UNIVERSITY OF CALIFORNIA, SAN DIEGO

Fault Localization in Backbone Networks

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

by

Ramana Rao Kompella

Committee in charge:

Professor George Varghese, Co-Chair
Professor Alex C. Snoeren, Co-Chair
Professor Rene Cruz
Professor Bill Lin
Professor Stefan Savage

2007

The dissertation of Ramana Rao Kompella is approved, and it is acceptable in quality and form for publication on microfilm:

Co-Chair

Co-Chair

University of California, San Diego

2007

To Amma and Nannagaru

"Take up one idea. Make that one idea your life—think of it, dream of it, live on that idea. Let the brain, muscles, nerves, every part of your body, be full of that idea, and just leave every other idea alone. This is the way to success; that is the way great philosophers are produced."

-Swami Vivekananda

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I am indebted to George for convincing me to do a PhD in the first place when we first met at SwitchOn Networks. Without him, I would not have been able to get the inspiration to start my doctoral studies. Second, I express my deepest gratitude to Alex for taking me under his wing especially when George left to the bay area to achieve fame and fortune. I am extremely fortunate to have two advisors with entirely different perspectives about everything. Alex's penchant for perfection and attention to detail is exemplary. George's enthusiasm for thinking and research and his ability to come up with cool solutions to real problems is truly remarkable. Even if I have learnt a small fraction of these bag of qualities, I believe I would be successful. Apart from George and Alex, I have had the fortune of interacting with the coolest young faculty in Stefan, Geoff and Amin in any systems and networking groups in the world. Special thanks are due to Julie Conner who is perhaps the best and most caring graduate coordinator you can ever ask for.

I would like to thank Jen at AT&T Labs–Research and Albert at Microsoft Research for mentoring me throughout my thesis. Besides the formal interaction twice as a summer intern as well as continuous collaboration even while at UCSD, I am fortunate to know two of the best people in this world. Jen's intensity and ability to handle five different projects at the same time is truly amazing! Albert's unique combination of vision and leadership while still somehow managing to remain so accessible is unparalleled. I would also like to thank Jennifer Rexford for great interaction and for being my co-author during my internship at AT&T Research. Many thanks are due to Aman, Carsten, Zihui, David Applegate (for volleyball during summer) and many other co-interns such as Vyas, Harsha, Sachin, Aditya, Kashi to name a few.

I can't thank enough the large number of people I have closely interacted during my entire four and half year stay at UCSD; I would still like to mention a few that stand out. A constant source of inspiration from every direction has been

Sumeet, who had been my apartment mate as well as my lab mate for two years. His long political rants as well as company during night-outs in the old AP&M building will be missed thoroughly. I will also miss a great set of friends in Sriram and Ishwar, who made my stay at Mesa a pleasant one, and with whom I used to play tennis during weekends and hit the gym once in a while. My sincere thanks to many of lab mates including Cristi, Florin, Michael and Justin (in old AP&M building), and Yuchung, Jeannie, Chris, Marvin, Marti, Meg, Alper, Kiran (in the new building). I also thank Kashi and Priya for frisbee and darts and general corridor talk. Finally, I would like to thank Barath, Alvin, David, Chip, Diwaker and all the others that make the sysnet group at UCSD truly a remarkable one.

Last but not the least, none of this would have been possible without the constant support from my mom, dad, my brother Dr. Sastry Kompella. Finally, my life has changed forever after I got married to my lovely wife, Swetha, who endured extensive stretches of absence at home, especially during deadlines. I can't express in words the amount of gratitude I sincerely owe my family.

I would like to acknowledge the various set of papers primarily investigated by the dissertation author, from which this thesis derives material from:

Chapter IV is based on the paper titled "IP Fault Localization via Risk Modeling", that appeared in the *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation*, held at Boston in May 2005, which is joint work with Jennifer Yates (at AT&T Labs – Research), Albert Greenberg (at Microsoft Research), and Alex C. Snoeren.

Chapter V is based on the paper titled "Detection and Localization of Network Black holes", appeared in the *Proceedings of the Infocom* , held at Anchorage in May 2007, which is joint work with Jennifer Yates, Albert Greenberg and Alex C. Snoeren.

Chapter VI is based on the paper titled "Cross-layer visibility as a service", that appeared in the *Proceedings of the Fourth ACM Workshop on Hot Topics in Networks*, held at College Park, MD, in Nov 2005, co-authored with

Jennifer Rexford (at Princeton University), Jennifer Yates, Albert Greenberg and Alex C. Snoeren.

Chapter VII is based on joint work with George Varghese and Alex C. Snoeren. The material has not yet been published.

VITA

1995–1999	B.Tech., Indian Institute of Technology, Bombay.
1999–2001	M.S., Stanford University.
2000–2001	Software Engineer, PMC Sierra Inc.
2001–2004	Member of Technical Staff, Chelsio Communications.
2003–2007	Ph.D., University of California, San Diego.

PUBLICATIONS

1. Ramana Rao Kompella, Alex C. Snoeren, Jennifer Yates, Albert Greenberg, "Detection and Localization of Network Blackholes", in Proceedings of IEEE Infocom, Alaska, USA, April 2007 (Infocom 2007).

2. Ramana Rao Kompella, Sumeet Singh, George Varghese, "On scalable attack detection in the Network", in IEEE/ACM Transactions on Networking, 15(1), pp. 14–25, February 2007 (ToN 2007).

3. Ramana Rao Kompella, Albert Greenberg, Jennifer Rexford, Alex C. Snoeren, Jennifer Yates, "Cross-layer visibility as a Service", in Proceedings of the 4th ACM Workshop on Hot Topics in Networks, College Park, MD, November 2005 (HotNets 2005).

4. Ramana Rao Kompella, Cristian Estan, "The Power of Slicing in Internet Flow Measurement", in Proceedings of the 3rd ACM/USENIX Internet Measurement Conference, Berkeley, CA, October 2005 (IMC 2005).

5. Ramana Rao Kompella, Jennifer Yates, Albert Greenberg, Alex C. Snoeren, "IP Fault Localization via Risk Modeling", in Proceedings of the 2nd Symposium on Networked Systems Design and Implementation, Boston, May 2005 (NSDI 2005).

6. Ramana Rao Kompella, George Varghese, "Detecting DoS attacks in the Network", in Proceedings of the 2nd ACM/USENIX Internet Measurement Conference, Taormina, Sicily, Italy, October 2004 (IMC 2004).

7. Ramana Rao Kompella, George Varghese, "Reduced state Fair Queuing for Core and Edge routers", in Proceedings of the 14th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video, Kinsale, County Cork, Ireland, June 2004 (NOSSDAV 2004).

8. Ishwar Ramani, Ramana Rao Kompella, Sriram Ramabhadran, Alex C. Snoeren, "Efficient Cooperative Scheduling in 802.11 Wireless Networks", UCSD Technical Report CS2005-0830, La Jolla, CA, July 2005 (UCSD-TR 2005)

9. Ramana Rao Kompella, Sriram Ramabhadran, Ishwar Ramani, Alex C. Snoeren, "Cooperative Scheduling via pipelining in 802.11 Wireless Networks", in the Proceedings of ACM SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis, Philadelphia, PA, August, 2005 (SIGCOMM E-WIND 2005).

10. Ramana Rao Kompella, Alex C. Snoeren, "Practical Lazy Scheduling in Wireless Sensor Networks", in Proceedings of the First Conference on Embedded Sensor Systems, November 2003, Los Angeles, California, USA (SENSYS 2003).

11. Ramana Rao Kompella, Alex C. Snoeren, "SPARTA : Scheduled Rate and Power Adaptation in Wireless Networks", *Extended Abstract* in ACM SIGCOMM, August 2003, Karlsruhe, Germany (SIGCOMM 2003).

12. Sundar Iyer, Ramana Rao Kompella, Nick McKeown, "Designing Buffers for Router Line Cards", Stanford University HPNG Technical Report, TR02-HPNG-031001, Stanford. Also *under review* in IEEE/ACM Transactions on Networking, Nov, 2002 (ToN 2002).

13. Sundar Iyer, Ramana Rao Kompella, Nick McKeown, "Analysis of Memory Architecture for Fast Packet Buffers " in the IEEE Workshop on High Performance Switching and Routing, Dallas, May 2001 (HPSR 2001).

14. Sundar Iyer, Ramana Rao Kompella, Nick McKeown, "Techniques for Fast Packet Buffers", Gigabit Networking Workshop, Anchorage, Alaska, April, 2001 (GBN 2001).

15. Sundar Iyer, Ramana Rao Kompella, Ajit Shelat, "ClassiPI : An Architecture for Fast and Flexible Packet Classification", in IEEE Network Special Issue on Fast IP Packet Forwarding and Classification for Next Generation Internet Services", March-April 2001 (IEEE NETWORK 2001).

ABSTRACT OF THE DISSERTATION

Fault Localization in Backbone Networks

by

Ramana Rao Kompella
Doctor of Philosophy in Computer Science
University of California, San Diego, 2007
Professor George Varghese, Co-Chair
Professor Alex C. Snoeren, Co-Chair

Automated, rapid and effective fault management is a central goal of large operational IP networks. Yet, today's networks suffer from a wide and volatile set of failure modes, where the underlying fault proves difficult to be detected and localized. In this dissertation, we introduce a fault localization methodology based on the use of risk models. At a high level, risk modeling involves constructing a bi-partite dependency relationship between a set of observable failure symptoms and associated root causes. It then uses novel fault-localization algorithms that use the set of observed failure symptoms and the constructed risk models to output a set of candidate root causes that best explain the symptoms.

Using observations from monitoring data commonly available today in ISP networks, we apply risk-modeling methodology to two different fault-localization problems—IP link and black hole localization—commonly observed in practice. For these two fault-localization problems, we have designed, implemented, evaluated and deployed systems in a real tier-one ISP network. Our experience indicates that risk modeling is effective in narrowing down the set of root causes of failures significantly, thus assisting network operators respond quickly to common failure modes.

While our systems indicate tremendous promise in the risk-modeling approach, still, risk modeling is an indirect inference mechanism born out of necessity, especially in situations where direct isolation mechanisms do not exist. Thus, we propose composition-based architecture called m-Plane that utilizes specialized router primitives to directly isolate the location of failures that affect traffic. In addition to monitoring connectivity problems, our architecture can also generalize to localizing other end-to-end performance degradations such as delay and loss.

Chapter I

Introduction

Over the past few decades, the Internet has become an integral part of our lives. The Internet today is a common platform used by people all around the globe for data communications, web browsing, online shopping, etc., as well as by enterprises that depend on it for conducting their businesses (e.g., financial transactions, e-commerce). However, the Internet still lacks the kind of reliability and robustness we expect from critical infrastructure. Network events, such as equipment failures, often cause disruptions in service, or even complete loss of connectivity between end hosts, causing frustration to end users and financial damage to enterprises. It is therefore not surprising that Internet service providers (ISPs) desire to quickly eliminate such unwanted disruptions in their network, or at least reduce the impact of such events whenever they happen.

However, IP networks are surprisingly hard to manage. First, barring a few statistics counters, there is no intrinsic support in the architecture for efficient network monitoring and fault isolation. Second, new services such as voice and video require strict performance guarantees, while IP networks have been designed to be best effort. Third, IP networks are constantly evolving in size (in terms of number of nodes), technology (e.g., ultra long-haul in optics) and protocols (e.g., MPLS). This evolution in technology often outpaces the development and deployment of associated monitoring infrastructure. Finally, many pernicious faults are caused by complex interactions across layers—which are traditionally studied in isolation—and, hence, are hard to debug and diagnose.

In the face of these challenges, service providers often resort to maintaining a large number of network engineers to respond to connectivity and performance failures. For instance, British Telecom (BT) reports that it's capital expenditure for the fiscal year 2006 was 2 Billion Pounds [2], while operating expenses were 17 Billion Pounds, of which BT estimates 5 Billion Pounds just in staffing costs. Thus, operating expenditure of IP network providers is significantly higher than capital expenditures. Of course, high operating expenditure does not automatically translate to quick response to failures, since manual processes often tend to be too slow for many practical failure scenarios.

One way to reduce the operating expenditure is to focus on improving device reliability (resulting in higher capital expenditure); if devices do not fail (or fail less often), networks can be managed with a smaller work force. Unfortunately, studies estimate that 80% of failures are caused by people and process errors [13, 20]. For such failures, improving device reliability is not helpful. Besides, complexity added to devices for robustness often exposes them to new fragilities, thus leading to the complexity-robustness spiral [25]. Therefore, improving device reliability alone is not going to be sufficient, and, in many cases, even results in new and more complicated failure scenarios.

Given that failures are inevitable, service providers need mechanisms for effective *fault management*. The essential problem of fault management is to detect, localize, diagnose and ultimately correct any condition that degrades network performance. To assist fault management, many network elements detect and generate alarms on failures, which are then used to perform diagnosis and repair. For example, an IP router is often programmed to generate an alarm whenever the router-to-router keep-alive messages that monitor IP link-level connectivity are lost. However, an IP link consists of a number of components: a line card at origin router, a physical fiber to an optical amplifier, a second segment of fiber from the optical amplifier to the destination router's line card. While in this example, there are only a few components, IP links in practice can consist of greater than a hundred optical components. Operators first need to determine which among these components caused the link failure—a step commonly referred to as *fault localization*—before they can diagnose the root cause of the failure. For example, operators need to localize the failure to an optical amplifier before determining that the root cause is a faulty laser within the optical amplifier.

Operational experience shows that the total turn-around time of a failure (from detection to repair) is often dominated by the amount of time it takes to localize the failure. Once the fault is localized, typically the failure can be repaired relatively quickly; in many cases, the actual root cause can be diagnosed and repaired offline. Thus, the main focus of this dissertation is *fast and automated fault localization in backbone networks.* We describe the architecture of our fault localization systems based on risk modeling, that we have deployed in a tier-one backbone network. Based on our experience with commonly observed fault localization problems, we also focus on clean-slate architectural mechanisms that make use of new router-level primitives to assist fault localization.

The remainder of this chapter is organized as follows. First, we review fault management and, in particular, fault localization in the next section. We then outline challenges associated with fault localization before summarizing the contributions of the dissertation in the following section.

I.A Fault management in backbone networks

Operational backbone networks today lack intrinsic robustness; serious faults and outages are not infrequent. To appreciate why this is so, it may help to consider a network operator tasked with fault management. After much effort, network hardware has been designed and implemented, the protocols controlling the network have been designed (often in compliance with published standards), and the associated software implemented. In accord with the network architecture, the network elements have been deployed, connected, and configured. Yet, these are all tremendously complex endeavors, carried out by multiple teams at rapid pace, involving a large and distributed software component, producing an artifact that will face an operational environment far richer in behavior than can ever be approximated in a lab.

Errors *will* be introduced at each stage of network definition and go undetected despite best practices in design, implementation, and test. External factors, including bugs of all types (memory leaks, inadequate performance separation between processes, etc.) in router software and environmental factors such as denial-of-service attacks and routing events originating in peer networks significantly raise the level of difficulty. It is the task of fault management to cope with the result, continually learning of and dealing with new failure modes in the field. Thus, a large amount of time and effort is spent by service providers in fault management.

Fault management in operational backbone networks is challenging in practice. These networks are large, geographically distributed, and constantly evolving with complex hardware and software artifacts. To appreciate the complexity better, consider a typical tier-one backbone network: It consists of about 1,000 routers from different vendors, with different features, and acting in different roles in the network architecture, supported by access and core optical transport networks involving more than two orders of magnitude more network elements. The layered structure of backbone networks is intended to help contain complexity within simple well-defined abstractions; in practice, however, layering often gives rise to additional, complex failure modes involving cross-layer interactions which are hard to debug.

For example, IP networks are overlaid on optical networks; an IP link is implemented as a path through a set of optical components, some of which are shared across multiple IP links (in a one-to-many, many-to-one fashion). Similarly, in many tier-one ISP networks, there may be an intermediate multi-protocol label switching (MPLS) layer, wherein IP packets are transported via label-switched paths (LSPs), which in turn are established using IP routing protocols such as open shortest path first (OSPF). Many of these LSPs share individual IP links. It is possible to run IP and MPLS routing and forwarding concurrently on the same label-switch router. Moreover, multiple virtual private networks (VPNs) may be overlaid on top of the MPLS topologies, with paths between IP interfaces on multiple VPNs traversing the same label-switched path.

Thus, complicated dependencies exist between various layers in the network, often giving rise to unforseen failure modes difficult to diagnose. Often, however, a certain amount of resiliency is already built into the network, in order to automatically recover from many common failure modes such as link failures. For example, routing protocols (e.g., OSPF) typically respond to IP link failures by computing new paths that avoid the failed link, thus making sure that the network "self heals" and traffic is not disrupted. Additionally, because it is part of the network design, this temporary fix to the problem is essentially free of cost to the service provider.

However, the service provider still needs to apply a *permanent* fix to the problem. Otherwise, the chances of another failure occurring before the original failure is repaired increases. Further, the more the time it takes to repair, the higher the probability that the second one occurs before repair is complete. Therefore, the network operator needs to quickly isolate and repair the problem. In some cases, the network itself can fail to detect and self heal; thus, many customers can be disconnected for extended periods of time, until the problem is repaired. In this scenario, it is even more important for network operators to quickly detect, localize and repair the problem.

