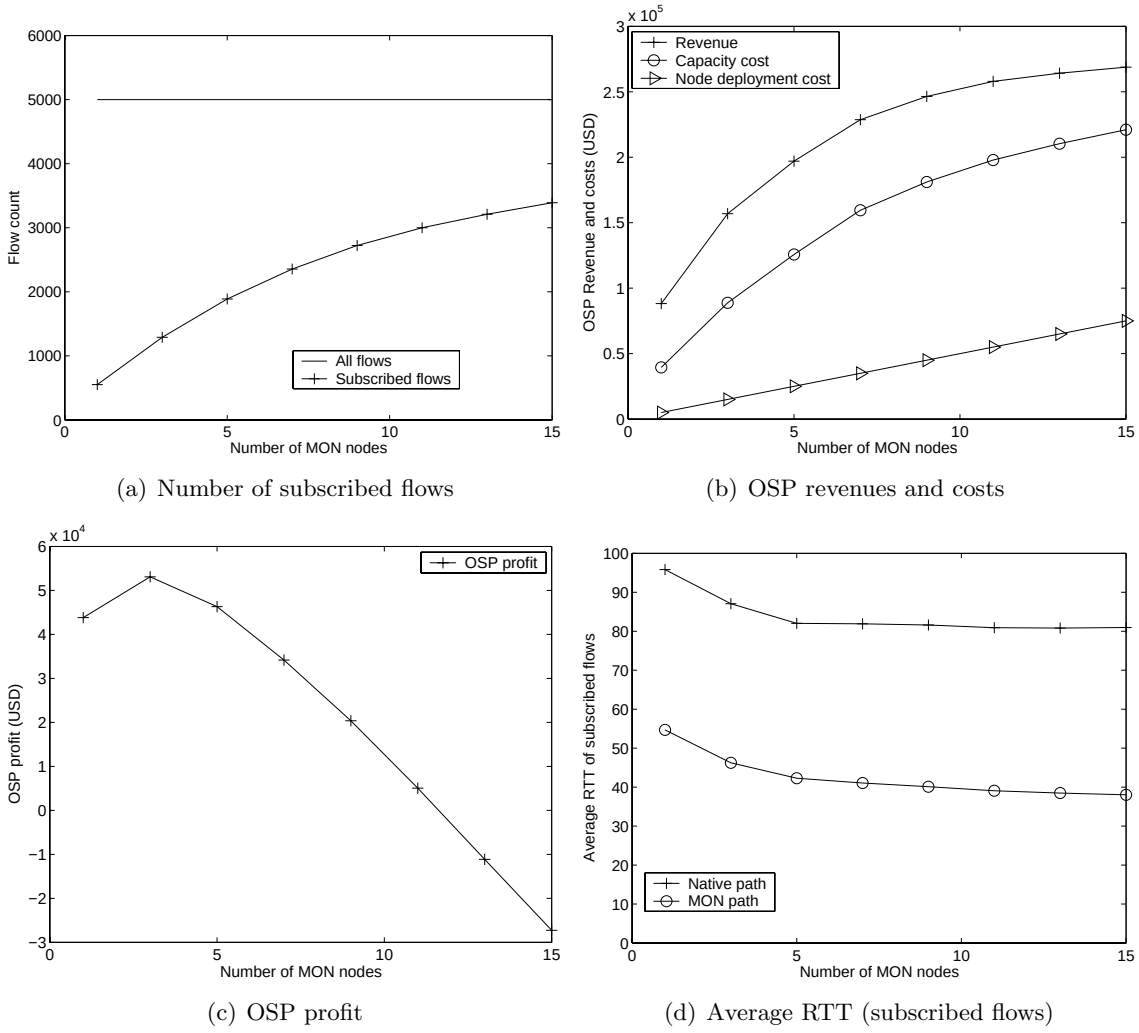
(d) CUST-UNFRM, RATE-UNFRM

Figure 49: OSP profitability under four customer and traffic distribution models

6.5.3 Optimal number of MON nodes

Here we investigate the effect of the number N of MON nodes on OSP performance and profitability. Figure 50(a) shows that the total number of flows from subscribed customers increases as more nodes are deployed. This is because more customers become collocated with MON nodes and the OSP can provide more paths towards each destination POP. As a result, the number of subscribed customers also increases with N.

Figure 50(b) shows that both the OSP revenues and costs increase with N. The reason that the revenue increase rate slows down as N increase is that, gradually, there are fewer new subscribed customers per node. The increase in capacity costs also follows a concave



(a) Number of subscribed flows

(b) OSP revenues and costs

(c) OSP profit

(d) Average RTT (subscribed flows)

Figure 50: Effects of number of MON nodes

shape, because the OSP traffic volume follows a similar pattern. But as N increases, the amount of traffic per MON node grows more slowly, reducing the economic benefit of aggregation. On the other hand, the deployment costs increase linearly. The net result, at least in these simulations where K=2, is that the OSP achieves maximum profit when N is only 3-4 nodes (as shown in Figure 50(c)). Comparing Figures 50(a) and 50(c), we note that deploying more nodes attracts more customer flows, but that does not increase the OSP profit due to increased costs.

Figure 50(d) shows the average native RTT, as well as the average MON RTT, for all flows from subscribed customers. The results show that by subscribing to the OSP, customers reduce the average RTT of their flows significantly, by about 40msec in these

simulations. This is despite the fact that only H=70% of the customer traffic is guaranteed to have lower MON RTT than native RTT. The reason for this large decrease is because OSP customers are not a random sample of the customer population. Instead, most of their traffic goes through long interdomain routes, allowing the OSP to offer significant RTT improvements through multihoming and overlay routing.

The reason both native and MON RTTs decrease with N is as follows. As N increases, more flows become subscribed to the OSP, and thus included in the calculation of the average RTT. These flows, however, tend to have lower native RTTs because they become OSP-preferred only after the OSP has deployed more than a number of nodes. For the same reason, the average MON RTT also decreases with N.

6.5.4 Effect of OSP routing strategy

We now examine the effect of the OSP routing strategy, comparing Direct-Routing-First (DRF) with 1) Minimum-MON-Delay-Routing (MDR), *i.e.,* the OSP routes flows through the MON (direct or indirect) path with the minimum RTT, and 2) Direct-Routing-Only (DRO), *i.e.,* the OSP routes flows through direct MON paths only. With the DRO strategy, the OSP operates similarly to InterNAP, a commercial network provider that utilizes intelligent route control and multihoming, but not overlay routing.

Figure 51(a) shows the average RTT of all flows using these three routing strategies; the average native RTT is also given for comparison. The results indicate that the OSP can reduce the RTT relative to native routing with any of the previous routing strategies. However, by combining multihoming with overlay routing, both DRF and MDR perform significantly better than DRO. Of course, by definition, MDR results in lower RTT than DRF, especially for larger N because the number of possible paths increases quickly with N.

On the other hand, Figure 51(b) shows that the DRF strategy leads to significantly higher profit than MDR. The reason is that MDR makes extensive use of overlay routing and indirect paths, and so the OSP often has to pay for upstream capacity at two locations instead of one to route a flow. The comparison between DRO and DRF is less clear and

consistent, making the two strategies roughly equivalent in terms of profit. The previous results show that the DRF strategy achieves a good tradeoff between OSP performance and profitability: DRF is close to MDR in terms of average RTT, which is much better than DRO. At the same time DRF is much more profitable than MDR and it is as profitable as DRO.

6.5.5 Effect of pricing ratio

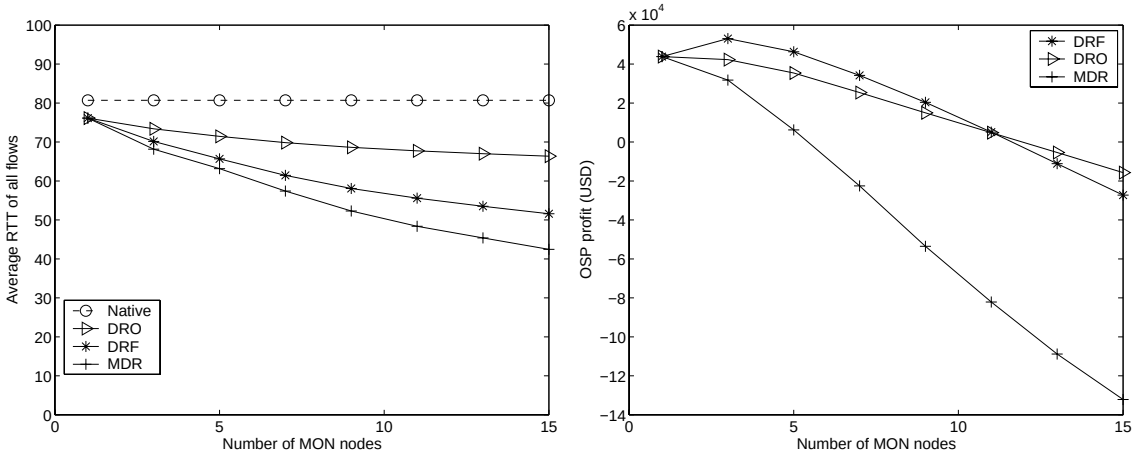
When the pricing ratio R_p is less than one, the OSP can offer a less expensive service than the native ISPs for the same traffic volume. If this is a profitable scenario, then the OSP can be both better and less expensive than native ISPs. Setting R_p too high can also hurt OSP's profitability because customers may not be willing to subscribe to the OSP despite the performance improvements.

In this simulation, we vary R_p from 0.4 to 2, monitoring the OSP profit as a function of the number of OSP customers. Figure 52 shows that the OSP can be profitable even when the OSP charges 40% of the ISP price, as long as there are enough customers. The reason is that, with enough customers, the OSP can achieve large traffic aggregation, and so it can purchase capacity from upstream ISPs at a lower price per Mbps than what it charges to its customers. An OSP with lower R_p needs more customers to be profitable. For example, the OSP only needs 50 customers to break even when R_p=2, but it needs more than 300 customers to make profit when $R_p = 0.8$.

6.5.6 Effect of maximum multihoming degree

The maximum multihoming degree K is another important parameter because it determines the local path diversity at each MON node. In the simulations so far, K was set to 2. Now we examine four values of K, from one (singlehoming) to four ISPs, as we increase the number of MON nodes N. Figure 53 shows the OSP profit under two node deployment costs: 1) $d(l)$=\$5,000, representing a new OSP that creates a MON from scratch, and 2) $d(l)$=\$100, representing an OSP that already has a distributed infrastructure in place (*e.g.,* Akamai).