Thus, the essential components of fault management include detection, localization and repair as shown Figure I.1. Specific mechanisms (e.g., network probes) are instrumented in (or outside) the network to *detect* faults and generate associated alarms. A fault is typically characterized by monitoring traffic, either

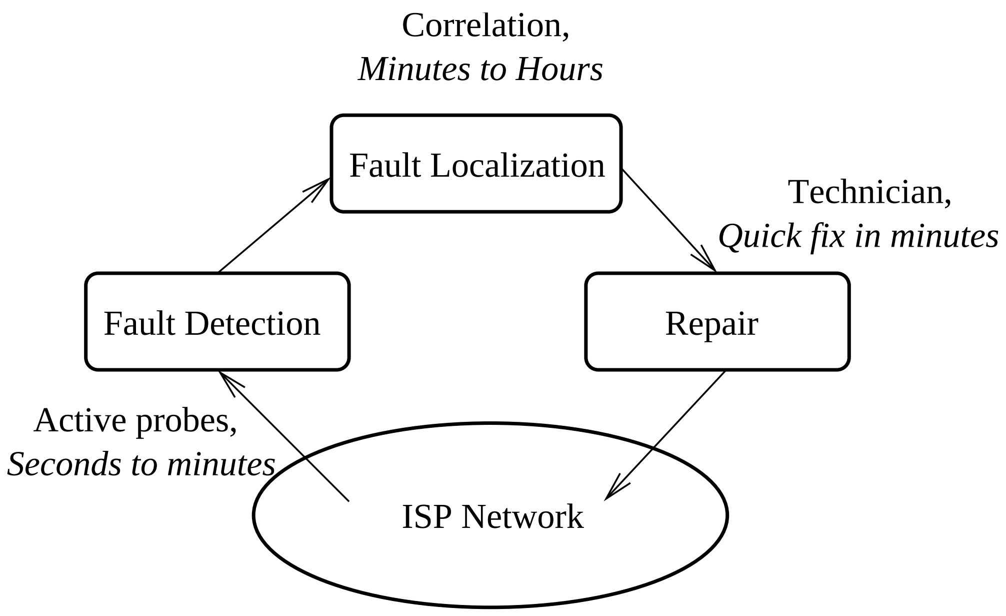


Figure I.1: Three steps in fault-management: The first step involves detection of a fault. The second step involves fault localization—identifying where exactly a fault happened. Finally, once the fault is localized, a technician can permanently fix the problem.

passively by observing existing traffic in the network, or actively by injecting synthetic traffic and observing its performance characteristics. Lower layers even use physical symptoms to detect faults, such as signal quality or loss of signal at the optical interface.

While alarms report almost instantaneously (within a few seconds) that a fault occurred, they frequently do not directly indicate the root cause of the failure. Therefore, the second step in fault management involves *localization* to determine where the fault happened. The time required to localize the fault can range anywhere from a few hours to days. This localization step is the primary reconnaissance to a final—and often (necessarily) manual—step of actually diagnosing the root cause of the failure. Once the fault is diagnosed, the third step involves a technician that *repairs* the failed component, typically by replacing it with a new component within minutes. Therefore, fast and automated fault localization can significantly reduce the outage or failure duration.

I.B Challenges in fault localization

There are primarily two challenges that form the basis of this dissertation: First, typical monitoring systems only detect failures but do not indicate root cause, thus requiring indirect inference mechanisms to infer the root causes. Second, there are many operational inefficiencies in the monitoring systems that complicate the task of inference. We explain these two challenges briefly in the following subsections.

I.B.1 Indirect inference

Network elements are designed to generate alarms for many common failure modes, such as IP link failures. However, these alarms do not automatically report the exact location of the failure. For example, an optical amplifier failure triggers multiple alarms at IP and optical layers. But none of these alarms directly indicate that it was an optical amplifier failure; they only indicate that a link failed. Of course, if they did indicate the root cause of the failure, there would be no need for an intermediate fault-localization step in the first place.

In some catastrophic failure cases, the network elements even fail to detect and generate alarms. In such situations, network operators monitor failures by injecting synthetic probes into the data traffic and observing the reachability and/or other performance characteristics of the probe traffic. Given that these probes treat the network as a black box, they typically detect, but fail to isolate the location of the failures. Thus, fault localization again requires indirect mechanisms to infer the root cause.

Network operators often use some form of correlation, particularly for faults that affect multiple layers. At a bare minimum, correlation is used to group together alarms that are typically generated due to the same root cause. In many backbone networks, however, operations are naturally managed according to layers, with the alarms and other monitoring data housed in separate work groups across geographically diverse locations within the same domain, or even across domains (e.g., different companies managing different layers). For instance physical-layer faults may be handled by a dark-fiber provider, while IP-layer faults are managed by the ISP itself. In such cases, we cannot correlate alarm data across layers easily.

Even when we can correlate alarms across layers, determining causality remains difficult, because layers are interdependent, with faults in one affecting the other and vice-versa. For example, IP-layer link failures can cause the optical lasers to shutdown as well, thus causing a loss of signal (LoS) alarm at the optical layer. Similarly, an optical amplifier failure can cause an alarm generated by the router that the IP link is down. When one observes both the LoS alarm at optical layer as well as IP link down alarm at IP layer, it is not clear what caused the failure or even in which layer the failure lies.

I.B.2 Operational inefficiencies

The highest priority job of a router during failures is to flood the link-failure information to the rest of the routers in the network and compute new paths that avoid the failed link to various destinations. Routers, therefore, often use unreliable messaging protocols such as UDP—instead of reliable protocols that require more CPU and memory resources—to transmit alarms; hence, alarms may or may not reach the operator.

Often, the databases that govern the associations across layers (e.g., IP and optics) are maintained by humans and, as such, are prone to errors. For example, a technician performing maintenance may reroute an optical circuit, but forget to log the change in the database. In such cases, the databases can drift significantly from reality making even inference hard.

I.C Thesis contributions

Confounded by the various challenges in fault localization, today's service providers often use a combination of *ad hoc* proprietary problem-specific mechanisms guided by experience and wisdom for fault localization, that are either too slow or labor-intensive. Thus, in this dissertation, our primary goal is to address these challenges and provide a generic framework for automated fault localization in backbone networks with existing monitoring mechanisms. While this framework is effective in the short term, it is important to design and develop long-term clean-slate architectural primitives to make fault localization, and performance management, a first-class entity in network elements. Thus, a secondary goal of this dissertation is to identify fundamental mechanisms that can assist performance management in backbone networks.

This dissertation makes the following contributions in pursuit of our goals.

- *Risk modeling.* We develop a general fault-localization methodology based on risk-modeling. Intuitively, risk modeling involves creating bipartite dependency relationships between a set of observable symptoms associated with various root causes of failures. We abstract the fault-localization problem as finding the minimum set cover in a bipartite graph that represents the dependencies between symptoms and root causes—an NP-hard problem in general. We apply the risk-modeling approach to two fault-localization problems observed in practice—IP fault localization (Chapter IV) and MPLS black hole localization (Chapter V). For these problem domains, we design two new algorithms SCORE and MAX-COVERAGE based upon a greedy approximation to finding the minimum set cover. Further, we build and evaluate two systems that are based on the above algorithms to assist network operators in localizing faults in these failure scenarios.
- *Cross-layer visibility.* In our experience with the above systems, we found that IP and optical cross-layer association databases, which are used widely

in many network management tasks including fault localization, are typically filled with errors. While a natural way to obtain accurate cross-layer associations is by fattening the interfaces between layers, we argue that such an approach can be hard to achieve due to a variety of reasons including complexity, interoperability and security. Therefore, we propose an architecture (in Chapter VI) that does not alter the current layer boundaries, instead maintains accurate dependencies by joining different databases that can evolve over time both in accuracy as well as timeliness.

- *Composition-based measurement architecture.* We suggest novel router primitives to allow direct localization of faults, and, in general, any type of performance degradation in the network. We also propose a new measurement architecture, called m-Plane (in Chapter VII), that allows composition of end-to-end path metrics directly from individual router- and link-level measurements reported by the routers, thus improving the scalability of active probes. While clean-slate design of m-Plane is relatively straightforward, incremental deployment is not. Therefore, we design mechanisms for incrementally upgrading only few routers in the network. Using Rocketfuel topologies [84], we show the benefit of incremental deployment in terms of probe bandwidth.

I.D Thesis organization

The organization of the dissertation is as follows. First, Chapter II presents the necessary Internet architecture background and an overview of the various network management tasks. The next chapter (Chapter III) outlines a general fault-localization methodology consisting of three basic steps. The first step consists of detection of failure symptoms using a variety of monitoring mechanisms, some of which network elements today already employ. The second step involves creation of a risk model that embeds the dependencies between various symptoms observed and their associated root causes. Finally, the third step consists of a localization algorithm that outputs a likely set of root causes that can explain the given set of observed symptoms.

In Chapter IV and Chapter V, we discuss the application of the risk-modeling methodology to two instances of failures commonly found in today's backbone networks—IP link fault-localization and MPLS tunnel fault-localization. These two chapters present the algorithms and system architecture details for the fault-localization systems we have designed, implemented and deployed in a real tier-I network. These chapters also present detailed evaluation of these systems with simulated and real failure data obtained from a real tier-I ISP.

From our experience with these two systems, there are two specific challenges—accurate cross-layer dependencies and scalable measurement—that we address in the next two chapters. In Chapter VI, we discuss, at a high-level, an architecture that allows automatic dependencies between symptoms and underlying root causes for the link-fault localization problem with high-levels of accuracy while allowing evolution in the network architecture. On the other hand, Chapter VII discusses router-level primitives for direct fault localization, in addition to reducing the measurement overhead for failure detection significantly. Finally, we conclude with a summary of the dissertation and some open challenges in Chapter VIII.

Chapter II

Background

In this chapter, we provide a brief background on the architecture of the Internet today, in particular focusing on concepts required to understand the rest of this dissertation. This chapter is organized as follows. First, we describe the architecture of today's service provider networks. We then outline some of the main network management functions, followed by a more detailed description of various mechanisms for effective performance management in these networks.

II.A Internet architecture

The Internet consists of a large set of autonomous systems (ASes) loosely coupled together to provide connectivity between various end hosts. Today, there are more than 20,000 ASes across the entire world [7]. Each AS is managed by a single administrative authority (an Internet service provider or a large enterprise customer) and owns the network within that domain. Each AS can be viewed as a graph with a set of nodes (routers) connected via edges (physical interconnections between routers). Packets can either originate (terminate) in the AS (e.g., from a DSL/Cable customer directly part of this AS) or enter (exit) from (to) other ASes, i.e., enterprises or other service providers. Packets transit between ASes through peering links that connect a router in one AS to one in another AS. Policies governing the cooperation between ASes through peering is often enforced

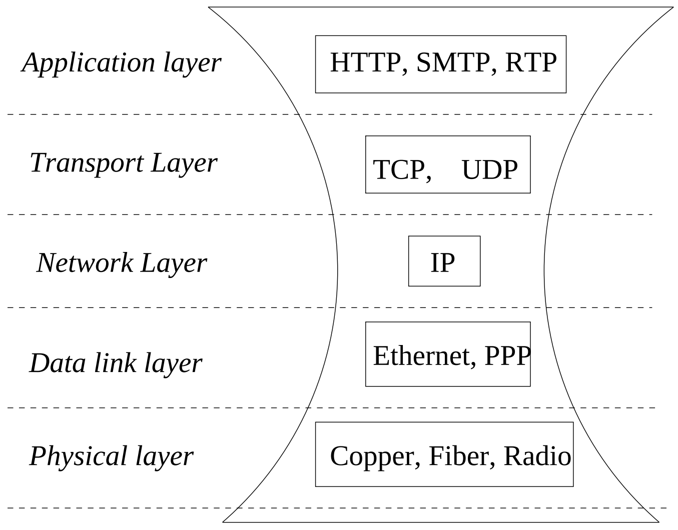


Figure II.1: Communication functions in the Internet are organized as a stack of layers, commonly referred to as the *Internet protocol* stack represented in the form of an hourglass.

via bi-lateral (or multi-lateral) contractual agreements.

In the Internet, all communication functions are organized in a hierarchical set of layers, with each layer performing a subset of functions required to communicate with another system. Each layer depends on the layer below for a specific function, while exposing a specific function to the layer above. Thus, overall communication functionality is broken down into these layers with specific interfaces between layers. Naturally, the architecture of an any particular AS is also structured in layers.

Layering in IP networks is often represented by the *hourglass* model as shown in Figure II.1, commonly referred to as the *Internet protocol stack*. At the bottom of the IP stack is the physical medium that is responsible for carrying bits of information from one system to another. The data-link layer allows for error detection and reliable transmission in units of frames while the network layer is responsible for addressing and forwarding packets between communication end systems that are not directly connected to each other. On top of the network layer is the transport layer that includes protocols such as transmission control protocol

(TCP) that provides the abstraction of a reliable, in-order byte stream and user datagram protocol (UDP), that provides an unreliable message communication abstraction. Several common applications such as email, web-use protocols such as simple mail transfer protocol (SMTP), hyper-text transfer protocol (HTTP) constituting the application layer.

The communication functions in layers above IP, i.e., the transport and application layers, are implemented on the hosts at both ends of communication. A typical ISP only deals with the layer below that provide mechanisms for transporting packets between end points. Therefore, we describe the architecture of ISP networks by focusing on the physical, data link and network layers in more detail, while referring the reader to [88, 92] for further details on layers above IP.

II.A.1 Physical layer

The lowest layer in the IP protocol stack is the physical layer. The physical layer consists of mechanisms to transmit bits reliably on a physical medium such as copper wire in local area networks (e.g., in Ethernet), or optical fiber in wide-area networks (e.g., synchronous optical network or SONET). The main goal of the physical layer is to provide the abstraction of a logical communication channel between two end points, often referred to as a link. In the context of wide-area backbone networks, links are also referred to as optical circuits, since a link is frequently a light path of certain wavelength between two end points transported via optical fibers.

Typical wide-area optical circuits, which are of importance in this dissertation, extend over long distances (hundreds to thousands of miles). Over these long distances, the optical signal both attenuates significantly as well as distorts with noise, making it difficult to reconstruct the signal at the other end. In order to address this issue, an *optical amplifier* is used every few hundred miles to amplify the optical signal. However, since the optical amplifier amplifies both the actual signal as well as noise, after every few such optical amplifiers, an *optical regener-*

ator, commonly referred to as a dense wavelength division multiplexing (DWDM) system, is required to recover the signal into digital domain and freshly regenerate the optical signal. Thus, an optical circuit consists of a series of one or more fibers, optical amplifiers, DWDM systems and many other components such as add-drop multiplexors, etc. A full treatise on all these components is outside the scope of this dissertation and we refer the reader to [72] for more details.

II.A.2 Data-link layer

While the physical layer provides basic mechanisms to transfer bit streams between two directly connected nodes, the data-link layer attempts to make the physical link reliable. Typical examples of protocols at the data-link layer include the point-to-point protocol (PPP), the high-level data link control (HDLC) for point-to-point communication (in wide-area networks) and Ethernet for local area networks. In order to provide a reliable link interface, most data-link protocols provide basic error detection by computing a checksum, cyclic-redundancy check (CRC) or at the least a parity bit of the message that is transmitted [92]. In automatic repeat-request (ARQ) protocols such as stop-and-wait, go-back-n, selective repeat [92], the sender and receiver use acknowledgements and timeouts to determine whether to resend a given frame.

In wide-area networks such as the Internet, communication is often between two end systems that are either not directly connected to each other or not part of the same local area network. In such cases, messages are transported over a number of data links, each functioning independently; the higher layers are not completely relieved of error detection and correction functionality [76]. The network layer which we describe next, provides the basic addressing and routing mechanisms required to forward packets in the Internet. Contrary to the data-link layer, however, the network layer is best effort; it does not provide guarantees that a packet injected into the network reaches the destination. Applications rely on the transport protocols such as TCP on top of the network layer for reliable data delivery.

II.A.3 Network layer

In the Internet, the primary network layer protocol used for communication is the Internet protocol (IP). Each host in the Internet is assigned an IP address—a unique 32-bit integer (often represented as four bytes in dotted decimal format, A.B.C.D) in IP version 4—that uniquely identifies a given interface. Each packet consists of an IP header that contains a source IP address and a destination IP address. Packets are forwarded from a source to a given destination by various routers along the path. In each of these routers, forwarding is typically destination based. For every packet, the router determines the next hop based on the destination IP address in the packet and a forwarding table that consists of a mapping between IP prefixes (e.g., 132.*.*.* that represents the range of addresses between 132.0.0.0 and 132.255.255.255) and the next hop. In the presence of multiple prefixes that match a given destination IP address, the router looks up the longest (or most specific) prefix to determine the next hop for forwarding.

Routers populate the forwarding tables using a combination of inter- and intra-domain routing protocols. These protocols typically exchange reachability information (at the granularity of a prefix) necessary to compute a path towards a given destination. For inter-domain routing, the border gateway protocol (BGP) [73] is used, while open shortest path first (OSPF) [65] or intermediate system - intermediate system (IS-IS) [14] is used for intra-domain routing protocols, commonly referred to as interior gateway protocols (IGPs).