Figure 53(a) shows that, with the higher deployment cost, the best strategy is to place



(a) Average RTT (all flows) (b) OSP profit

Figure 51: Effect of OSP routing strategy

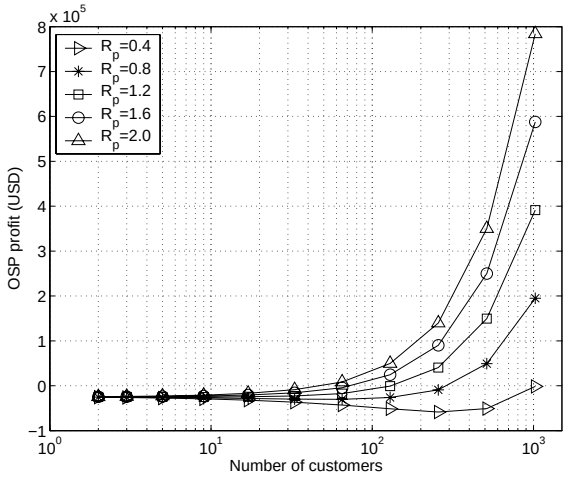
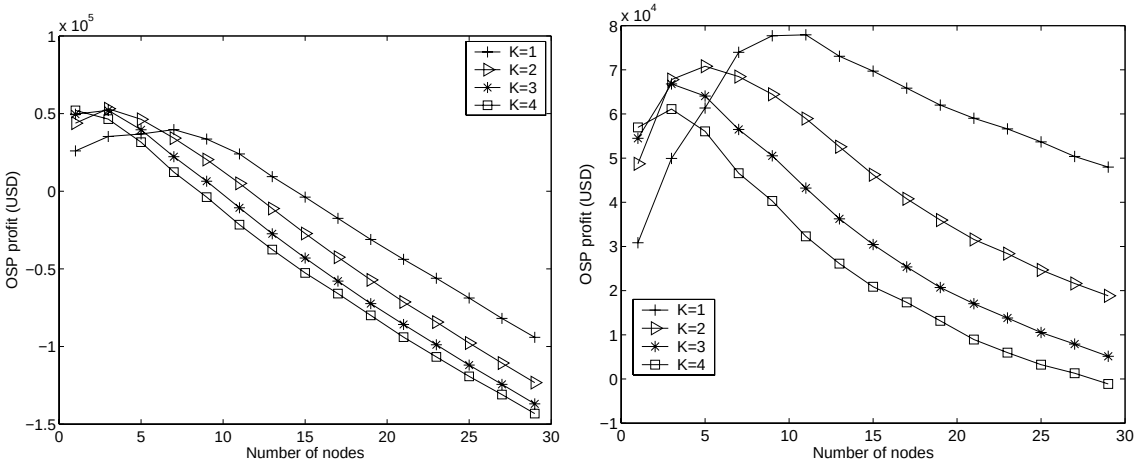


Figure 52: Effect of OSP/ISP pricing ratio

a small number of MON nodes (3-5 nodes in our simulations) and connect each node to 2-3 ISPs. Singlehoming is clearly suboptimal, and deploying more than 5-6 nodes quickly makes the OSP unprofitable. In contrast, Figure 53(b) shows that, with the lower deployment cost, placing several (around 10) singlehomed nodes is a more profitable OSP configuration. More generally, there is certainly a trade-off between the number of MON nodes and their multihoming degree. The most profitable point in this trade-off depends on the node deployment cost relative to what the OSP has to pay for capacity at each POP.



(a) Deployment cost: $5,000 per node

(b) Deployment cost: $100 per node

Figure 53: Effect of maximum multihoming degree

6.6 Sumarry

We examined the effectiveness of combing multihoming and overlay routing from the pragmatic perspective of a network provider (OSP) that attempts to be both profitable and also offer better and less expensive Internet access to its customers. Interestingly, we found out that it is possible to meet all previous objectives, as long as the OSP can follow the following basic guidelines: 1) use a performance-aware MON design heuristic (such as PERF) even if that requires additional inputs and measurements, 2) deploy nodes at few locations where significant traffic aggregation is possible, 3) connect each MON node to ISPs that can directly reach traffic-heavy destination POPs, 4) give direct paths higher priority than indirect paths, 5) charge less than the competing native ISPs for the same traffic rate to attract more customers, and 6) determine the best trade-off between the number of MON nodes and multihoming degree based on the node deployment cost.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

In this thesis, we address various issues of overlay network design, including the overlay routing algorithms, overlay network assignment and multihomed overlay networks. We also examine the behavior of overlay networks under a wide range of network settings and identify key factors that affects the performance of overlay networks. Based on these findings, practical design guidelines are also given. In this chapter, we summarize the major contributions of this thesis, and suggest several topic of future research.

7.1 Research summary

7.1.1 Dynamic overlay routing

In this thesis, we presented a simulation study of dynamic overlay routing. Given that most previous work focused on delay-driven path selection, we focused instead on avail-bw based overlay routing algorithms leveraging the recently developed measurements techniques for end-to-end avail-bw. We considered two main approaches for overlay routing, proactive and reactive, as well as a number of factors that can affect the performance of these routing algorithms.

The main conclusions of this study are:

- Reactive overlay routing performs better in terms of efficiency than native or proactive overlay routing. The efficiency gain compared to native routing can be substantial, especially if the network is not very lightly loaded. Also, reactive routing is much much stable than proactive routing.

- Proactive overlay routing performs better in terms of headroom (safety margin) than native and reactive overlay routing.

- A single intermediate overlay node is sufficient for reactive routing to achieve its

throughput and headroom gain over native routing. For proactive routing, limiting the maximum overlay hop count H_{max} to two is even more critical in terms of efficiency and stability.

- The reactive algorithm is quite robust to stale link-state information, and it performs better than native routing even when the link-state refresh period P_r is a few seconds. The proactive algorithm, on the other hand, is very sensitive to link-state staleness, and P_r should be as short as possible.