An example is shown in Figure II.2. A customer network (say UCSD's network) connected to a given AS (say Internet2) is assigned an IP prefix (e.g., 137.110.222.*), and Internet2 advertises the reachability of this IP prefix to other peering ASes (say Verizon) using BGP, which in turn further advertise the route to other peering ASes (say AT&T) through Verizon. Using a combination of BGP and IGPs—BGP to determine the particular border router and IGPs to calculate

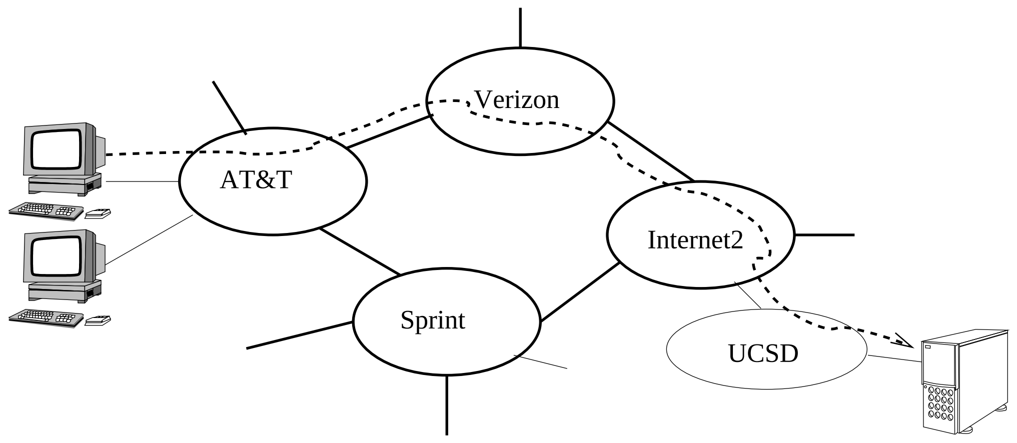


Figure II.2: A packet between two hosts in the Internet can cross multiple autonomous systems (ASes).

the shortest path to the border router—AT&T's routers are able to forward packets with destination address in UCSD's network directly to the border router in Verizon's network. Similarly, the routers in Verizon's network forwards the packets to Internet2, and Internet2's routers forward the packets to UCSD's network in turn. Once the packet reaches UCSD, the packet is forwarded to the destination server using a combination of IGP forwarding and data-link layer protocols such as Ethernet.

Thus, a combination of BGP and IGPs enable routers in the Internet to determine the next hop to forward a packet destined towards a given destination. While typical data communication between various end hosts is facilitated by the use of BGP and IGPs, today's ISP networks also provide inter-connection services to several geographically distributed offices belonging to a single enterprise. Such networks, referred to as virtual private networks (VPNs), while configured differently and in fact isolated from the public-domain traffic, they share the same underlying physical infrastructure (i.e., routers and links) for forwarding. Since VPNs are of importance in this dissertation, we describe them in greater detail next.

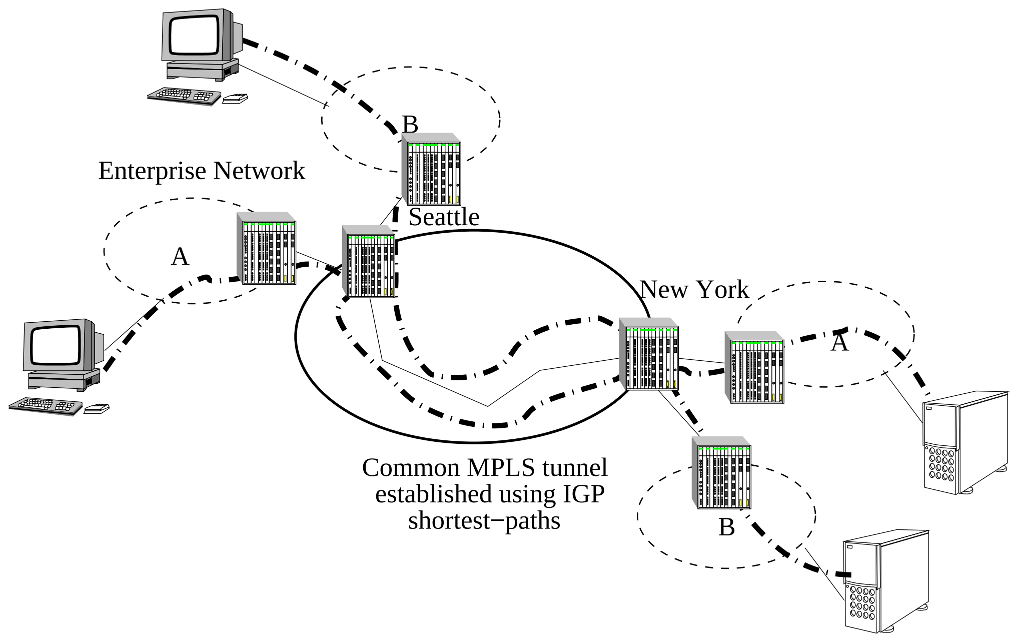


Figure II.3: Two enterprises A and B establish VPNs over an MPLS tunnel between Seattle and New York provider-edge routers. The MPLS tunnel itself is routed using the IGP shortest paths between these routers.

II.B Virtual private networks

ISPs today provide virtual private networks (VPNs) to securely connect geographically distributed offices that belong to a single large enterprise. While there are many ways in which VPNs can be administered, one of the most common is to use multi-protocol label switching (MPLS) [75].

MPLS provides a mechanism to create “tunnels” between two tunnel end points. Although, in theory these tunnel end points can belong to different service providers, today there is no unified inter-carrier mechanism to provide such a secure tunnel across providers. Therefore, typically MPLS tunnel end points lie within a given service provider. Multiple VPNs may be simultaneously carried within any given MPLS tunnel thus making the solution scalable. Packets in VPNs are forwarded using virtual routing and forwarding (VRF) tables that are distinct from the forwarding tables populated through conventional routing protocols.

An example of two enterprises using MPLS tunnels for their VPNs is shown in Figure II.3. In the figure, an ISP network with Seattle and New York provider-edge (PE) routers are shown with enterprise networks A and B with offices in Seattle and New York communicating over the ISP network. Since both these VPNs start and end at the same edge routers, they both can be overlaid on top of an MPLS tunnel between the Seattle and New York PE routers.

MPLS frames encapsulate IP packets with an additional header that contains one or more 20-bit MPLS labels in the form of a label stack. A router that is capable of label switching uses the outer-most label in the stack to determine the forwarding decision (the outgoing interface to transmit this packet) and one of push, pop or swap operations on the outermost label header. Push operation into the label stack is typically performed when an unlabeled packet that needs to be mapped on to an MPLS tunnel first enters the provider's network. Similarly, pop operation is performed when an MPLS packet exits the tunnel. Intermediate routers in the tunnel swap the incoming label with a new outgoing label (based on the state in the MPLS forwarding tables).

MPLS tunnels are set-up using either of two protocols, the label distribution protocol (LDP) [8] or reservation protocol (RSVP) [100]. In many providers, the exact path followed by a tunnel is often dictated by the underlying IGPs. For example, in the Figure II.3, the MPLS tunnel between Seattle and New York routers rides along the shortest path identified by the IGPs. Therefore, underlying IGP changes in topology triggers re-establishment of label switched paths between the tunnel end-points using either LDP or RSVP. Many other details of MPLS and administering VPNs using MPLS can be found in [21].

While we have so far described the basic Internet architecture as well as VPNs, there are several other facets of modern ISP networks that we have not addressed. For example, multi-homing [6] that allows enterprises to access the Internet from multiple service providers is another feature commonly provided to many customers. In general, providing these vast variety of services with different requirements requires careful management of network resources, thus presenting a signficant challenge to service providers. Not surprisingly, service providers spend a lot of effort and capital on various aspects of *network management* such as configuration, provisioning and performance management. We describe these various aspects of network management in more detail next.

II.C Network management

The essential task of managing a given AS lies with the particular service provider. In this section, we provide an overview of network management, in particular describing some of the most common network management tasks in more detail.

II.C.1 Network provisioning

A large ISP network today comprises of about a thousand routers, assisted by few hundred thousand network elements at physical and data-link layers. Network provisioning refers to the task of properly configuring these routers and other optical network elements to fulfill various objectives of the service provider. A real ISP network today offers many different types of services to various customers; proper provisioning is required to ensure that the network resources are configured to meet the service demands of customers, while ensuring conformation to the policies. This includes properly configuring the right protocols to use for various types of traffic (e.g., MPLS for VPNs), providing quality-of-service guarantees for particular types of traffic, and even choosing the right technology to use for a particular service (e.g., IP layer protection vs optical protection).

In addition to the initial configuration to boot-strap the network, a network operator needs to continuously perform planned maintenance and upgrade of both software as well as hardware components in the network. As technology evolves, the services offered by the particular ISP also mature in both variety as well as granularity, thus making provisioning a continuous task for a network operator. However, this evolution in technology is often at large time-scales (typically months to years), and, hence, is not the most pressing day-to-day task of a network operator.

II.C.2 Quality of service

The Internet has been designed mainly for providing a best-effort communication service with no performance guarantees. However, the transition from the experimental to commercial Internet requires performance guarantees. This is especially true due to the maturing of rich media such as voice and video on the Internet. In addition, VPNs require SLA guarantees from the service provider, that includes traffic isolation, bandwidth guarantees and so on. Providing quality of service (QoS) is therefore an important aspect of network management that a network operator has to deal with. While configuring the routers to provide service guarantees is an essential aspect of provisioning, service providers rely on several router-level mechanisms to provide the required QoS features in their network.

More than a decade of research has gone into designing efficient router-level mechanisms for providing performance guarantees in the Internet (e.g., [11, 22, 53, 80, 89]). In addition, for scalability reasons, these routers provide QoS at the granularity of classes instead of providing per-flow guarantees. From a user perspective, it is important to guarantee end-to-end QoS, not just at any given router or with in any given AS. Inter-domain QoS has proved to be difficult in practice, with the result that service providers typically confine themselves to providing QoS within the AS. In spite of all the research, we believe that QoS will continue to be an active area of research, more importantly, as service providers expand beyond the traditional best-effort service model. Similar to network provisioning, QoS aspects of network management affect a service provider at large time scales.

II.C.3 Traffic engineering

Traffic engineering is the art of mapping flows onto IP links, and is quite related to provisioning. A service provider performs traffic engineering to manage the installed capacity effectively and efficiently and to enhance end-user perceptions of network service quality while minimizing costs. Traffic engineering depends on having a set of performance objectives that guide the selection of paths, as well as effective mechanisms for the routers to select paths that satisfy these objectives.

While many mechanisms exist in the literature for effective traffic engineering (e.g., [27, 33, 44, 50, 103]), it continues to remain an active area of research, as new services evolve and new applications (e.g., peer-to-peer networks, IP television) are developed. While traffic engineering is typically performed on large time-scales, network operators also constantly adjust traffic in response to short-term unanticipated events such as flash crowds, denial-of-service (DoS) attacks or failures.

II.C.4 Attack mitigation

While the very success of the Internet is due to its open model in which any computer can send to any other computer, this openness also allows attackers to send malicious messages that can cause damage to other hosts and networks, sometimes at great cost. From a service provider perspective, even in the presence of router-level QoS mechanisms discussed above, congestion can occur at various locations in the network due to denial-of-service (DoS) attacks. DoS attacks and worms can result in huge volumes of traffic chewing up bandwidth and other valuable resources in the network. Hence, network operators often employ mechanisms in the network to detect and mitigate such attacks.

There is a large body of literature dealing with in-network mechanisms to thwart the effect of such attacks (e.g., [28, 42, 52, 60, 98, 97]). While attack detection and mitigation continues to remain an important challenge to network operators, many of the afore-mentioned mechanisms help operators deal with attacks effectively. In addition, such attacks only congest the network degrading the service quality due to their high volumes, but the network still remains connected in many cases. In contrast, failures are in many cases more fatal that bandwidth based attacks, since depending on the particular failure scenario, faults can completely partition the network; many customers can be out of network connectivity for a significant amount of time. Therefore, we believe that that *fault management* is perhaps the most (or at least one of the most) important aspects of network management. Given that fault management is the primary focus of this dissertation, we discuss the various aspects of fault management in detail in the next section.

II.D State-of-the-art fault management

The essential task of fault management is the detection, localization and correction of any performance degradation in the network. The first step in fault management is *fault detection*. To assist fault detection, many network elements (such as routers) are designed to continuously monitor certain behaviors (e.g., link connectivity) and raise an alarm in the event of a failure. These alarms, however, only indicate that a network element observed a deviation from normal behavior (e.g., link failure or probe packet loss); the actual fault could lie anywhere in the network (e.g., a downstream router reboot or optical amplifier failure). Thus, *fault localization* needs to be performed to identify (or or at least narrow down) the root cause of the failure. Once the root cause is obtained, an operator can take proper *repair* actions necessary to permanently fix the failure. We discuss these three steps, i.e., fault detection, localization and repair, in more detail in this section.

II.D.1 Fault detection

It is a fact of life that network elements fail. Not surprisingly, network elements are often equipped with basic mechanisms to monitor and observe failures and generate alarms whenever a failure condition is observed. The network elements are also designed to "self-heal" from failures by re-routing packets through other paths. Therefore, routers continuously monitor connectivity to their neighbors in order to check whether a link is functional. As soon as a router determines that a link has failed, they re-route packets through other links that are functional. A network operator still needs alarms from the routers that a link has failed. This is because the network operator needs to fix the problem permanently and restore connectivity quickly; otherwise, there could be a second or a third failure that could disrupt traffic.

At the link level, routers continuously monitor connectivity to their neighbors by using one of two mechanisms. First, routing protocols such as OSPF and IS-IS periodically (typically once every ten seconds) inject connectivity probes, called "HELLO" messages, to the other end of the link [65]. If any given router does not receive a few (typically four) consecutive HELLO messages, the router concludes that the link is down. Second, lower layers such as SONET can indicate that the physical level connectivity between two routers is down. For example, an optical interface on the router can detect that there is no physical signal and can generate a loss of signal (LoS) alarm, which can then be used by the network layer to determine that the link is down. In both these cases, the router generates an alarm that can be used by the operator to fix the problem.

In some cases, particularly in the VPN-over-MPLS setting, link-level connectivity alone is not sufficient to guarantee that end-to-end paths are in tact. In such cases, the network elements themselves directly do not detect disruption in connectivity, and hence do not automatically re-route. Therefore, network operators detect such failures by injecting active probes periodically between measurement servers that are connected to routers at the edge of the network. The measurement servers, as soon as they detect end-to-end performance problems, can raise an alarm so that the network operator can then fix the problem quickly.

While the above monitoring mechanisms detect that a particular failure has happened, that too within a short duration of the failure (typically a few seconds), they fail to indicate the root cause of the failure. For example, the loss of IGP "HELLO" messages enables a router to detect a link failure and subsequently generate an alarm to a network operator. However, the alarm by itself does not indicate to the operator whether the failure is because of an optical amplifier failure or a fiber cut. Therefore, an operator performs fault localization to determine the location of the failure.

II.D.2 Fault localization

Unless the root cause for the failure is known, nececssary repair actions cannot be taken. In many cases, the total duration of failure—from detection to repair—can be dominated by the amount of time it takes to identify the root cause. Fast fault localization, therefore, can significantly cut down the duration of failure.

It is not surprising, therefore, that a number of research prototypes [17, 23, 36, 58, 68, 70] and commercial products have been developed to diagnose problems in IP and telephone networks. Commercial network fault management systems such as NetFACT [38], OpenView [39], IMPACT [41], EXCpert [68], and SMARTS [81], provide powerful, generic frameworks for handling fault indicators, particularly diverse SNMP-based [16] measurements, and rule-based correlation capabilities. These systems present a unified reporting interfaces to operators and other production network management systems.

While such systems assist network operators with simple alarm correlation capability to reduce the number of alarms, fault localization still is predominantly performed in a manual fashion. For example, on a link failure, many optical elements along the link can generate an alarm indicating the optical circuit is down. Simple correlation systems club them together into one operational ticket used by the operators to manually localize the root cause. Localization in such cases, involves the operator verifying which among the several optical components has failed along the optical circuit. Therefore, fault localization can be both slow as well as labor-intensive. The main focus of this dissertation is to provide automated mechanisms for fast and accurate fault localization.

II.D.3 Repair

Once the fault is localized, depending on the nature of the fault, the condition can be repaired relatively easily. For example, once a network operator diagnoses a fault and finds that an optical amplifier has failed, he/she can replace the optical amplifier very fast. In more complicated failure conditions, such as a software bugs, it might still be easier to put a temporary fix before the exact nature of the problem is reproduced in the lab and a permanent fix is found for the particular problem.

II.E Summary

In this chapter, we have provided an overview of the architecture of today's service provider networks. In particular, we have focused on both public-domain Internet as well as enterprise virtual private networks, and pointed out that ISPs today offer a wide variety of services that need to be efficiently managed and administered. Therfore, the second part of the chapter focused on various network management functions of a typical Internet service provider. We argued that, while a typical network operator has to deal with several different network management functions, such as provisioning, QoS services, traffic engineering and attack mitigation, a significant amount of time and effort is spent towards effective fault management. We outlined three major components in fault management—fault detection, localization and finally repair—of which fault localization is both slow as well as labor-intensive thus dominating the overall down-time (from detection to failure) of a particular fault. In the next chapter, we develop a general methodology for fault localization that we apply towards specific failure scenarios.

Chapter III

Risk-modeling approach

Operational backbone networks are intrinsically exposed to a wide variety of faults and impairments. With the growing variety of services offered by ISPs today and the ever-increasing complexity of protocols, software and hardware, service providers are constantly engaged with a wide variety of complex failure modes that often prove to be difficult to detect, diagnose and recover from. The complete spectrum of faults in backbone networks is too large to explore and perhaps one solution cannot possibly address them all. In this dissertation, therefore, we focus on two different types of faults that commonly appear in practice—IP link failures and MPLS black holes.

In IP link failures, while current state-of-the-art monitoring systems within network elements (e.g., routers, optical amplifiers, DWDM systems) generate alarms on detecting a link failure, they do not automatically report the exact location of the failure. For example, a failure at an optical amplifier triggers alarms at IP, PPP and optical layers and across various components. These alarms, however, do not directly indicate that an optical amplifier failed, thus requiring a labor-intensive fault-localization step to identify the root cause of the failure. Experience has shown that the mean time to repair is often dominated by the localization step; in many cases traffic is immediately recovered once localized and the failure is resolved offline.

Second, in the context of VPNs provided by ISPs, as we have discussed before in Section II.B, customer VPNs are set up using MPLS tunnels between provider edge routers. These MPLS tunnels in turn use IGP protocols such as OSPF or IS-IS to determine the exact forwarding path within the AS. A "black hole" scenario can occur when the underlying IP infrastructure may be operational—each IP hop along the route is functioning properly—but the corresponding MPLS tunnel fails to deliver packets. Such a black hole can be silent in nature, with no alert/alarm indicating that the MPLS tunnel is actually broken. In such catastrophic cases, potentially many customers are out of service for extended periods of time—until the failure can be detected, localized and action be taken to recover (re-route) traffic.