- A hybrid algorithm that acts reactively in about 90% of the time and proactively in about 10% of the time, can achieve a good compromise between high throughput, stability and safety margin, combining the best features of reactive and proactive routing.

- Overlay routing performs better with longer overlay flows, because the latter create lower traffic variability. Cross traffic variations can also decrease the performance of overlay routing, especially when these variations are significant in lower time scales than the path update period P_u.

- Relative errors in the avail-bw estimation process (which are common and probably unavoidable) will have negligible impact on the efficiency of hybrid overlay routing. Absolute or random errors, on the other hand, can have a significant impact.

- Even though native sharing effects can affect the performance of hybrid overlay routing, ignoring native sharing performs almost equally well with having complete information about the native network.

7.1.2 Virtual network assignment

In this thesis, we developed a basic scheme for VN assignment without reconfiguration and use it as a building block for all other advanced algorithms. Subdividing heuristics and adaptive optimization strategies are presented to further improve the performance. We also developed a selective VN reconfiguration scheme that prioritizes the reconfiguration for the most critical VNs.

We evaluate the performance of the proposed VN assignment algorithms through extensive experiments and our findings are summarized as follows:

- The basic VN assignment algorithm performs consistently better than the least-load algorithm.
- The benefits of subdividing the VN topology are more significant when the VN topology is sparse.
- The advantage of proposed VN assignment algorithms is more prominent when the substrate network is sparsely connected.
- The proposed VN assignment algorithms can effectively avoid hot spots or congestion in the substrate network (such as the transit-stub links).
- For VN reconfiguration, reconfiguring only a subset of the existing VNs that are most critical achieves most of the benefits of dynamic reconfiguration while keeping a low cost. Small reconfiguration threshold and reconfiguration period should be used but their exact values should be determined by the specific substrate topology to achieve the desired tradeoff between performance and cost.

7.1.3 Overlay network assignment tool

In this thesis, we design and implement NetFinder, an automatic overlay network configuration tool for PlanetLab. NetFinder continuously collects information about the resource utilization of PlanetLab and accepts the user-defined overlay topology as inputs. It then selects the set of PlanetLab nodes and their interconnection for the user overlay. Performance evaluation using realistic PlanetLab data demonstrates the advantage of our service by comparing it with alternative heuristics or existing tools and indicate that NetFinder is highly effective in locating the available PlanetLab resources and avoid overloaded nodes and links.

7.1.4 Multihomed overlay network

This thesis examines the effectiveness of combing multihoming and overlay routing from the pragmatic perspective of a network provider (OSP) that attempts to be both profitable and also offer better and less expensive Internet access to its customers. Interestingly, we found out that it is possible to meet all previous objectives, as long as the OSP can follow the following basic guidelines: 1) use a performance-aware MON design heuristic (such as PERF) even if that requires additional inputs and measurements, 2) deploy nodes at few locations where significant traffic aggregation is possible, 3) connect each MON node to ISPs that can directly reach traffic-heavy destination POPs, 4) give direct paths higher priority than indirect paths, 5) charge less than the competing native ISPs for the same traffic rate to attract more customers, and 6) determine the best trade-off between the number of MON nodes and multihoming degree based on the node deployment cost.

7.2 Future work

This thesis suggests several topic for future work.

7.2.1 Dynamic overlay routing

In this thesis we chose to ignore bandwidth sharing issues in congested links and congestion responsive transport mechanisms. Congestion control adds a feedback loop between the overlay nodes and the network that may interact with the overlay routing feedback loop, causing effects that are currently largely unexplored [12]. These interactions will be an interesting topic for future work.

7.2.2 Network virtualization

The future research direction in network virtualization can be categorized into the following two aspects.

7.2.2.1 Substrate Monitoring and Data Processing

Substrate network monitoring refers to the task of periodically obtaining the multi-metric substrate network resource availability information. Since the substrate network may have a

large number of nodes/links, the major challenge is to obtain this information in a timely and scalable manner. It deserves a sperate study to investigate and develop scalable performance monitoring schemes to continuously monitor the resource availability in a large substrate network. To reach this goal, future research should cover the following issues:

- Substrate performance metrics: Evaluate various substrate network performance metrics and identify the set of substrate performance metrics most relevant for the purpose of network virtualization. New performance metrics may also defined when necessary.

- Measurement requirement: Identify key factors that affect the VN assignment performance, investigate the sensitivity of VN assignment performance to the quality of measurement results such as staleness, granularity, and measurement error. Based on these findings, we can determine the requirement of substrate layer information, including the measurement frequency, accuracy and granularity.

- Measurement techniques: For the selected metrics and their measurement requirements, it is necessary to investigate existing measurement tools and techniques, identify their feasibility and limitations on large scale substrate networks. We may propose possible modifications or design new measurement tools for network visualization to meet the measurement requirements. Protocols to exchange measurement results also need to be addressed.

- Scalable and efficient monitoring schemes: Besides modifying the measurement tools themselves, we can also improve the scalability of the substrate monitoring by performing substrate network measurements in a more intelligent way. For example, one observation is that not all parts of the substrate network are important for VN assignment. Instead, the overloaded substrate nodes and links are unlikely to be chosen in the VN assignment process. This means that the information regarding over loaded parts of the substrate network is not critical to the VN assignment decision and we may use this information to prioritize the measurement for more critical parts of the substrate network.

Another approach is to measure the substrate network in the unit of region. The intuition behind this is that the overloaded/underloaded links/nodes of the substrate network tends to form clusters since the resource are used in the unit of a whole VN instead of isolated virtual links, shown in Figure 54. Therefore, overloaded regions can be efficiently removed and fine grained measurements only need to be performed in the remaining substrate network.

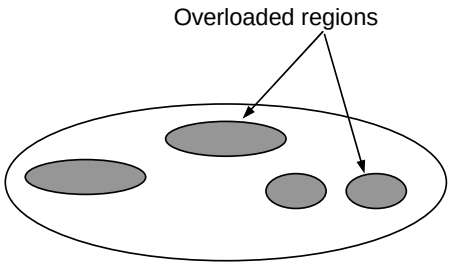


Figure 54: Overloaded regions in the substrate network

Both of the above ideas are expected to decrease the measurement overhead without penalizing the VN assignment performance.

- Data storage and processing: After getting the substrate network measurement results, the next step is to process and store the substrate information to enable scalable resource discovery. The measurement results are processed to keep only the useful information for VN assignment. Future work should also explore suitable data structure and searching techniques to support scalable storage and querying for the purpose of substrate network resource discovery.

7.2.2.2 Control and Maintenance

As the network and demand grow, we may need some specialized VN manager, who is responsible for discovering resources for VN request and controlling and maintaining VNs in the substrate network. Future research efforts on VN manager should cover various design issues as follows:

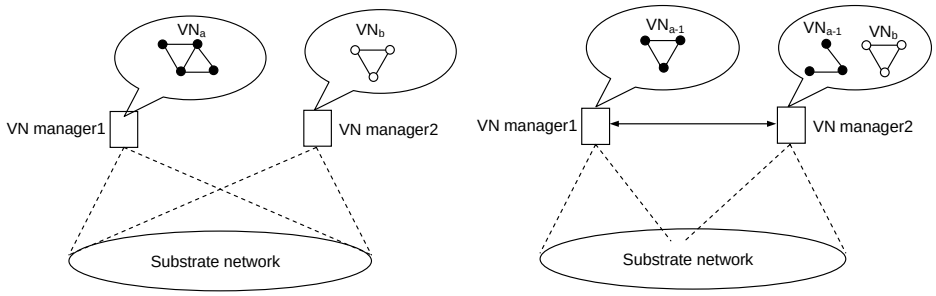
- Adapting VN changes: One purpose of using VN is for the deployment new network

technologies. In such scenario, VNs could be deployed in multiple phases. Starting from small test networks with low bandwidth requirement, the VN can be gradually expanded in size and bandwidth over time. When a certain VN changes (either topology change or constraint change), the VN manager should be able to adjust the assignment accordingly. The adjustment allows the changed VN to better fit the substrate network and results in more efficient substrate network resource utilization. However, since the adjustment involves shifting the existing VN to its new assignments, it is desirable that the adjustment could be carried out with minimum disruption to the existing VN. Furthermore, when future VN changes are planned, it is an interesting problem to see how a VN manager could assign the substrate network resources such that future changes do not cause significant adjustments.

- Resilient VN assignment: In the event of network failure, parts of the substrate network may become unavailable which will cause service disruption for all VNs using them. If VNs become widely deployed, this type of interruption may potentially be catastrophic. Therefore, providing protection and failure recovery is important to the continuing development of network virtualization. There are two approaches to build resilient VNs. First, the substrate network could provide protection from network failure. However, substrate protection may be slow in responding to network failures. Alternatively, the individual VN manager could also perform quick recovery for those affected VNs. However, providing resilience to VNs is challenging since multiple virtual links/nodes are considered concurrently. These two options could also be coordinated to achieve even better resilience. It will be interesting to investigate both options and the interaction of these two.

- Distributed VN management: The idea of a VN manager is very flexible. There are different options for the VN manager: 1) centralized manager where a single manager controls and maintains all VNs in the network. 2) Disseminated VN manager, where there are multiple VN managers and each manager manages a certain number of VNs

(*e.g.,* VN manager in CA is responsible for all VN requests coming from CA). 3) Distributed VN manager, where a single VN is requested from and controlled by multiple coordinating managers. Each manager controls part of the VN. Therefore, distributed VN managers needs to communicate with each other. Examples of disseminated and distributed VN manager are shown in Figure 55. We can investigate different design options for VN manager.



(a) Disseminated VN manager (b) Distributed VN manager

Figure 55: Control plan architecture of VN managers

- More intelligent VN manager: In the VN assignment problem we considered before, a user submits the VN topology to the VN manager, which in turn discovers and allocates the substrate resources for the request VN. However, in some cases, an ordinary user may only have a traffic matrix and some constraints (cost, performance) without a clear idea of what VN topology they need. Therefore, a more intelligent VN manager could first design the VN topology for the user and then perform VN assignment, control and maintenance. Therefore, given the substrate network information, how to design a virtual network to satisfy certain traffic demand and constraints is another interesting problem.

7.2.3 Multihomed overlay network

The MON design problem studied in this thesis is performed at large time scales, namely months. Therefore, we only focus on static performance metrics such as propagation delay based RTT. At the same time, there are more dynamic events, such as fluctuations in available bandwidth and loss rate, network failures. They occur in smaller time scales in the

native network. To handle these events, the OSP could implement certain dynamic routing algorithms in the MON network such that MON paths to each important destination is selected dynamically based on current network situation. Since the MON network combines both multihoming and overlay network, MHO routing includes both routing within an overlay and multihoming path selection. It is interesting to investigate the dynamic routing in the MON scenario and consider the possible interaction/coordiantion between static MON design and dynamic MON routing.