At first glance, these two failure scenarios appear completely different, one involving IP links and the other involving MPLS tunnels. However, the basic abstraction is quite similar—*faults result in loss of connectivity or performance degradation across multiple entities.* Therefore, while fault-localization approaches are typically problem-dependent, we show that a methodology based on *risk modeling* is efficient in localizing the root cause in both of these failure scenarios. Of course, we still need to take into account particular domain-specific constraints and challenges before applying the risk-modeling approach for these two problem domains. In this chapter, we describe the abstraction and algorithmic approaches for fault localization leaving the problem-specific aspects to later chapters.

The rest of this chapter is organized as follows. First, we provide the basic terminology used throughout the dissertation in Section III.A. Next, we formally describe the risk-modeling abstraction in Section III.B. Finally, we outline the algorithmic approach we use in this dissertation in Section III.C, followed by a summary of previous approaches to the problem in Section III.D. Finally, we summarize this chapter in Section III.E.

III.A Terminology

We begin by defining the notation we shall use throughout the remainder of the dissertation. A *symptom* is defined to be an observable event that is indicative of a failure in the network. Typically, monitoring systems in the network are designed to detect various symptoms associated with a given failure. For example, if a link fails in the network, an alarm is generated by routers at both ends of the link with a timestamp indicating the exact detection time of the link failure. These alarms represent the symptoms observed in the network.

We use the term *root cause* to denote the main reason for a particular failure in the network. For example, a laser failure in an optical amplifier can be the root cause for a failure. Similarly, a bug in a routing protocol module within a router could also be a root cause. The granularity of a root cause, however, varies significantly depending on the situation. For example, along an end-to-end path spanning several ASes, it is meaningful to call one particular AS the root cause for problems along the particular path. On the other hand, a router or an optical amplifier could be the root cause for problems that occur on a given link.

A *risk group* is a set of symptoms dependent on a given root cause. If a failure occurs with a given root cause, then the set of symptoms that would be observed because of that failure constitutes a risk group. For example, let us suppose that if some component A fails, then symptoms B, C and D would be observed. Then, A is the root cause while the symptom set $\{B, C, D\}$ constitutes the risk group. A *risk model* is an association between several different root causes and sets of symptoms corresponding to each root cause.

We define a *failure signature* as a group of correlated symptoms. In practice, symptoms are monitored by different network elements, and, thus, are observed individually. Therefore, a failure signature is typically obtained by grouping together symptoms which are correlated either temporally or otherwise. The specific mechanism to correlate symptoms is problem dependent. One heuristic that we use in this dissertation is temporal clustering, where we form the failure signature by grouping together symptoms that are observed almost simultaneously.

Temporal correlation, however, is not a necessary condition. For example, if there is a particular software bug in the routers, then the root cause is the particular version of software running in the routers. However, this bug might not cause all the routers to exhibit an associated symptom at the same time, although they share the same root cause. It is also important to recognize the difference between a failure signature and a risk group. A failure signature is formed by correlating observed symptoms, and, thus, is reflective of coincidence. On the other hand, a risk group refers to a direct dependency between a root cause and a set of symptoms, and thus represents causality.

Finally, we define a *hypothesis* to represent a candidate set of root causes that could explain the failure signature. By "explain," we mean that each symptom in a failure signature should have direct causality with at least one of the root causes in the hypothesis. The term *ground truth* is often used to denote the real root cause of a failure. In contrast to the ground truth, a hypothesis consists of root causes inferred from symptoms, while the ground truth constitutes reality.

Ideally, each symptom corresponding to every root cause in the ground truth will be observed in the failure signature. However, this observation is not always guaranteed, as there could be symptoms missing from the failure signature because of inaccuracies in the detection or reporting system. In some other cases, spurious symptoms could be added to the the failure signature due to inherent noise in the network. Thus, a hypothesis is merely a set of root causes that explain the symptoms in the failure signature, regardless of whether there are extra and/or lost symptoms in the failure signature.

The goal of a *fault-localization algorithm* is to output a hypothesis that approximates the ground truth as closely as possible. Using this hypothesis, a network operator navigates through the potential root causes and determines which root cause is responsible for the problem in order to repair it permanently. In the next section, we describe the fault-localization problem formally.

III.B Theoretical problem formulation

We can define the problem formally as follows. Given a set of root causes, $C = \{c_1, c_2, \ldots, c_n\}$, and a set of symptoms $S = \{s_1, s_2, \ldots, s_m\}$. Each root cause, $c_i \in C$, is associated with a set of symptoms $S_i = \{s_{i1}, s_{i2}, \ldots, s_{iu}\} \subset S$ that would be observed if that particular root cause c_i fails. Given an input failure signature consisting of a set of symptoms, $F = \{s_{j1}, s_{j2}, \ldots, s_{jv}\} \subset S$, the problem is to identify the *best* possible hypothesis, $H = \{c_{k1}, c_{k2}, \ldots, c_{kw}\} \subset C$ such that H explains F, i.e., every symptom in F is associated with at least one root cause in H.

The hypothesis H is essentially a set of symptom sets, since each c_i can be re-written as the symptom set S_i, i.e., $H = \{S_{i1}, S_{i2}, ..., S_{ik}\}$. In the literature, given a set F, any set of sets H, such that the union of all the sets in H equals F, i.e., $\bigcup S_{il} = F, \forall S_{il} \in H$, is referred to as a *set cover* [46]. In general, for a given set F, there can exist many set covers H. In other words, multiple hypotheses can exist for the same signature. Therefore, the notion of what constitutes the best hypothesis needs to be precisely stated. If we have access to an oracle, we can enumerate all possible hypotheses and ask the oracle to pick the best hypothesis. In practice, however, we do not have access to such an oracle. Therefore, we use probabilities to rank different hypothesis, and pick the one with the highest probability. We split this discussion into two different cases depending on whether all root causes are equally likely.

III.B.1 Case I: All failures equally likely

If all root causes are equally likely, i.e., each c_i has equal chance of failure, say with probability p, and are independent, then the probability of a hypothesis of cardinality k is p^k. Thus, the smaller the size of the hypothesis, the higher the

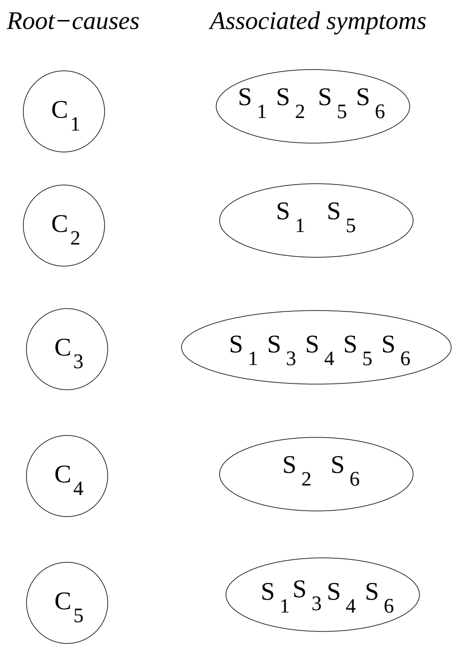


Figure III.1: An example risk model with associations between 5 root causes and 6 symptoms.

likelihood of the hypothesis; in many situations including the scenarios we consider in this dissertation, therefore, we prefer simpler hypotheses as they are more likely. Our preference is also in accordance with the general principle of Occam's razor [5], which is often stated as, "all things being equal, the simplest solution tends to be the best one." In our representation using set covers, therefore, the problem reduces to identifying the *minimum set cover* for a given failure signature set F.

We illustrate the problem with the help of an example. In Figure III.1, we show a risk model consisting of five root causes (labeled c_1 through c_5) and six symptoms (s_1 through s_6). In many networks, dependencies are ground-up; given a root cause we can identify the set of symptoms dependent on that root cause. Naturally, we begin with such symptom sets to represent the risk model—the relationship between root causes and symptoms—as shown in the Figure III.1. For example, the root cause c_1 has the set of symptoms $\{s_1, s_2, s_5, s_6\}$ dependent on it. Similarly, c_2's failure will affect the set of symptoms $\{s_1, s_5\}$. Notice that both c_1 and c_2 have the symptoms s_1 and s_2 in common; thus, different root causes

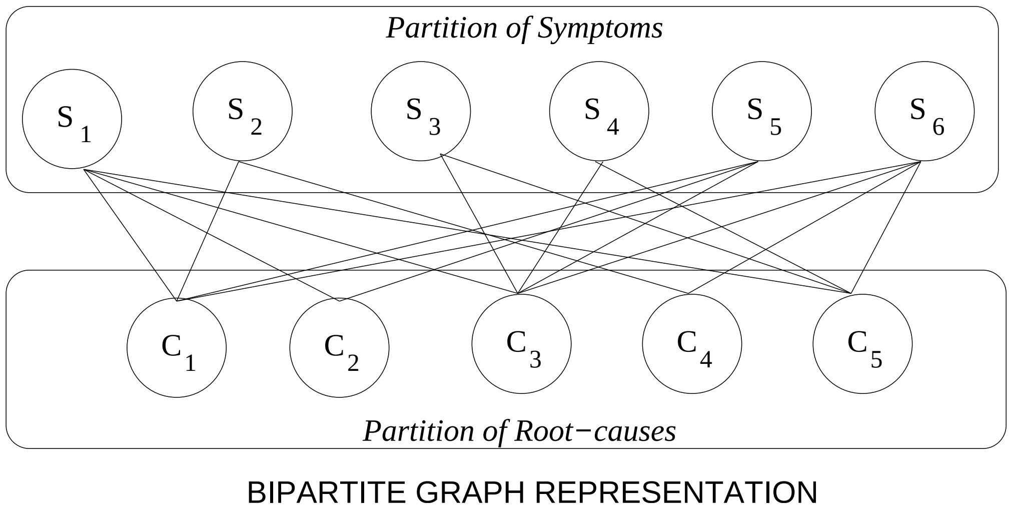


Figure III.2: Risk-model representation using a bi-partite graph.

can have common symptoms. In general, therefore, a one-to-many and a many-to-one (or jointly a many-to-many) relationship exists between root causes and symptoms.

While it is natural to represent dependencies using sets, it is easier to visualize the risk model using a bipartite graph as shown in Figure III.2. In the top partition, the set of symptoms s_1 through s_6 are shown, while the bottom partition contains the set of root causes c_1 through c_5. An edge exists from a root cause c_i to a symptom s_j, if the failure of c_i causes the symptom s_j to be observed. The goal of the localization algorithm therefore is to find the best hypothesis that explains all symptoms in the failure signature. Therefore, at least one root cause in the hypothesis should have an edge to each and every symptom in the failure signature.

Given a set of symptoms, say $\{s_1, s_2, s_5, s_6\}$, as the failure signature, the fault localization algorithm needs to determine a hypothesis, i.e., the minimum set of root causes that can completely explain all the symptoms observed in the failure signature. For the particular symptom set $\{s_1, s_2, s_5, s_6\}$, we would like the fault localization algorithm to output $H_1 = \{c_1\}$ as the root cause, since according to the risk model, a failure due to c_1 results in all the above symptoms to be observed.

For the same example, note that there is another competing hypothesis, $H_2 = \{c_2, c_4\}$ that can explain all the observed symptoms in the failure signature. The localization algorithm specifically does not know if the observed symptoms are due to one or multiple failures. Thus, the question is how to pick between H_1 and H_2 in this case. As mentioned before, the likelihood of H_1 is p and H_2 is p^2 if all root causes are equally likely to fail with probability p and are independent. Thus, we would want the algorithm to produce H_1, the simpler hypothesis, as the best hypothesis for the given failure signature. Next, we look at the case when this all failures are not equally likely.

III.B.2 Case II: All failures not equally likely

If all failures are not equally likely, it is not guaranteed that minimum cardinality hypothesis is the most likely hypothesis. Therefore, we need to identify the highest probability set cover for the failure signature, which in turn requires failure probabilities for individual root causes. In some scenarios, characterizing the probabilities of failures is feasible using either theoretical or empirical failure rates. For example, [95] outlines failure rate statistics based on empirical observations from ISPs. Such empirical failure rates often prove to be valuable for many network management tasks, including provisioning, capacity planning and traffic engineering.

It is unclear, however, whether these failure probabilities can be as valuable in the context of fault localization. We believe this is because real ISP networks are constantly in a state of flux with new technologies deployed and old equipment upgraded. Determining the failure models can be quite challenging, especially for newer technologies with little prior history (e.g., ultra long-haul optics). In such cases, one can perhaps approximate using existing models that correspond to obsolete technologies. However, it may not be desirable to use older and inaccurate fault models, because they can lead to inaccurate biases in the hypothesis output by the fault localization algorithm.

Therefore, in this dissertation, we use a structural approach to dependency relationships, by constructing the set of symptoms that directly depend on a root cause. If a certain symptom is observed, all the associated root causes are equally likely. The more symptoms we observe, the better we can disambiguate between different root causes. However, we hasten to add that, one can adapt the algorithms we proposed in this dissertation easily to scenarios where we not only have access to failure probabilities but also trust them.

For the rest of the dissertation, therefore, we assume that all failures are equally likely. Even with this simplifying assumption, we observed that our localization algorithms were quite effective in localizing the root causes in several empirically observed failure scenarios. In addition, we note that this assumption in fact reduces the run-time complexity of the localization algorithm significantly, which is an important requirement for any practical system. Thus our algorithms exhibit better scaling properties, as we apply our algorithms to large backbone networks with several thousand to millions of symptoms and root causes. In the next section, we describe our core algorithmic approach in more detail.

III.C Algorithmic framework for fault localization

As noted before, assuming all failures can occur with equal probability, the goal of the fault localization algorithm is to identify the minimum set cover for a given set of symptoms in a failure signature. Finding minimum set cover is a classic problem that has been shown to be NP-complete in [46], thus is computationally expensive. Fortunately, however, an approximation algorithm using a greedy approach exists for the minimum set cover problem [74]. Further, the greedy approximation is guaranteed to find a set cover with cardinality no more than $O(log\ n)$ times the size of the minimum set cover. More important than this bound on the cardinality, the greedy approach is effective in identifying those set of root causes that are the most significant, i.e., those that explain the most set of

Algorithm 1 GREEDY(FailureSignature F)

```
E = {}; // Explained set
U = F;
H = {}; // Hypothesis set
R = {}; // RootCauseVector
while (U ≠ {}) do
    for (symptom s ∈ U) do
        //All root causes associated with s
        R = R ⋃ getAllRootCauses (s);
    end for
    //Calculate metrics for comparing root causes
    calculateMetrics(R, F);
    bestRootCausesSet = identifyCandidates(R);
    //Move observations covered by root causes in bestRootCausesSet
    //from U to E
    moveSymptoms(bestRootCausesSet, E, U);
    addToHypothesis(H, bestRootCausesSet);
end while
return H;
```

symptoms in a given failure signature. It is this feature of the greedy approach in addition to low run-time complexity that makes it appropriate for the problem at hand.

The greedy approximation algorithm for fault localization, GREEDY, is shown in Algorithm 1. GREEDY initializes two sets E and U to the null set ϕ and the failure signature F respectively. The sets E and U correspond to the explained and unexplained set of symptoms in F. Then, GREEDY first obtains a root-cause vector R, that contains the union of all root causes associated with at least one symptom $s \in F$ (i.e., have an edge between s and that particular root cause). It then computes a greedy metric for each root cause in R as a function of the failure signature F, in the `calculateMetrics`() routine. In the next step, GREEDY returns the best candidate root causes ranked based on the metrics associated with each root cause in R in the `identifyCandidates`() function. It then prunes the set of observations explained by these candidate root causes (in `moveSymptoms`()) from the unexplained set U to E and repeats the process until no more symptoms remain in the unexplained set U.

The greedy approximation to the classic set-cover problem picks the sets covering most number of elements in every iteration [74]. However, depending on the particular problem, the algorithm can be adapted to use additional degrees of freedom in selecting the right candidate in every iteration. For example, we can choose to consider only those root causes for which all the symptoms associated with the root cause are found in the failure signature, irrespective of whether a large majority of symptoms in the failure signature are being explained by that particular given root cause. To preserve such flexibility, we abstract the `identifyCandidates`() and `calculateMetrics`() routines as black boxes. We describe these routines in detail when we apply the algorithm to specific localization problems later in the dissertation.

Depending on the particular problem, there could be other imperfections that need to be taken into account while designing the right fault localization algorithm. These imperfections include spurious and lost symptoms in the failure signature and inaccuracies in the risk models that we describe briefly next.

III.C.1 Imperfections in failure signature

In most real networks, there is inherent noise due to various types of routing, congestion, maintenance and other related events. Due to this noise, there could be *spurious symptoms* in the failure signature that are not directly related to the particular real failures that we are localizing, thus complicating localization. Besides noise, inherent inefficiencies in the detection system or loss of certain symptoms during reporting due to unreliable transport mechanisms could lead to an *incomplete failure signature*. The extent of both these imperfections varies based on the particular domain, and, therefore, has to be dealt with separately for each domain.

III.C.2 Imperfections in the risk model

As is the case with the failure signature, there are likely to be imperfections in the risk model as well. These imperfections stem from inherent churn in the network and once again depend on the particular failure domain under consideration. Depending on whether the risk model is dynamic, the risk model may need to be refreshed at an appropriate frequency. In addition, risk models can potentially drift away from reality leading to errors in the database. These errors can affect the localization results and hence must be factored into the localization algorithm. Because these operational realities are problem-dependent, similar to the imperfections in failure signature, we defer to the subsequent chapters when we apply the algorithms to the particular scenarios to explain how we adapt to these operational constraints.