REFERENCES

[1] InterNAP, http://www.internap.com/solutions/routecontroll.

[2] http://merit.edu/mail.archives/nanog/2004-08/msg00269.html.

[3] "Iperf." http://dast.nlanr.net/Projects/Iperf/.

[4] "Collocation: More than just a real estate play," *Equinix white paper*, 2001.

[5] Akella, A., Chawla, S., and Seshan, S., "Mechanisms for Internet routing: A study," Tech. Rep. CMU-CS-02-163, Carnegie Mellon University, 2002.

[6] Akella, A., Maggs, B., Seshan, S., Shaikh, A., and Sitaraman, R., "A measurement-based analysis of multihoming," in *Proc. ACM SIGCOMM*, pp. 353–364, 2003.

[7] Akella, A., Pang, J., Maggs, B., Seshan, S., and Shaikh, A., "A comparison of overlay routing and multihoming route control," in *Proc. ACM SIGCOMM*, 2004.

[8] Akella, A., Seshan, S., and Shaikh, A., "An empirical evaluation of wide-area Internet bottlenecks," in *Proc. ACM SIGCOMM Conference on Internet Measurement*, pp. 101–114, 2003.

[9] Andersen, D. G., Balakrishnan, H., Kaashoek, M. F., and Morris, R., "Resilient overlay networks," in *Proc. ACM SOSP*, 2001.

[10] Andersen, D. G., Balakrishnan, H., Kaashoek, M. F., and Rao, R., "Improving web availability for clients with monet," in *Proc. Symposium on Networked Systems Design and Implementation (NSDI)*, 2005.

[11] Andersen, D. G., Snoeren, A. C., and Balakrishnan, H., "Best-path vs. multi-path overlay routing," in *Proc. ACM SIGCOMM conference on Internet measurement*, pp. 91–100, 2003.

[12] Anderson, E. J. and Anderson, T. E., "On the stability of adaptive routing in the presence of congestion control," in *Proc. IEEE INFOCOM*, 2003.

[13] Aspnes, J., Azar, Y., Fiat, A., Plotkin, S., and Waarts, O., "On-line load balancing with applications to machine scheduling and virtual circuit routing," in *Proc. ACM symposium on Theory of computing*, pp. 623–631, 1993.

[14] Azar, Y., Broder, A. Z., and Karlin, A. R., "On-line load balancing," in *Proc. IEEE Symposium Foundations of Computer Science*, pp. 218–225, 1992.

[15] Azar, Y., Kalyanasundaram, B., Plotkin, S. A., Pruhs, K., and Waarts, O., "Online load balancing of temporary tasks," in *Proc. Workshop on Algorithms and Data Structures*, pp. 119–130, 1993.

[16] Bauer, D., Rooney, S., Scotton, P., Iliadis, I., and Buchegger, S., "The performance of measurement-based overlay networks," in *QofIS*, (Switzerland), 2002.

[17] Bertsekas, D. and Gallager, R., *Data Networks*. Prentice-Hall, 2 ed., 1992.

[18] Byers, J., Considine, J., Mitzenmacher, M., and Rost, S., "Informed content delivery across adaptive overlay networks," in *Proc. ACM SIGCOMM*, pp. 47–60, August 2002.

[19] Cahill, A. J. and Sreenan, C. J., "An efficient CDN placement algorithm for the delivery of high-quality tv content," in *Proceedings of the ACM international conference on Multimedia*, 2004.

[20] Carter, R. L. and Crovella, M. E., "Server selection using dynamic path characterization in wide-area networks," in *Proc. IEEE Infocom*, 1997.

[21] Cha, M., Moon, S., Park, C.-D., and Shaikh, A., "Placing relay nodes for intra-domain path diversity," in *Proc. IEEE INFOCOM*, 2006.

[22] Chen, Y., Bindel, D., Song, H., and Katz, R. H., "An algebraic approach to practical and scalable overlay network monitoring," in *Proc. ACM SIGCOMM*, 2004.

[23] CoMon. http://comon.cs.princeton.edu/.

[24] Dhamdhere, A. and Dovrolis, C., "ISP and egress path selection for multihomed networks," in *Proc. IEEE INFOCOM*, 2006.

[25] Eriksson, H., "MBONE: the multicast backbone," *Communications of the ACM*, vol. 37, no. 8, pp. 54–60, 1994.

[26] F5 Networks, "BIG-IP link controller." http://www.f5.com/f5products/products/bigip.

[27] Fang, Q., Cobb, J., and Leiss, E., "A pre-selection routing scheme for virtual circuit networks," in *Proc. IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS)*, 2000.

[28] Ferreira, P. M., "Implications of decreasing bandwidth price on allocating traffic between transit and peering agreements," Master's thesis, Massachusetts Institute of Technology, 2002.

[29] Gao, R., Dovrolis, C., and Zegura, E. W., "Avoiding oscillations due to intelligent route control systems," in *Proc. IEEE INFOCOM*, 2006.

[30] Goldenberg, D. K., Qiu, L., Xie, H., Yang, Y. R., and Zhang, Y., "Optimizing cost and performance for multihoming," in *SIGCOMM*, 2004.

[31] Gupta, A., Kleinberg, J., Kumar, A., Rastogi, R., and Yener, B., "Provisioning a virtual private network: a network design problem for multicommodity flow," in *Proc. ACM symposium on Theory of computing (STOC)*, pp. 389–398, 2001.

[32] Han, J., Watson, D., and Jahanian, F., "Topology aware overlay networks," in *Proc. IEEE INFOCOM*, 2005.

[33] Haque, A. and Ho, P.-H., "Design of survivable optical virtual private networks (O-VPNs)," in *Proc. the 1st IEEE International Workshop on Provisioning and Transport for Hybrid Networks*, 2004.

[34] Hu, N. and Steenkiste, P., "Evaluation and characterization of available bandwidth probing techniques," *IEEE Journal of Selected Areas in Communications*, vol. 21, no. 6, pp. 879–894, 2003.

[35] hua Chu, Y., Rao, S. G., and Zhang, H., "A case for end system multicast," in *Proc. ACM SIGMETRICS*, pp. 1–12, 2000.

[36] IIAS, "Internet In a Slice." https://wiki.planet-lab.org/twiki/bin/view/PlanetLab/InternetInASlice.

[37] Jacobson, V., "pathchar: A tool to infer characteristics of Internet paths." ftp://ftp.ee.lbl.gov/pathchar/.

[38] Jain, M. and Dovrolis, C., "End-to-end available bandwidth: measurement methodology, dynamics, and relation with tcp throughput," *IEEE/ACM Transaction on Networking*, vol. 11, no. 4, pp. 537–549, 2003.

[39] Keromytis, A. D., Misra, V., and Rubenstein, D., "SOS: Secure overlay services," in *Proc. ACM SIGCOMM*, pp. 61–72, 2002.

[40] Kleinberg, J. M., *Approximation algorithms for disjoint paths problems*. PhD thesis, Massachusetts Institute of Technology, 1996.

[41] Kolliopoulos, S. G. and Stein, C., "Improved approximation algorithms for unsplittable flow problems," in *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, p. 426, 1997.

[42] Krishnan, P., Raz, D., and Shavitt, Y., "The cache location problem," *IEEE/ACM Trans. Networking*, vol. 8, no. 5, pp. 568–582, 2000.

[43] Labovitz, C., Ahuja, A., Bose, A., and Jahanian, F., "Delayed Internet routing convergence," in *Proc. ACM SIGCOMM*, pp. 175–187, 2000.

[44] Lao, L., Cui, J.-H., and Gerla, M., "Multicast service overlay design," in *Proc. International Symposium on Performance Evaluation of Computer and Telecommuni cation Systems (SPECTS)*, 2005.

[45] Lee, S.-J., Sharma, P., Banerjee, S., Basu, S., and Fonseca, R., "Measuring bandwidth between planetlab nodes," in *Proc. PAM 2005*, 2005.

[46] Li, B., Golin, M. J., Italiano, G. F., and Deng, X., "On the optimal placement of web proxies in the internet," in *Proc. IEEE INFOCOM*, 1999.

[47] Li, Z. and Mohapatra, P., "The impact of topology on overlay routing service," in *Proc. IEEE INFOCOM*, 2004.

[48] Li, Z. and Mohapatra, P., "QRON: QoS-aware routing in overlay networks," *IEEE JSAC*, vol. 22, no. 1, pp. 29–40, 2004.

[49] Ma, Q. and Steenkiste, P., "On path selection for traffic with bandwidth guarantees," in *Proc. IEEE ICNP*, pp. 191–202, 1997.

[50] Mah, B. A., "pchar: A tool for measuring Internet path characteristics." http://www.kitchenlab.org/www/bmah/Software/pchar/.

[51] Mao, Z. M., Rexford, J., Wang, J., and Katz, R. H., "Towards an accurate as-level traceroute tool," in *SIGCOMM*, 2003.

[52] Nakao, A., Peterson, L., and Bavier, A., "A routing underlay for overlay networks," in *Proc. ACM SIGCOMM*, pp. 11–18, 2003.

[53] Nortel Networks, "Alteon link optimizer."

[54] Oppenheimer, D., Patterson, D. A., and Vahdat, A., "A case for informed service placement on planetlab," Tech. Rep. PDN–04–025, PlanetLab Consortium, December 2004.

[55] Paxson, V. and S.Floyd, "Wide Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Transactions on Networking*, vol. 3, pp. 226–244, June 1995.

[56] Paxson, V., "End-to-end routing behavior in the Internet," in *Proc. ACM SIGCOMM*, pp. 25–38, 1996.

[57] Peterson, L., Shenker, S., and Turner, J., "Overcoming the Internet impasse through virtualization," in *Proc. ACM Workshop on Hot Topics in Networks (HotNets)*, 2004.

[58] PlanetLab. https://www.planet-lab.org/.

[59] Qiu, L., Padmanabhan, V. N., and Voelker, G. M., "On the placement of web server replicas," in *Proc. IEEE INFOCOM*, 2001.

[60] Qiu, L., Yang, Y. R., Zhang, Y., and Shenker, S., "On selfish routing in Internet-like environments," in *Proc. ACM SIGCOMM*, pp. 151–162, 2003.

[61] Radware, "Peer Director." http://www.radware.com/content/products/pd.

[62] Raghavan, P. and Thompson, C. D., "Provably good routing in graphs: regular arrays," in *Proc. ACM symposium on Theory of computing*, pp. 79–87, 1985.

[63] Rahul, H., Kasbekar, M., Sitaraman, R., and Berger, A., "Towards realizing the performance and availability benefits of a global overlay network," in *Proc. Passive and Active Measurements (PAM) conference*, 2006.

[64] Ratnasamy, S., Handley, M., Karp, R., and Shenker, S., "Topologically-aware overlay construction and server selection," in *Proc. IEEE INFOCOM*, 2002.