In the next section, we review other algorithmic choices, besides the greedy approximation to minimum set-cover, for fault localization that have been previously proposed in the literature.

III.D Previous approaches

While our focus in this dissertation is in the context of backbone networks, many other similar problems have been observed in practice (for a more thorough survey, please see [87]). In a majority of these problem scenarios, the risk model is typically represented as a bipartite graph, that encodes the dependencies between symptoms and the actual faults, similar to our risk-model representation [17, 31, 48, 49]. However, the particular solution techniques are usually specific to the particular fault localization problem they solve. This is primarily because operational constraints often dictate the feasibility of a given technique.

Regardless of domain, fault detection systems have taken three basic approaches: rule or model-based reasoning [39, 12, 32], codebook approaches [81, 99], or machine learning (such as Bayesian or Belief Networks [96, 85, 19]). Rule-based and codebook systems (otherwise known as "expert systems") are often even more specific, only being able to diagnose events that are explicitly programmed. Model-based approaches are more general, but require detailed information about the system under test. Dependency-based systems like ours, on the other hand, allow general inference without requiring undue specificity. Indeed, the specific use of dependency graphs for problem diagnosis has been explored before [34, 86] but not for the particular problems we consider in this dissertation. Finally, the difficulty with probabilistic or machine learning approaches is that they are not prescriptive: it is not clear what sets of scenarios they can handle besides the specific training data. In addition, such approaches tend to be too slow when the number of nodes in the network is large such as ours.

Perhaps, the closest to our greedy approach on bipartite risk model is a Bayesian network [45, 86]. Using a prior distribution of probabilities (or assuming all probabilities identical) to each root cause, Bayesian analysis approaches attempt to identify the most probable hypothesis for a given set of observed symptoms. Given an assignment of 0 or 1 to a random variable representing the state of all the symptoms, $(s_1, s_2, ..., s_n)$, the Bayesian inference algorithm finds the most likely joint assignment of values to random variables representing the root causes, $(c_1, c_2, ..., c_m)$.

$$arg\ max_{c_1,...,c_m}\ P(c_1, c_2, ..., c_m | s_1, s_2, ..., s_n)$$

where $c_i \in \{0, 1\}$ and $s_i \in \{0, 1\}$.

The basic Bayesian approach as explored in [86] does not automatically deal with inaccuracies stemming from operational realities. It is also computationally expensive and hence does not scale well. Shrink [45] addresses these limitations by using two techniques. First, it augments additional edges in the bipartite graph to represent errors in the model. Second, it hypothesizes that the number of simultaneous failures is always less than a pre-determined k, typically $k < 4$. By exhaustively trying out $\binom{n}{k}$ failure assignments and computing the associated probabilities, the algorithm performs in $O(n^{k+1})$ time complexity which is typically much faster than the general approach in [86].

While Shrink addresses some of the basic problems with general Bayesian inference approaches, still, the algorithms are generally not easily applicable in our problem domains due to two reasons: First, although Shrink brings down the complexity to polynomial time, $O(n^4)$ is typically large for the problem domains under consideration in this dissertation. Second, Bayesian approaches fundamentally require access to the complete risk model, which as we shall show in Chapter V is not always true in practice. While one of the advantages claimed by the Bayesian approaches is that they can easily incorporate prior distribution of failure probabilities (for individual root causes), our basic greedy can easily be extended to incorporate weights with each root cause as well.

III.E Summary

In this chapter, we described two fault-localization problems considered in this dissertation. Both these problems share the same basic abstraction, in which each fault affects multiple entities simultaneously that are monitored in the network. Based on this abstraction, we developed a risk-modeling methodology that creates a risk model (i.e., a dependency relationship) between a set of root causes with a set of symptoms monitored in the network. Given a set of observed symptoms and the associated risk model, the goal of the fault-localization algorithm is to infer the root causes that can explain the set of observed symptoms. We showed that this problem can be stated as finding a set cover for the set of observed symptoms. If all failures are equally likely, then the most probable hypothesis is in fact a minimum set cover. Because finding minimum set cover is NP-complete, we approximate finding minimum set cover using a greedy approach which we outlined in Algorithm 1. Finally, we reviewed some operational imperfections that a localization algorithm needs to taken into account. These imperfections, however, are problem-domain dependent; hence, we explain how the algorithms can be adapted in the next two chapters, when we apply the risk-modeling approach to the particular problems.

Chapter IV

IP Link Fault Localization

In this chapter, we apply the risk-modeling methodology developed in the previous chapter to the specific problem of fault localization across IP and optical network layers, a daunting problem faced by network operators today. Currently, when network operators receives router-interface alarms indicating link failures, they are often faced with time-intensive manual investigation determining which layer the problem occurred, where, and why. This task is hampered by the architecture of the underlying network: IP uses optics for transport and (in some cases) for self-healing services (e.g., SONET ring restoration) in an overlay fashion. The task of managing each of the two network layers is naturally separated into independent software systems.

Joining dynamic fault data across IP and optical systems is challenging—the network elements, supporting standards and information models are totally different. Though network operators routinely use fields, such as circuit IDs, to join databases across IP and optical databases, they indicate the lack of automated mechanisms to ensure the accuracy of these joins. Unfortunately, the network elements and protocols provide little help as well. Path-trace capabilities (counterparts of IP traceroute) are often not available at the optical layer, and, even when available, do not work in a multi-vendor environment (e.g., where the DWDM systems are provided by multiple vendors). In optical systems such as SONET, there is no counterpart to IP utilization statistics, which might be used to correlate traffic at the IP layer with the optical layer. Both IP and optical network topologies are rapidly changing as equipment is upgraded, network reach is extended, and the topologies and capacities are re-engineered to manage changing demands.

One of the key contributions of this dissertation is the application of our risk-modeling methodology to localize faults across the IP and optical layers in operational networks. Roughly speaking, a physical object such as a fiber span or an optical amplifier represents a *shared risk* for a group of logical entities (routers or IP links) at the IP layer. That is, if the optical device fails or degrades, all of the IP components that had relied upon that object fail or degrade. In the literature, these associations are referred to as *shared risk link groups* or SRLGs [18]. Using only event data gathered at IP layer, and topology data gathered at both IP and optical layers, we bridge the gap between the operational information network managers need and what is actually reported at IP layer. Our system assists operators alleviate the burden of cross-correlating dynamic fault information from two disparate network layers. Once the layer and the location of the fault has been determined, other systems and tools at the appropriate layer can be targeted towards identifying the precise characteristics (for example, rule-based or statistical methods [39, 81]) of the particular failure sometimes required before repair can be performed.

The rest of this chapter is organized as follows: First, we discuss troubleshooting IP link failures using SRLGs in Section IV.A. Then we describe the application of risk-modeling methodology in the context of IP fault localization in Section IV.B, including dealing with imperfections stemming from operational constraints. Next, we discuss the system architecture in Section IV.C followed by evaluation using simulations as well as real failure data in Section IV.D and Section IV.E respectively.

IV.A Troubleshooting IP link failures using shared-risks

Faults that affect IP layer performance come in many flavors—some disappear of their own accord, and are thus never investigated or are investigated offline to identify chronic conditions. Others, such as fiber cuts, can cause a number of links in the network to fail simultaneously; impacted capacity remains unavailable until the failure is repaired. In addition to faults which affect capacity, networks are also subject to failure modes which impair performance—such as by introducing errors into packets transmitted through the network. For example, very low level spurious errors are a fact of life with optical transmission systems, but typically have minimal or no impact on customer service [72]. However, as components degrade over time, these error rates can increase and can start impacting IP layer services. Thus, it is critical for operators to be repair such performance degradations pro-actively before they turn catastrophic.

Network operators in large-scale IP networks need to detect and rapidly troubleshoot before they can repair a failed component or a performance degradation. Thus, the life-cycle of a typical fault involves the following steps.

- *Network monitoring.* Network elements and components are continually monitored, with notifications generated in the form of SNMP [16] traps or alarms when "problematic" conditions arise. A centralized management system collects and correlates these alarms to eliminate duplicates and dampen intermittent faults that do not require immediate further investigation (left for offline analysis).

- *Fault localization and diagnosis.* Once a fault has been reported, network operators must determine the root cause before it can be repaired. Typically, the job of isolating and diagnosing a problem requires highly skilled staff to collect relevant information regarding the failure, and gather additional data to assist in the diagnosis.

- *Verifying components.* As part of diagnosis, supporting performance, topology and other information can be critical to isolating and further verifying the problem. In some cases, by purely trying out all combinations and wild guesses, operations try and troubleshoot the problem.

- *Repair*. After the troubled component is isolated and the problem identified, an appropriate course of action must be determined and repair actuated. In simple fiber cuts, this will likely involve sending personnel out into the field to splice fibers. However, other types of problems may require software "bug" fixes or configuration changes.

Depending on the root cause of a fault, diagnosis and repair can be very time consuming. Numerous problems arise in managing large networks where it is extremely challenging to sort through the relevant data to identify the root cause of a problem—particularly when it involves inter-layer inter-working, software or configuration related faults or interactions. In most situations, networks are designed to have sufficient capacity to re-route traffic around failures; so, as long as repair is completed before another large failure, customers see little or no performance impact. However, despite careful engineering practices, the edge of the network is inherently a single point of failure—certain failures can interrupt customer service until they are repaired. Rapid fault isolation and repair are thus of significant importance here.

Network elements today, however, typically generate alarms on an individual basis, thus requiring a manual correlation to determine that they are all because of a common network element. For example, a router failure will appear as a failure of all of the links terminating at that router. Best current practice requires correlation of these link failures to determine that it was a router failure. In some failure scenarios, it can be substantially more challenging to group individual alarms into groups, and often difficult to even identify in which layer the fault occurred (e.g., in the transport network interconnecting routers, or in the routers themselves). By identifying the set of possible components that could have caused the observed symptoms, risk modeling using SRLGs can serve as the first step of diagnosing the root cause of a network problem.

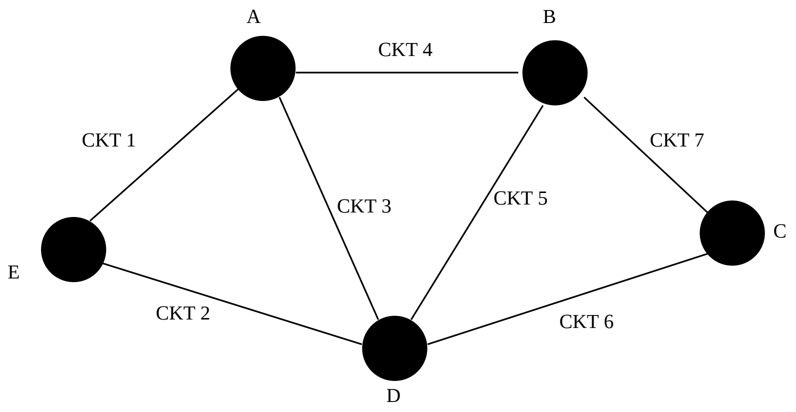
IV.A.1 Shared risks in IP networks

Our challenge is to construct a model of risks that represents the set of IP links that would likely be impacted by the failure of each component within the network. The complexity of hardware and software upon which an IP network is built implies that constructing a model that accounts for every possible failure mode (e.g., individual circuit boards within a DWDM system) is impractical. Instead, we identify the key components of the risk model that represent the prevalent network failure modes and those that do not require deep knowledge of each vendor's equipment used within the network. We hasten to add that the finer the granularity of the risk model, the more precise the fault diagnosis can be.

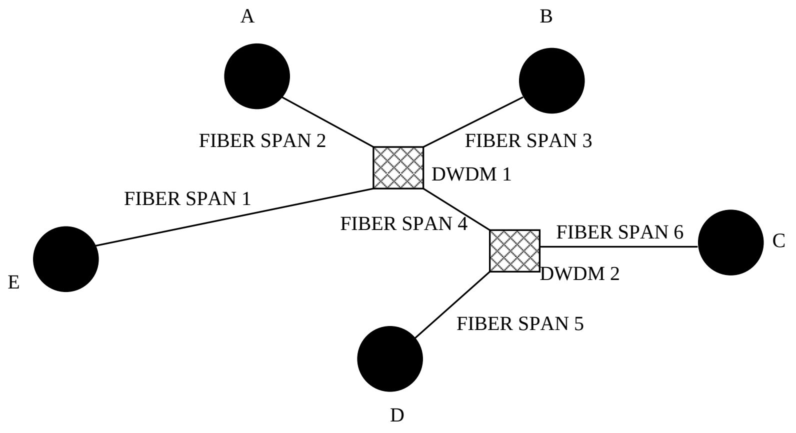
The basic network topology can be represented as a set of nodes interconnected via links. Intra- and inter-domain routing protocols such as OSPF and BGP operate with a basic abstraction of a point-to-point link between a routers. Figure IV.1(a) illustrates the logical view of a simple network consisting of five nodes connected via seven links or optical circuits. So if a fault occurs on a particular link (say CKT 1 in the figure), there would be a corresponding OSPF/BGP message that refers to this failure.

Each inter-office IP link is carried on an optical circuit (typically using SONET). This optical circuit in turns consists of a series of one or more fibers, optical amplifiers, SONET rings, intelligent optical mesh networks and/or DWDM systems [72]. These systems consist of network elements that provide optical to electrical to optical (O-E-O) conversion and, in the case of SONET rings or mesh optical networks, protection/restoration to recover from optical layer failures. Multiple optical fibers are then carried in a single conduit, commonly known as a fiber span.

Typically, each optical component may carry multiple IP links—the fail-



(a) Logical topology



(b) Physical topology

Figure IV.1: Example topology showing logical as well as physical topologies of an IP network. IP circuits in the logical topology share various optical components in the physical topology

ure of these components would result in the failure of all of these IP links. We illustrate this concept in the Figure IV.1(b), where we show the optical layer topology over which the IP links are routed. In the Figure IV.1(b), these shared risks are denoted as FIBERSPAN 1 to 6, DWDM 1 and 2. CKT 3 and CKT 5 are both routed over FIBER SPAN 4 and, thus, would both fail with the failure of FIBER SPAN 4. Similarly, DWDM 1 is shared between CKT 1, 3, 4 and 5, while CKT 6 and CKT 7 share DWDM 2.

In essence, each network element represents a shared risk among all the links that traverse through this element. Hence, this set of links represents an SRLG [43, 90], as we have discussed before. The concept of SRLGs is widely used in the context of traffic engineering, where primary and backup paths are chosen such that they do not have any SRLG in common. However, to the best of our knowledge, we are the first to apply the notion of SRLGs to fault localization. Note that our approach is mainly effective in scenarios where the root cause of the failure affects many different IP links. If only one link fails, then our approach is not as effective since the failure of another link cannot be used to disambiguate between many different root causes. We now discuss the different types of SRLGs that can be used to create the risk model for fault localization.

IV.A.2 Network SRLGs

We divide the risk model into hardware-related risks and software risks. Note that this model is not exhaustive, and can be expanded to incorporate, for example, additional software protocols.

Hardware-related SRLGs

- Fiber: At the lowest level, a single optical fiber carries multiple wavelengths using DWDM. One or more IP links are carried on a given wavelength. All wavelengths that propagate through a fiber form an SRLG with the fiber being the risk element. A single fiber cut can simultaneously induce faults on all of the IP links that ride over that fiber.

- Fiber Span: In practice, a set of fibers are carried together through a cable. A set of cables are laid out in a conduit. A cut (from, e.g., a backhoe) can simultaneously cause all links carried through the conduit to fail. These set of circuits that ride through the conduit, therefore, form a fiber span SRLG.

- SONET Network Elements: We collectively group the SONET network elements together to the SONET SRLG category: these consist of optical amplifiers, add-drop multiplexers (used to construct optical rings), and DWDM O-E-O converters and other similar components.

- Router Modules: A router is usually composed of a set of modules, each of which can terminate one or more IP links [1]. A module-related SRLG denotes all of the IP links terminating on the given module, as these would all fail should the module die.
- Router: A router typically terminates a significant number of IP links, all of which would likely be impacted by a router failure (caused by either router software or hardware). Hence, all of the IP links terminating on a given router collectively belong to a router SRLG.
- Ports: An individual link on a router can also fail due to the failure of a single port on the router (impacting only the one link), or through other failure modes that impact only the single link. Thus, we also include Link SRLGs in our model. Note that, even though each port is a singleton set consisting of only one link, still, it is important to consider them in the risk model because otherwise, we would miss ports from being root causes in the risk model.

Software-related SRLGs

- Autonomous System: An autonomous system (AS) is a logical grouping of routers within a single enterprise or provider network (typically managed by a common team and systems). These routers typically all run a common instance of intra-domain protocol software. Hence, although extremely rare, a single IGP software implementation bug can cause an entire AS to fail.
- OSPF Areas: Although an OSPF area is a logical grouping of a set of links for intra-domain routing purposes, there can be instances where a faulty routing protocol implementation can cause disruptions across the entire area. Hence, the IP links in a particular area form an OSPF Area SRLG.