[65] Rewaskar, S. and Kaur, J., "Testing the scalability of overlay routing infrastructures," in *Proc. Passive and Active Measurements (PAM) workshop*, 2004.

[66] Ribeiro, V., Riedi, R., Baraniuk, R., Navratil, J., and Cottrell, L., "pathChirp: Efficient available bandwidth estimation for network paths," in *Passive and Active Measurement Workshop*, 2003.

[67] Ribeiro, V. J., Riedi, R. H., Baraniuk, R. G., Navratil, J., and Cottrell, L., "pathChirp: Efficient available bandwidth estimation for network paths," in *Proc. Passive and Active Measurements (PAM) workshop*, 2003.

[68] Roughan, M., Thorup, M., and Zhang, Y., "Performance of estimated traffic matrices in traffic engineering," in *Proc. ACM SIGMETRICS*, 2003.

[69] Roughgarden, T. and Tardos, E., "How bad is selfish routing?," *Journal of ACM*, vol. 49, no. 2, pp. 236–259, 2002.

[70] Saroiu, S., Gummadi, P. K., and Gribble, S. D., "SProbe: A fast technique for measuring bottleneck bandwidth in uncooperative environments," in *Proc. IEEE Infocom*, 2002.

[71] Savage, S., Anderson, T., Aggarwal, A., Becker, D., Cardwell, N., Collins, A., Hoffman, E., Snell, J., Vahdat, A., Voelker, G., and Zahorjan, J., "Detour: a case for informed Internet routing and transport," Tech. Rep. TR-98-10-05, 1998.

[72] Savage, S., Collins, A., Hoffman, E., Snell, J., and Anderson, T. E., "The end-to-end effects of Internet path selection," in *Proc. ACM SIGCOMM*, pp. 289–299, 1999.

[73] Sayal, M., Breitbart, Y., Scheuermann, P., and Vingralek, R., "Selection algorithms for replicated web servers," *SIGMETRICS Perform. Eval. Rev.*, vol. 26, no. 3, pp. 44–50, 1998.

[74] Seshadri, M. and Katz, R. H., "Dynamics of simultaneous overlay network routing," Tech. Rep. UCB//CSD-03-1291, University of California, Berkeley, 2003.

[75] Shi, S. and Turner, J. S., "Placing servers in overlay networks," in *Proc. International Symposium on Performance Evaluation of Computer and Telecommuni cation Systems (SPECTS)*, 2002.

[76] Spring, N., Mahajan, R., and Anderson, T., "Quantifying the causes of path inflation," in *Proc. ACM SIGCOMM*, 2003.

[77] Spring, N., Mahajan, R., and Wetherall, D., "Measuring ISP topologies with rocketfuel," in *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, (New York, NY, USA), pp. 133–145, ACM Press, 2002.

[78] Strauss, J., Katabi, D., and Kaashoek, F., "A measurement study of available bandwidth estimation tools," in *Proc. ACM SIGCOMM Conference on Internet Measurement*, pp. 39–44, 2003.

[79] Subramanian, L., Stoica, I., Balakrishnan, H., and Katz, R., "OverQoS: Offering QoS using overlays," in *First Workshop on Hop Topics in Networks (HotNets-I)*, October 2002.

[80] SWORD. http://www.cs.berkeley.edu/ davidopp/sword/.

[81] Szeto, W., Iraqi, Y., and Boutaba, R., "A multi-commodity flow based approach to virtual network resource allocation," in *Proc. GLOBECOM: IEEE Global Telecommunications Conference*, 2003.

[82] Tangmunarunkit, H., Govindan, R., and Shenker, S., "Internet path inflation due to policy routing," in *SPIE ITCOM*, 2003.

[83] Tao, S., Xu, K., Xu, Y., Fei, T., Gao, L., Guerin, R., Kurose, J., Towsley, D., and Zhang, Z.-L., "Exploring the performance benefits of end-to-end path switching," *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 418–419, 2004.

[84] Taylor, D. and Turner, J., "Towards a diversified Internet," November 2004.

[85] Wang, H., Xie, H., Qiu, L., Silberschatz, A., and Yang, Y. R., "Optimal ISP subscription for internet multihoming: Algorithm design and implication analysis," in *Proc. IEEE INFOCOM*, 2005.

[86] Wang, Z. and Crowcroft, J., "Quality-of-service routing for supporting multimedia applications," *IEEE Journal of Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, 1996.

[87] Wong, E., Chan, A. K. M., and Yum, T.-S., "A taxonomy of rerouting in circuit-switched networks," *IEEE Communications Magazine*, vol. 37, no. 11, pp. 568–582, 1999.

[88] Xie, H., Qiu, L., Yang, Y. R., and Zhang, Y., "On self adaptive routing in dynamic environments," in *Proc. IEEE ICNP*, 2004.

[89] Yalagandula, P., Sharma, P., Banerjee, S., Lee, S.-J., and Basu, S., "S^3: Scalable sensing service for planetlab." http://networking.hpl.hp.com/s-cube/PL/.

[90] Zegura, E. W., Calvert, K., and Bhattacharjee, B., "How to model an internetwork," in *Proc. IEEE Infocom*, 1996.

[91] Zhu, Y. and Ammar, M., "Algorithms for assigning substrate network resources to virtual network components," in *Proc. IEEE INFOCOM*, 2006.

[92] Zhu, Y., Dovrolis, C., and Ammar, M., "Dynamic overlay routing based on available bandwidth estimation: A simulation study," *Computer Networks Journal (Elsevier)*, vol. 50, pp. 739–876, 2006.