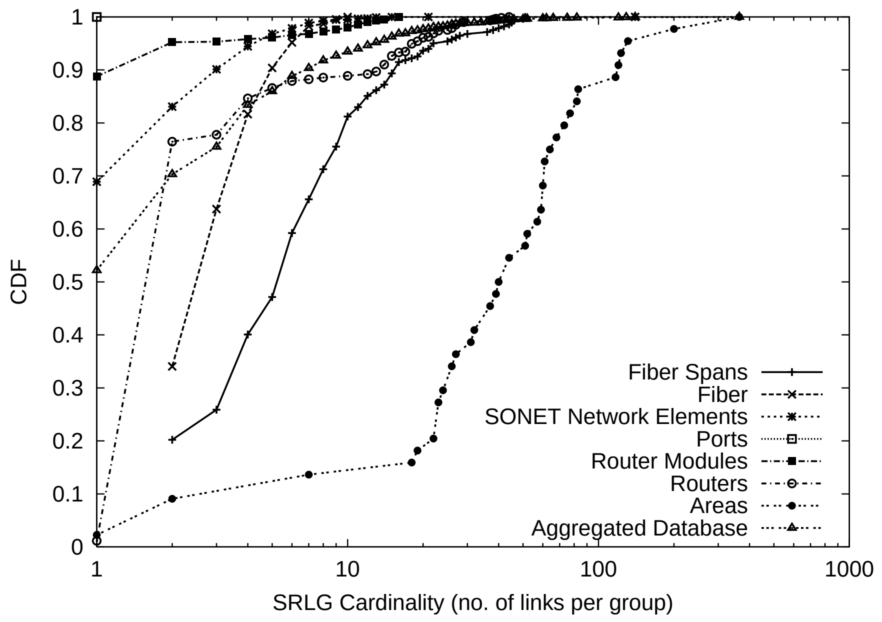


Figure IV.2: CDF of shared risks among real SRLGs.

Not all SRLGs have corresponding failure diagnosis tools associated with them. For example, a fiber span is a physical piece of conduit that generally cannot indicate to the network operator that it has been cut. Similarly, there is no monitoring at the OSPF area level that can indicate if the whole area was affected. Diagnosis is therefore based on inference from correlated failures that can be attributed to a particular SRLG. In the absence of monitoring information directly from the equipment, this is the only known approach to localize the root cause of the failure in the network.

IV.A.3 Shared risk in real networks

Fault localization using spatial correlation is inherently enabled by richness in the overlaps between between SRLGs. In particular, spatial correlation will typically be most effective in networks where SRLGs consist of multiple IP links, and each IP link consists of multiple SRLGs. Figure IV.2 depicts the cumulative distribution function (CDF) of the SRLG cardinality (the number of IP links in each SRLG) in a segment of a large tier-one IP backbone network (in particular, customer-facing interfaces are not included here). The figure shows the cumulative distribution function (CDF) of both individual SRLGs as well as for the aggregated database. We can observe from this figure that, as expected, OSPF areas typically consist of a large number of links (and, hence, are included in their SRLG), whereas port SRLGs (by definition) comprise only a single circuit. In between, we can see that fiber spans typically have a significant number of IP links sharing them, while SONET network elements typically have fewer. The important observation here is that there is a significant degree of sharing of network components that can be utilized in spatial correlation. Studies of the number of SRLGs along each IP link show similar results. Thus, we conclude that risk modeling using SRLGs holds great promise for large-scale IP networks. We describe the application of risk modeling to IP fault localization in the next section.

IV.B Fault localization using risk modeling

Given the risk model constructed using SRLGs and link failure notifications from routers, the essential problem of IP fault localization is to determine the set of SRLGs that can explain a given set of link failure notifications. It is easy to observe that this problem can be directly reduced to the general fault-localization problem formulation in Section III.B, with link failure notifications l_i corresponding to the symptoms s_i, and SRLGs g_j to the root causes c_j. Therefore, given an input failure signature comprising of link failures, $F = \{l_{i1}, l_{i2}, \ldots, l_{in}\}$, therefore, the goal is to identify the hypothesis, $H = \{g_{h1}, g_{h2}, \ldots, g_{hk}\}$ such that H is a minimum set cover for F, assuming all root causes g_j can occur with same probability.

Similar to the representation in Section III.B, we can also model the problem using a bipartite graph as shown in Figure IV.3. Each link, l_i, and group, g_j, is represented by a node in the graph. The bottom partition consists of nodes corresponding to the risk groups; the top nodes correspond to links. An edge

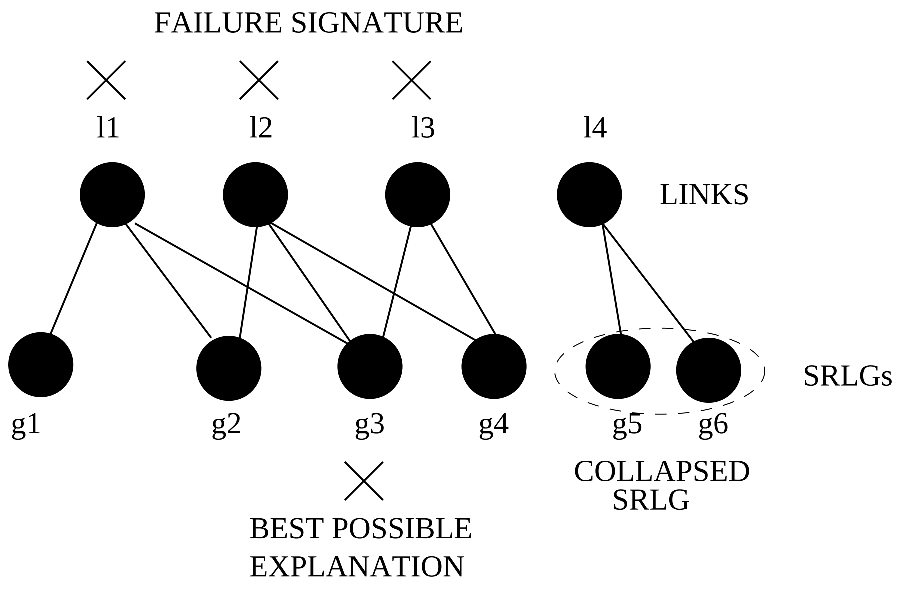


Figure IV.3: Representing shared-risk groups as a bipartite graph. At the bottom partition are the risk groups corresponding to the root causes. The top partition consists of the individual IP links.

exists between a link node and a group node if that link is a member of the risk group. Given this bipartite graph and a subset of vertexes in the top partition (corresponding to the failure signature), the problem is to identify the smallest possible set of groups that cover the link failure events in the failure signature.

Before we proceed further, we observe that if multiple risk groups have the same membership—that is, the same set of circuits may fail for two or more different reasons—it is impossible to distinguish between the root causes. We call any such risk groups *aliases*, and collapse all identical groups into one in our set of risk groups. For example, in Figure IV.3, group $g5$ and $g6$ have the same membership: $l4$. Hence, $g5$ and $g6$ are collapsed into a single group as a pre-processing step.

Given that determining the minimum set cover is computationally prohibitive, we compute an approximation instead using the greedy approach outlined in Section III.C. Our algorithm called SCORE is predominantly based on Algorithm 1, except that we specify the particular functions `calculateMetrics()` and

Algorithm 2 `calculateMetrics(rootCauses R, failureSignature F)`

```
1: for (rootCause r ∈ R) do
2:     r.hitRatio = |R ∩ F|/|R|;
3:     r.covRatio = |R ∩ F|/|F|;
4: end for
```

`selectCandidates()` left undefined in Algorithm 1. There is one more difference between SCORE and GREEDY—an additional threshold $T \in (0, 1]$ that is input to SCORE along with the failure signature, for reasons that would become clearer soon.

The function `calculateMetrics()` shown in Algorithm 2, computes two metrics for each SRLG—hit- and coverage-ratio. The *hit ratio* of a group G_i is defined as $|G_i \cap F|/|G_i|$, with $|G_i|$ denoting the cardinality of G_i. In other words, the hit ratio of a group is the fraction of circuits in the group that are part of the failure signature F. The *coverage ratio* of a group G_i is defined as $|G_I \cap F|/|F|$. Basically, the coverage ratio is the proportion of failure signature explained by a given risk group.

Intuitively, SCORE attempts to iteratively select the risk group that explains the greatest number of faults in the failure signature with the least error: in other words, the highest coverage and hit ratios. Ideally, if an SRLG fails, all the associated links would fail; we should, therefore, only consider those risk groups whose members are all part of the failure signature F (i.e., $G_i \cap F = G_i$). For such risk groups, clearly, hit ratio equals one. Thus, SCORE should ideally consider only risk groups with a perfect hit ratio. However, in order to take into account certain operational realities such as losses in failure notifications, we relax this requirement slightly using a threshold T input to SCORE. As shown in Algorithm 3, SCORE only considers those risk groups which have a hit ratio greater than a given threshold T. Among such risk groups, it picks the one which has the highest coverage ratio.

Considering risk groups that have a hit ratio less than one can lead to a

Algorithm 3 `selectCandidates (rootCauses R, threshold T)`

```
1: max = 0;
2: candidateSet = {}
3: for all rootCause r ∈ R do
4:    if (r.hitRatio ≥ T) then
5:       if (r.covRatio > max) then
6:          max = r.covRatio;
7:          candidateSet = {r};
8:       else if (r.covRatio == max) then
9:          candidateSet = candidateSet ⋃ {r};
10:      end if
11:   end if
12: end for
13: return candidateSet
```

hypothesis that potentially explains more circuit failures than actually occurred. In a straightforward failure model, such hypotheses are nonsensical. Operational realities described below, however, require us to consider risk groups with hit ratios less than one.

- *Incomplete/erroneous monitoring data.* The failure notices (e.g., SNMP traps) are often transmitted using unreliable protocols such as UDP which can result in partial failure observations. Hence, the accuracy of the diagnosis can be impacted if the data is erroneous or incomplete. For example, if due to the failure of a particular optical component failure, six links went down out of which only five, say, messages made it to the monitoring system. The hit ratio for the risk group representing the shared component is then 5/6. Without expressly allowing for the selection of this risk group, the algorithm would output a hypothesis, that, while plausible, is likely far from reality.

- *Inaccurate modeling of the shared-risk groups.* While theoretically it should be possible to precisely model all risk groups, it is impossible in practice to exactly capture all possible failure modes. This difficulty leads to two interesting cases of inaccurate modeling. One is failure to model high-level risk groups (e.g., all links terminating in a particular point of presence may share a power grid) while the other is failure to model low-level risk groups (for example, some internal risk group within a router). Our algorithm needs to be robust against imprecise failure groups and, if possible, learn from real observations.

- *Errors in databases.* Currently, the SRLG databases are constructed and maintained by humans, given the lack of architectural mechanisms to derive dependencies between IP links and optical components automatically. These errors can cause the database to drift away from reality, thus risk models constructed out of the SRLG databases are only an approximation of reality. For example, a human accidentally swaps fibers that belong to two different interfaces. Or, perhaps, a technician re-routes an optical circuit through another set of components, but forgets to log an entry into the database. In such case, the algorithm assumes that it would detect the complete risk-group failure, while due to the errors, some elements in the risk-group might be missing. Since IP to optical associations are important across other network management tasks such as traffic engineering, we consider architectural approaches to maintain these associations accurately in Chapter VI.

We allow for these operational realities by selecting the risk group with greatest coverage out of those with hit ratios above a certain error threshold. So, even if a particular circuit is omitted (either due to incorrect modeling or missing data), the error threshold allows consideration of groups that have most links but not quite all and cover a large number of failures.

It turns out to be extremely difficult to select a single error threshold for all failure instances, as it depends greatly on the size of individual risk groups

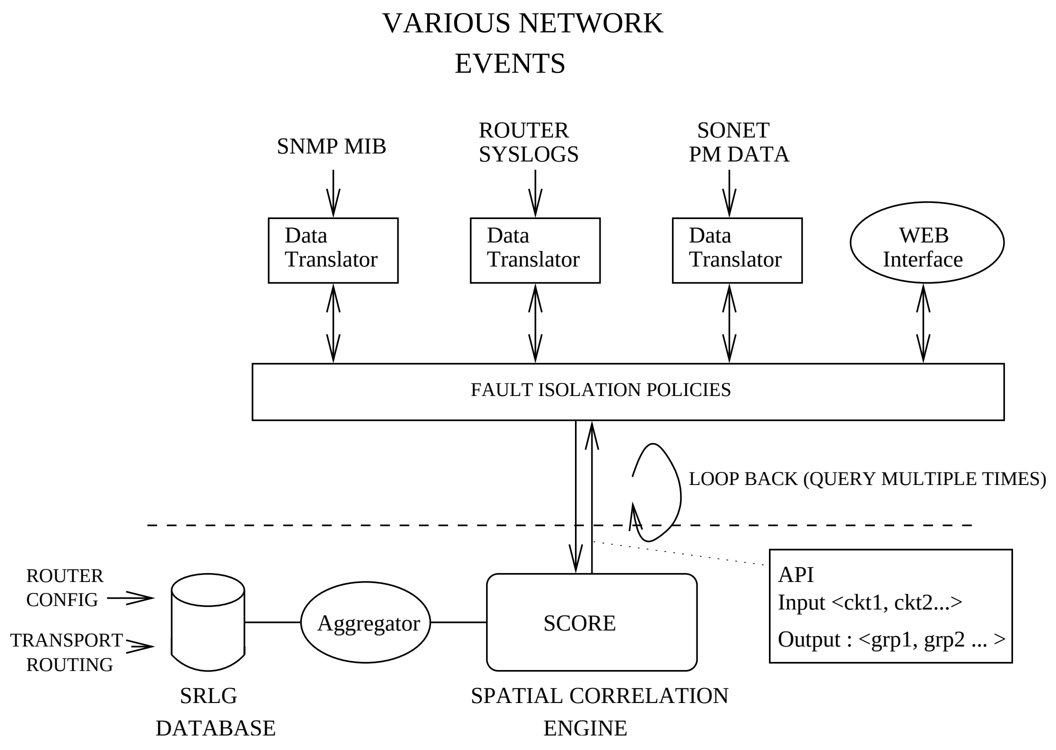


Figure IV.4: Architecture of the IP fault localization system.

involved in the failure. In practice, we run SCORE multiple times and generate hypothesis for decreasing error thresholds until a plausible hypothesis is generated. More generally, we assign a cost function to evaluate the confidence of a particular hypothesis and choose the one which has the lowest cost. Our cost function is directly proportional to the size of the hypothesis and inversely proportional to the error threshold. Thus, the cost function penalizes large hypothesis and large relaxation. Next, we present an overview of the system based on the SCORE algorithm.

IV.C System overview

We created the IP fault localization (IPFL) system with generality in mind. Accordingly, key systems and algorithmic components are factored out so that they may be reused in multiple problem domains or in variations for a single problem domain. A standalone SCORE algorithmic module is driven by an extensible set of problem-domain dependent diagnosis processes. Intelligence from the problem domain is built into the SRLG database, and is reflected in the IPFL queries. Figure IV.4 depicts the IPFL system architecture as it is implemented today. The following subsections describe the various modules in more detail.

IV.C.1 SRLG Database

The SRLG database manages SRLG groups and corresponding links. For example, in our application, the database atoms used to form SRLGs at the SONET layer describe SONET-level equipment IDs that particular IP links traverses, extracted from databases populated from operational optical-element management systems. Other risk groups such as area, router, modules, etc. are similarly formed from the native databases extracted from the various network elements (e.g. router configurations). We note that the underlying databases track the network and therefore exhibit churn. The IPFL software is currently snapshot driven, and copes with churn by reloading multiple times during the course of a day.

IV.C.2 SCORE localization algorithm

The SCORE algorithm described in Section IV.B forms the core of the system. The IPFL system periodically invokes the SCORE algorithm with the SRLG database, which then responds to queries for fault localization. That is, SCORE obtains the minimum set hypothesis using the SRLG database and a given set of inputs. An optional error threshold can be specified, as described in Section IV.B.

IV.C.3 Data Sources

The set of observations upon which spatial correlation is applied are obtained from network fault notifications and performance reports (including IP performance-related alarms). These in turn come from a wide range of data sources. We discuss below some of the more popular fault and performance-related data sources that have been used within the IPFL system to date. Though we describe certain optical-layer event data sources (such as SONET performance management data) and have indeed experimented with such sources with SCORE, we only focus on IP event sources in this dissertation due to lack of enough data from other sources.

IP-layer SNMP traps: Link failures and other faults will be observed by the routers and reported to centralized network operations systems via SNMP traps sent from the router. These SNMP traps provide the key event notifications that allow network operators to learn of faults as they occur.

Router syslogs: Router operating systems, much like Unix operating systems, log important events as they are observed. These are known as *router syslogs* and provide a wealth of useful information regarding network events (at least once they can be interpreted!). These can be used as additional information to complement the SNMP traps and the alarms that they generate. Table IV.1 shows sample syslog messages for a failure observed on a Cisco router, and another failure observed on an Avici router. The failures are reported at different layers—illustrated here for the SONET layer, PPP layer and IP layer (OSPF). Note that there is no standardized format for these messages as they are usually output for debugging purposes.

SNMP performance measurements: SNMP performance data is generated by the routers on either a per-interface or per-router basis, as applicable. It typically contains five-minute aggregate measurements of statistics such as traffic volumes, router CPU average utilization, memory utilization of the router, number of packet errors, packet discards and so on.

SONET performance monitoring (PM): Performance metrics are also available on a per-circuit basis from SONET network elements along an optical path (as are alarms, although these are not discussed here). Numerous param-

Table IV.1: Syslog messages output by Cisco and Avici Routers when a link goes down at different layers of the stack. When the link comes back up, the router writes similar messages indicating that each of the layer is back up.

Syslog Message on Cisco/Avici Routers	Layer	Router
Aug 16 04:01:29.302 EDT: %LINEPROTO-5-UPDOWN: Line protocol on Interface POS0/0, changed state to down	SONET layer	Cisco
Aug 16 04:01:29.305 EDT: %LINK-3-UPDOWN: Interface POS0/0, changed state to down	PPP layer	Cisco
Aug 16 04:01:29.308 EDT: %OSPF-5-ADJCHG: Process 11, Nbr 1.1.1.1 On POS0/0 from FULL to DOWN, Neighbor Down: Interface down or detached	OSPF/IP layer	Cisco
module0036:SUN SEP 12 17:23:29 2004 [030042FF] MINOR:snmp-traps :Sonet link POS 1/0/0 has new adminStatus up and operStatus up.	SONET Layer	Avici
server0001:SUN SEP 12 17:25:01 2004 [030042FF] MINOR:snmp-traps :PPP link POS 1/0/0 has new adminStatus up and operStatus up.	PPP layer	Avici
server0002:THU AUG 12 07:21:58 2004 [030042FF] MINOR:snmp-traps:OSPF with routerId 1.1.1.1 had non-virtual neighbor state change with neighbor 1.1.1.2 (address less 0) (router id 1.1.1.4) to state Down.	OSPF/IP layer	Avici

eters will be reported in, for example, fifteen-minute aggregates. These include parameters such as coding violations, errored sections and severely errored seconds (indicative of bit error rates and outages), and protection switching counts on SONET rings. More information on various types of events reported by SONET PM data can be found in [3].

IV.C.4 Data translation/normalization

Each of these types of monitoring data are usually collected from different network elements (such as routers, SONET DWDM equipment, etc.) and streamed to a centralized database. These different data types are usually stored in different formats with different candidate keys for database access. For example, the candidate key for SNMP database is an interface number as it collects interface-level statistics. OSPF messages are based on link IP addresses. SONET performance monitoring data is based on circuit ID. All these data sources are mapped into link circuit IDs using a set of mapping databases.[1]

IV.C.5 Fault localization policies

Fault localization is performed on various monitoring data sources (such as those mentioned in the previous section) using flexible data-dependent policies. In Figure IV.4, fault isolation policies form the bridge between the various monitoring data sources (translators) and the main SCORE localization engine. These policies dictate how to form the failure signature from the raw incoming event stream and query the SCORE algorithmic engine to identify the best hypothesis.

Data sources that are based on discrete asynchronous events (e.g. OSPF messages, syslog messages) need to be clustered to form the failure signature. Temporal clustering captures all the events that took place in a fixed time interval as potentially correlated. Note that a fault can trigger events that are slightly off in time either due to time synchronization issues across various elements, or

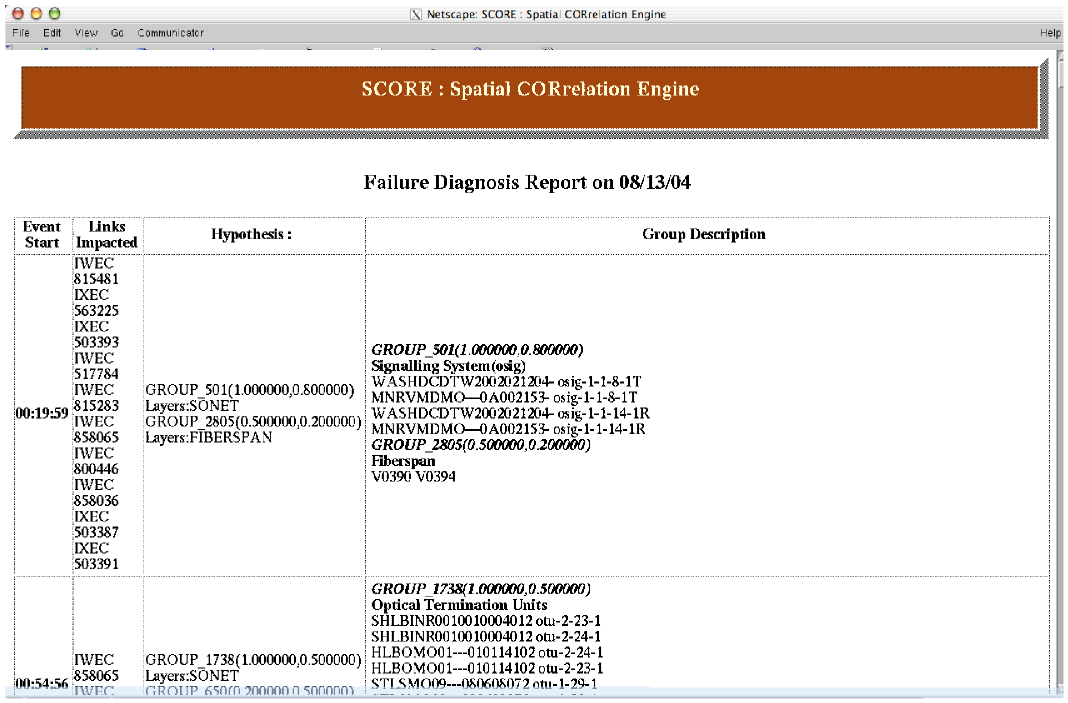
[1]We map all the databases into link circuit IDs since the network database itself is organized based on link circuit IDs. However, any unified format would work equally well.

propagation delays in an event to be recorded, or even delay in impact due to convergence in the network. Hence, event clustering has to account for these in recording observations.

There are many different ways to cluster events. A naive approach to clustering is based on fixed time bins. For example, we can make observations (set of links potentially correlated) by clustering together all events in a fixed five-minute bin. The problem with this approach, however, is the fact that events related to a particular failure can potentially straddle the time-bin boundary, thus creating two different observations for correlated events, which in turn may affect the accuracy of diagnosis.

In our system, we use a clustering algorithm based on gaps between failure events. We use the longest chain of events that are spaced apart within a set threshold (called a quiet period) as potentially correlated events. The intuition here is that two events that are separated by less than the quiet period are potentially correlated. The algorithm clusters events that are correlated in a transitive fashion. For example, if A and B are correlated, B and C are correlated, A and C are potentially correlated and belong in the same cluster. These clustered events are then fed to the SCORE engine to obtain a hypothesis that represents the failed components in the network.

Although currently we use temporal correlation as a good indication that events potentially can have the same root cause, it is possible to apply other methods to cluster events. For example, consider a bug that affects only routers with a particular version of software. In such a scenario, while all these routers share the same root cause, i.e., the version of software, still, the symptoms associated with this bug may not manifest at the same time. Therefore, an offline mechanism is required to identify these failure signatures, that then can be used to localize the root cause.



Netscape: SCORE : Spatial CORrelation Engine
File Edit View Go Communicator
Help

SCORE : Spatial CORrelation Engine

Failure Diagnosis Report on 08/13/04

Event Start	Links Impacted	Hypothesis :	Group Description
00:19:59	IWEC 815481 IXEC 563225 IXEC 503393 IWEC 517784 IWEC 815283 IWEC 858065 IWEC 800446 IWEC 858036 IXEC 503387 IXEC 503391	GROUP_501(1.000000,0.800000) Layers:SONET GROUP_2805(0.500000,0.200000) Layers:FIBERSPAN	***GROUP_501(1.000000,0.800000)*** **Signalling System(osig)** WASHDCDTW2002021204- osig-1-1-8-1T MNRVMDMO---0A002153- osig-1-1-8-1T WASHDCDTW2002021204- osig-1-1-14-1R MNRVMDMO---0A002153- osig-1-1-14-1R ***GROUP_2805(0.500000,0.200000)*** **Fiberspan** V0390 V0394
00:54:56	IWEC 858065 IWEC	GROUP_1738(1.000000,0.500000) Layers:SONET GROUP_650(0.200000,0.500000)	***GROUP_1738(1.000000,0.500000)*** **Optical Termination Units** SHLBINR0010010004012 otu-2-23-1 SHLBINR0010010004012 otu-2-24-1 HLBOMO01---010114102 otu-2-24-1 HLBOMO01---010114102 otu-2-23-1 STLSMO09---080608072 otu-1-29-1

Figure IV.5: Live screen shot of SCORE web interface.

IV.C.6 Web interface

The IPFL system also exports a convenient web user interface to assist operators. It consists of a table consisting of the following columns. Figure IV.5 shows a live screenshot of the SCORE web interface updating each and every time, a new set of events are generated for real-time diagnosis. The interface also allows viewing archived logs including raw events and their associated diagnosis results. The first column lists the actual event start time and the end time using one of the clustering algorithms. The second column represents the set of links that were impacted during according to the reported alarms during the failure scenario. The third and fourth columns give descriptions of the groups of components that form the diagnosis report for that observation. The diagnosis report also consists of the hit ratio, coverage ratio and finally error threshold used for the groups involved in the diagnosis.

IV.C.7 Implementation details

The SCORE algorithmic engine loads (and periodically refreshes) an SRLG database that defines the associations between SRLGs and links. It constructs two hashtables: one for the set of circuits and one for the set of groups. Each group consists of the circuit identifiers that can be used to query the circuits hashtable. This particular implementation allows for fast associations and traversals to speed up the implementation of the SCORE algorithm outlined in Section IV.B. This SCORE engine also has a server that listens at a particular port to which various diagnosis agents can connect via popular socket interface [88] and input a failure signature. The SCORE engine then responds with the hypotheses that best explains the failure signature.

The main function of the SRLG database module is to obtain risk groups from different databases that contain fiber, fiber-span, router, and other SRLGs. The other function the SRLG database provides is grouping aliases. The algorithm for grouping aliases is not a performance bottleneck as it is refreshed fairly infrequently (usually twice a day). Thus, the SRLG database module is implemented in Perl. The code for clustering events together into a failure signature is also implemented in Perl. Thus, the total number of lines of Perl code for the entire system including all the individual modules is greater than two thousand.

IV.D Simulated faults

We evaluated the performance of the SCORE algorithm using both artificially generated faults as well as real fault data. In this section, we investigate the performance of the algorithm on the artificially generated faults; the following section reports the performance evaluated using real network faults. The main goal of the initial experiments is to evaluate the accuracy of SCORE within a controlled environment by using emulated faults. We use an SRLG database constructed from the network topology and configuration data of a tier-one service provider's backbone. We then simulate different numbers of simultaneous faults that are injected into SCORE. We also study the efficacy of the SCORE in the presence of noisy data by simulating errors in the SRLG database and the failure signature.

IV.D.1 Algorithm accuracy

To evaluate the accuracy of SCORE, we simulated scenarios consisting of multiple simultaneous failures and evaluated the accuracy in terms of the number of correct hypotheses (faults correctly localized by the algorithm). We randomly generated a given number of simultaneous failures selected from the set of all network risk groups: the set of all SONET components, fiber spans, OSPF areas, routers, and router ports and modules in our SRLG database. Once the faults were selected for a given scenario, we computed the union of all impacted links. These link failures were then input to the SCORE algorithm and hypotheses were generated. The resulting hypotheses were compared to the actual injected failures. We use the term *accuracy* to denote the fraction of injected failures that are found in the hypothesis output by SCORE.

Figure IV.6 depicts the accuracy of localization as a function of the number of injected faults, where each data point represents an average across 100 independent simulations. The figure illustrates that the accuracy of the algorithm on these data sets is greater than 99% for ports, modules and routers, irrespective of the number of simultaneous failures generated. In general, the accuracy of the algorithm decreases as the number of simultaneous failures increases, although the accuracy remains greater than 95% for less than five simultaneous failures. In reality, it is extremely unlikely that more than one failure will occur (and be reported) at a single point in time. Thus, for failures such as fiber cuts, router failures, and module outages (corresponding to a single simultaneous failure), our results indicate that the accuracy of the system is near 100%.

However, it is entirely possible in a large network that multiple indepen-

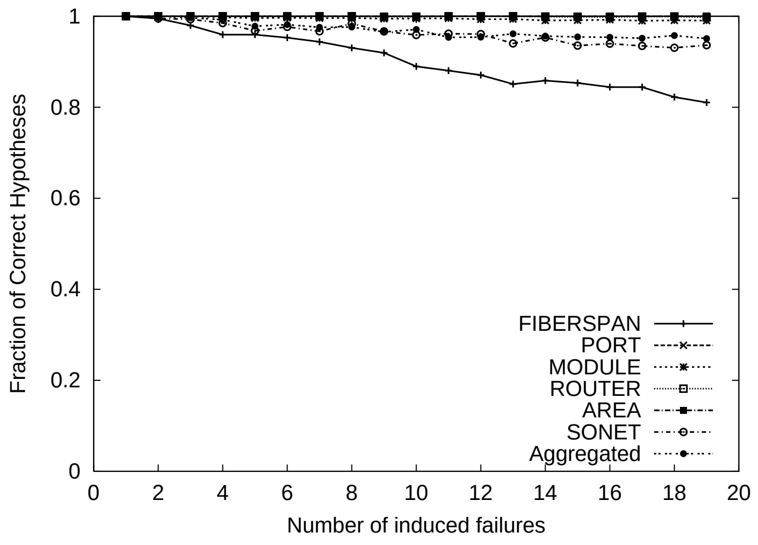


Figure IV.6: Fraction of correct hypotheses as a function of increasing number of injected simultaneous faults.

dent components will simultaneously be experiencing minor performance degradations, such as error rates, which are reported and investigated on a longer time scale. Thus, the results representing higher number of simultaneous failures are likely indicative of performance troubleshooting. However, we can still conclude that for realistic network SRLGs, SCORE algorithm is highly accurate when we have perfect knowledge of our SRLGs and failure observations.

IV.D.2 Imperfect fault notifications

The SRLG model provides a solid, but not perfect representation of the possible failure modes within a complex operational network. Thus, we expect to find scenarios where the set of observations cannot be perfectly described by any SRLG. Similarly, data loss associated with event notifications and database errors are inherent operational realities in managing large-scale IP backbones. In this section, therefore, we evaluate the accuracy of the SCORE algorithm in the presence of such losses and errors in both failure signature as well as risk model. We consider three parameters: the error threshold used in the SCORE algorithm, the number of simultaneous failures, and the loss probability (which represents the

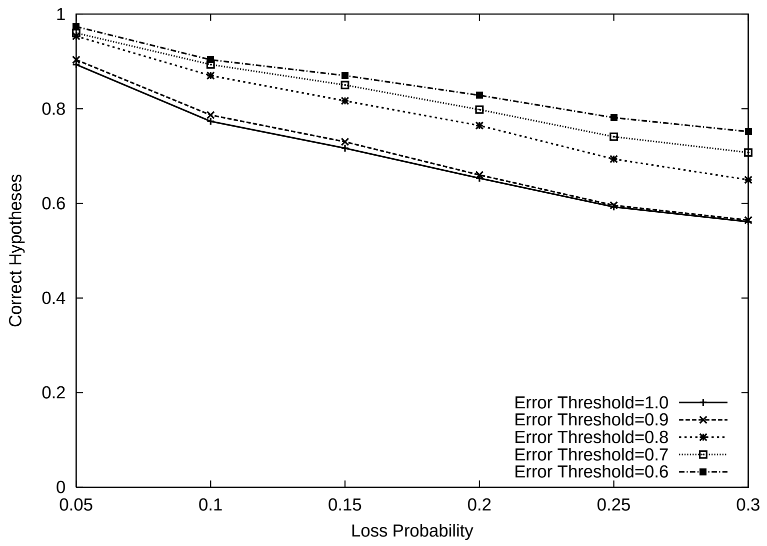


Figure IV.7: Accuracy as a function of loss probability for different error thresholds for three failures.

percentage of IP link failure notifications lost for a given failure scenario).

Figures Figure IV.7 and Figure IV.8 demonstrate the accuracy of the algorithm under a range of loss probabilities and algorithm error thresholds and for different number of simultaneous failures. Specifically, the figures plot the percentage of correct hypotheses as a function of the loss probability. In Figure IV.7, the algorithm error threshold is varied from 0.6 to 1.0, whilst the number of simultaneous failures is set to three. In Figure IV.8 the algorithm error threshold is fixed at 0.6 and the number of simultaneous failures is varied from one to five. As expected, increasing the loss probability reduces the accuracy of the algorithm. Under three simultaneous failure events and an loss probability of 0.1, we can observe from Figure IV.7 that an algorithm error threshold of between 0.7 and 0.8 restores the accuracy of the SCORE algorithm to around 90%. However, if we mandate perfect matching of failure observations to SRLGs (i.e., error threshold = 1.0), then our accuracy in isolating our fault drops to around 78%. This shows the necessity and effectiveness of the of the error thresholds introduced into the algorithm for fault localization in the face of noisy event observation data.

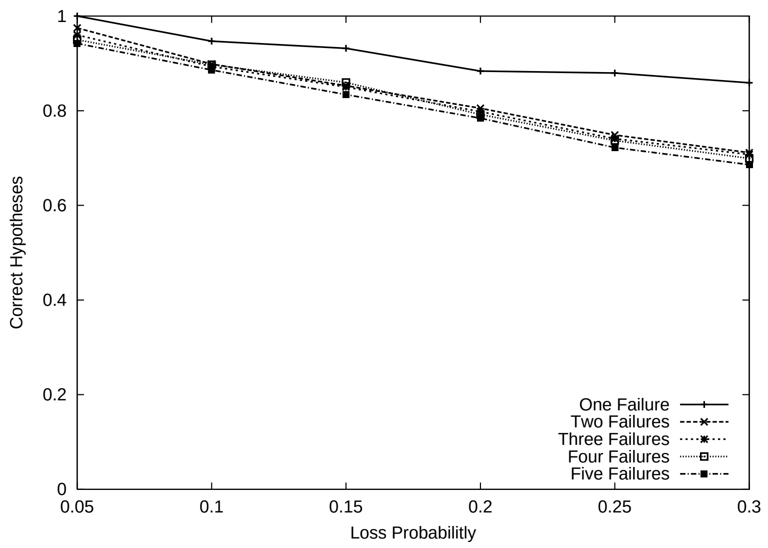


Figure IV.8: Accuracy as a function of loss probability and for varying number of simultaneous failures, with a fixed error threshold = 0.6.

IV.D.3 Performance results

The algorithm's execution time was also evaluated under a range of conditions. In general, execution time increases as the number of IP links (observations) impacted by the failures increases. This increase is because all of the SRLGs associated with each of the failed links must be included as part of the candidate set of SRLGs for localization, and thus must be evaluated. Thus, the execution time increased within increasing numbers of failures, but on average was below 150 ms for up to ten failures. Similarly, the execution time for scenarios involving router failures was typically higher than for other failure scenarios, as the routers typically involved larger numbers of links. Execution times of up to 400 ms were recorded for events involving large routers. However, even in these worst-case scenarios, the algorithm is more than fast enough for real-time operational environments.

IV.E Experience in a tier-one backbone

The IPFL system prototype implementation based on the SCORE algorithm has been deployed in a tier-one backbone network and is being used in an offline fashion to isolate IP link failures reported in the network. In this section, we discuss our experience with IP link failure events reported in router syslogs. Determining whether or not SCORE correctly localized a given fault requires identification of the root cause of the fault via other means. In many cases, identifying root causes involves sifting through large amounts of data and reports—a tedious process at best.

Therefore, we performed our evaluation in two parts: First, we evaluated localization accuracy of SCORE, by identifying the root cause of a set of 18 faults and comparing with the output reported by SCORE algorithm. Second, we studied localization efficiency of SCORE, in terms of the amount of reduction in the number of root causes, using about 3,000 different faults from real failure data in an automated fashion. For these failures, however, we did not manually verify the accuracy of our SCORE output.

IV.E.1 Localization accuracy

Table IV.2 denotes the results of our analysis of each of our 18 faults. For each failure scenario, we report:

- a name uniquely identifying the failed component,
- the number of SRLG groups localized when the algorithm was run with a threshold (t) of 1.0,
- the threshold used to generate a final conclusion,
- the number of SRLGs localized when the algorithm was run with the final threshold,
- the number of SRLGs correctly localized, and
- description of the reason why we had to reduce the threshold, or why we were unable to identify a single SRLG as the root cause in certain situations.

Table IV.2: Summary of failures we have observed in various traces.

Component name	size (t=1)	Final Thld	size (t=final)	correct	comment
Router A	27	0.8	1	1	No event reported by some links
Router B	20	0.9	3	3	No event reported by some links
Router C	12	0.7	1	1	No event reported by some links
Router D	1	1	1	1	-
Router E	18	0.8	1	1	No event reported by some links
Router F	1	1	1	1	-
Router G	4	1	4	4	-
Module A	1	1	1	1	-
Module B	1	1	1	1	-
Module C	1	1	1	1	-
Sonet A (OA)	8	0.9	2	1	No event reported by one link and database problem
Sonet B (Transceiver)	1	1	1	1	-
Sonet C (Flap)	2	0.7	1	1	No observation reported by one link
Sonet D (OA)	2	0.6	1	1	No observation reported by one link
Fiber A	3	0.5	1	1	Database problem
Fiber Span A	1	1	1	1	-
OSPF Area A	20	0.7	4	4	Incorrect SRLG modeling
OSPF Area B	4	1	4	4	OSPF Area A MPLS enabled interfaces

Overall, we were able to successfully localize all of the faults studied to the SRLGs in which the failed network elements were classified—except where we encountered errors in our SRLG database. However, when we used a threshold of 1.0 (i.e., mandated that an SRLG can be identified if and only if faults were observed on all IP links), then we were typically unsuccessful—particularly for router failures, and for the failure involving OSPF Area A in Table IV.2. In the majority of the router failures, even though these events corresponded to routers being rebooted, the remote ends of the links terminating on these routers did not always report associated link-level events. This phenomenon could be because of a number of possible reasons:

- events may never have been logged in the syslogs,
- data may have been lost from the syslogs,
- the links may have been operationally shut down and hence did not fail at this point in time, or
- the links were (inexplicably) not impacted by the reboot.

Independent of why the link notifications were not always observed, the router failures were all successfully localized when the threshold was marginally reduced. Thus, error threshold in SCORE algorithm is important to localize faults in operational networks.

Of course, router failures are typically easy to identify through visual correlation, as all of the links impacted have a common end point (the failed router). Optical-layer impairments, however, can impact seemingly logically independent links at the IP layer if these links are all routed through a common optical component. We study four different SONET network element failures, labeled SONET A through D in Table IV.2.

The first—an optical amplifier failure—induced faults reported on thirteen IP links. With a threshold of 1.0, our algorithm identified eight different

SRLGs as being involved. However, as the threshold was reduced to 0.9, we were able to isolate the fault to only two different SRLGs. Reducing the threshold further, however, did not reduce the number of SRLGs to which the fault was localized.

Upon further investigation, we uncovered an SRLG database problem where our SONET network element database did not contain any information regarding one of the circuits impacted by the fault. Thus, the SCORE algorithm was unable to localize the fault for this particular IP link to the SRLG containing the failed optical amplifier, and instead incorrectly concluded that a router port was also involved (the second SRLG). However, the SRLG containing the failed amplifier was also correctly identified for the other twelve IP links; the lower threshold was required because no fault notification was observed for one of the IP links routed through the optical amplifier.

This optical amplifier example highlights a particularly important capability of the SCORE system—the ability to highlight potential SRLG database errors. Links missing from databases, incorrect optical layer routing information regarding circuits and other potential errors in databases play havoc with capacity planning and network operations and so must be identified. In this scenario, the database error was highlighted by the fact that we were unable to identify a single SRLG for a single network failure, even after lowering threshold using in the SCORE algorithm.

The other three SONET failures were all correctly isolated to the SRLG containing the failed network element: in two cases we again had to lower the threshold used within the algorithm to account for links for which we had no failure notification (in one of these cases, the missing link was indeed a result of the interface having been operationally shut down before the failure).

We tested SCORE on a second, previously identified failure scenario impacted by a SRLG database error (Fiber span A in Table IV.2). Again, the SCORE system was unable to identify a single SRLG as being the culprit even as the threshold was lowered, because no SRLG in the database contained all of the circuits reporting the fault. So, again, a database error was highlighted by the inability of the system to correlate the failure to a single SRLG.

The final case that we evaluated involved a low-level protocol implementation problem (i.e., software bug) that impacted a number of links within a common OSPF area, labeled OSPF Area A in Table IV.2. This scenario occurred over an extended period of time, during which three other independent failures were simultaneously observed in other areas. When SCORE used a threshold of 1.0, it attributed the failure to 20 independent SRLG failures—a large number even for the extended period of time. As the threshold was reduced to a final value of 0.7, the event was isolated to four individual SRLGs: three SRLGs in other OSPF areas (corresponding to the other independent failures) and the OSPF area in question. Thus, the SCORE algorithm was correctly able to identify that the event corresponded to a common OSPF area.

More investigation into the matter uncovered that the reason why not all links in the OSPF area were impacted was that only those interfaces that were currently MPLS-enabled were affected. Thus, an additional SRLG was added to or database that incorporated the interfaces with MPLS in a given area; application of this enhanced SRLG database successfully localized all of the SRLGs impacted by the four simultaneous failures with a threshold of 1.0.

In general, any level of SRLG modeling can be inadequate, as there could be complicated failure scenarios not modeled *a priori*. Therefore, SCORE uses the error threshold in order to be robust against such inaccuracies. We also illustrated how we can continually learn new SRLGs through further analysis of new failure scenarios, thereby enhancing SRLG modeling. Of course, for these more esoteric failure scenarios, updating the SRLG models may not always be justified unless there is some likelihood that the particular failure scenario may be repeated again in the future.

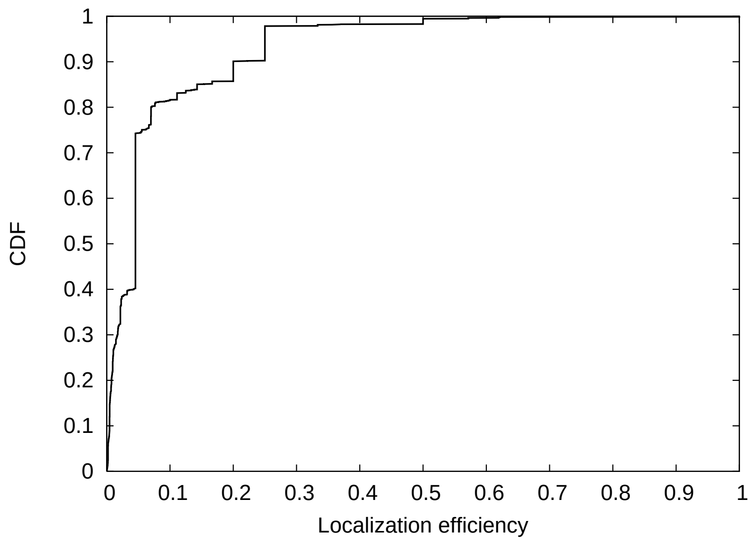


Figure IV.9: CDF of localization efficiency out of about 3,000 real faults we have been able to localize. Note that we have not verified manually the correctness of hypothesis for these faults.

IV.E.2 Localization efficiency

While the 18 faults we have studied demonstrate the ability of SCORE to correctly localize faults, they do not give an indication about the amount of reduction in the number of suspects. In this section, therefore, we evaluate SCORE using a metric we call *localization efficiency*, that is defined as the ratio of the number of components after localization to that before localization. In other words, it is the fraction of components that are likely to explain a particular fault (or observation) using our localization algorithm out of all the components that can cause a given fault.

Let $G_i = \{g_{i1}, g_{i2}, \cdots, g_{ik}\}$ denote the set of SRLGs to which a circuit c_i belongs. Therefore, for a failure signature $F = \{c_1, c_2, \cdots, c_i\}$, the set $G = \cup_{k=0}^{i} G_k$, represents the universe of all SRLGs associated with at least one circuit c_k in F. In other words, the set G represents the set of all possible root causes that can potentially explain F. Let H denote the hypothesis output by SCORE. Then, localization efficiency is given by $|H|/|G|$.

In Figure IV.9, the cumulative distribution function of the localization efficiency is shown. From the Figure IV.9, we can clearly observe that SCORE could localize faults to less than 5% for more than 40% of the failures and to less than 10% for more than 80% of the failures. We conclude that SCORE identifies likely root causes very precisely from a large set of possible causes for a given failure. We note, however, that we do not know the root cause of all 3000 faults shown here; we cannot speculate on SCOREs accuracy except for the 18 faults discussed previously.

IV.F Summary

Using the risk modeling methodology, we have developed a system that accurately localizes failures in an IP-over-optical tier-one backbone network. Given a set of IP-layer events occurring within a small time window, our heuristics pinpoint the shared risk (optical device) that best explains these events. Given the harsh operational reality of maintaining complex associations between objects in the two networking layers in separate databases, we found that it is necessary to go beyond identifying the single best explanation, and, instead, to generate a set of likely explanations in order to be robust to transient database glitches.

We put forward a simple, threshold-based scheme that looks for best explanations admitting inconsistencies in the data underlying the explanations up to a given threshold. We found that not only does this increase the accuracy and robustness of fault localization, it also provides a new capability for identifying topology database problems for which we have no alternative automated means of detecting. Getting shared-risk information correct is critical to IP network design—a misidentification of a shared risk might produce a design believed to be resilient to single SRLG failure, which in fact is not. Owing to the fundamental importance of shared risk information across network operations and capacity planning, the automated database auditing capability provided by the thresholding method is itself a significant contribution.

From an architectural perspective, however, we believe it is important to provide automated mechanisms to accurately query the cross-layer dependencies directly from the network. In Chapter VI, we qualitatively discuss the importance of cross-layer associations in major network management functions and argue that such associations should be provided as a service to many such management functions.

In the next chapter, we consider the problem of black hole localization, which is vastly different from IP fault localization. In this problem domain, the topology is huge and dynamic, thus forcing us to consider alternate algorithmic techniques to SCORE. Yet, we show that the problem can be effectively solved by applying the risk-modeling methodology.

IV.G Acknowledgments

This chapter is based on the paper titled “IP Fault Localization via Risk Modeling”, that appeared in the *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation*, held at Boston in May 2005, which is joint work with Jennifer Yates (at AT&T Labs – Research), Albert Greenberg (at Microsoft Research), and Alex C. Snoeren. The dissertation author was the primary investigator and author of this paper [54].

Chapter V

MPLS black hole localization

In this chapter, we apply the risk modeling methodology developed in Chapter III to localizing complex failure modes in current systems, known as silent failures or *black holes*. In contrast to the IP link failures in the previous chapter, the network fails to detect—and, therefore, is unable to recover from black holes, while packets are silently dropped in the network. Specifically, the black holes we refer to in this dissertation are in the context of VPN-over-MPLS-over-IP backbone networks, an architecture deployed by a number of tier-one ISPs. In this setting, packets belonging to customer VPNs are switched over MPLS tunnels, which themselves are established using standard IP routing protocols such as OSPF or IS-IS. A black-hole scenario can occur when the underlying IP infrastructure is operational, that is, each IP hop along the route is functioning properly, but the corresponding MPLS tunnel fails to deliver packets. We are particularly interested in cases where a black hole is silent in nature, with no router alarm indicating that the MPLS tunnel is actually broken.

Black holes have a variety of causes, ranging from delayed routing protocol convergence to mis-configurations to bugs in individual router implementations. While backbone networks are generally able to address each cause after an incident occurs (i.e., by asking the vendor to fix the bug), experience shows there is always another bug. Due to the ever-increasing complexity of router-control software, it is highly unlikely that they will ever disappear entirely. While such silent failures are rare, they can have a large and egregious impact; in many cases, a complete loss in VPN connectivity results in a significant financial damage to both the ISP as well as the customer. These failures are extremely time-consuming to localize (order of hours to days) because there are no alerts/alarms to guide operators to the location of the failure. Hence, from an operational stand-point, it is imperative to design a mechanism that can quickly detect such failures, and, moreover localize the root cause of the failure.

Luckily, most tier-one ISPs already conduct a significant amount of active probing within their networks to provide SLAs. In this chapter, therefore, we describe a black hole detection system that uses these end-to-end probes to detect the presence of black holes. Further, we show the efficacy of a system we designed based on the risk-modeling methodology in Chapter III to localize black holes. In particular, we analyze three known black holes in a tier-one backbone network. We show that, had our localization system been in place at the time of the failures, they would have taken considerably less time to detect and localize. In order to validate our methodology further, we collect probe data resulting from a large number of (less serious) network events (e.g., routing changes) that are usually logged by routers in syslogs. By definition, such failures are not silent in nature, but this approach gives us a reasonably large—albeit non-random—set of cases to test our technique. Note that the majority of these events are very short-term, which makes them hard to detect and localize using active probing mechanisms. Even with this challenging data set, we find that our technique is highly successful in localizing the problems; we achieve greater than 80% accuracy and 80% precision across a wide range of failure scenarios.

The remainder of this chapter is organized as follows. We begin by discussing the silent failure problem, our approach to detect and localize them, and finally the system architecture in the next section (Section V.A). We evaluate the system using simulations in Section V.B and using real failure data from a tier-one ISP network in Section V.C followed by related work in Section V.D and a summary in Section V.E.

V.A Silent network failures

As mentioned before in Chapter II, many customer VPNs are overlaid on top of MPLS tunnels between various provider-edge routers within an ISP network. Moreover, MPLS tunnels are often configured to route along the underlying shortest path between end points as dictated by the IGPs such as OSPF and IS-IS. Given the cross-layer dependency between MPLS and IGPs, any topology change at the IP layer, either due to IP link weight changes or link failures, automatically triggers rerouting at MPLS layer along the new IGP shortest paths. This signaling across layers allows the MPLS layer to automatically heal around the failure, thus providing a first line of defense against link failures.

Unfortunately, there is no guarantee that this first line of defense always succeeds. For example, one failure scenario that has been observed in practice is when OSPF re-routes due to a problem on a link but MPLS does not re-route. In such cases, MPLS forwarding table entries are not updated to reflect the link failure; many MPLS tunnels continue to have the failed link as the next hop in the forwarding table. Therefore, the forwarding element within the router continues to switch MPLS packets to the old interface (that is currently down), which are subsequently dropped by the interface hardware, thus resulting in a black hole. In other failure instances, the MPLS control plane is working properly (hence, no alarm), yet there is corruption in the forwarding elements due to bugs in the implementation and/or configuration errors. Many other subtle failure scenarios at the MPLS layer can be found in [30]. While router vendors continue to develop work-arounds to deal with such failure scenarios, it is unlikely such problems are going to disappear completely.

Given that routers do not automatically detect and alarm on black holes,

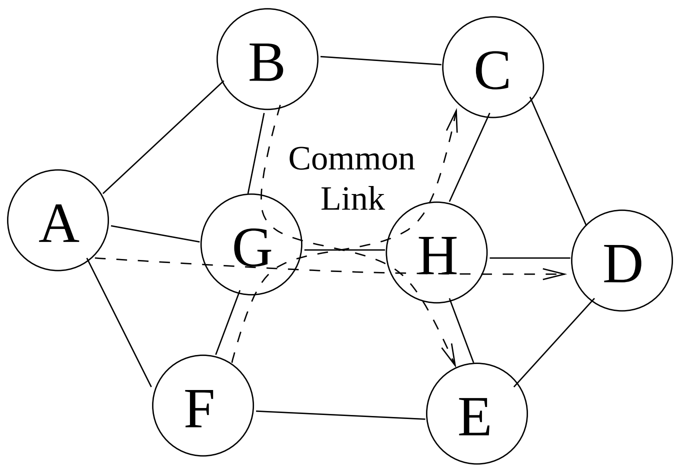


Figure V.1: Example topology with failure impacting a set of paths going through a given link (G-H).

network operators typically rely on a fault detection system that uses an end-to-end approach [76] to detect black holes. Edge routers or special measurement servers connecting to them issue probes periodically that test connectivity to other edge routers and report failures when probe packets are not acknowledged by their destination. While these end-to-end probes detect black holes, they do not determine the exact location of the failure, which is required by the operator to fix the problem and recover traffic. Thus, both fault detection and fault localization are important steps that need to be performed before operators can recover the network from silent failures.

The key idea we use for black-hole localization is risk modeling as discussed in Chapter III. A detection system based on active probes identifies the set of origin-destination pairs (OD-pairs) impacted by a black hole in the network. It then forms the failure signature by clustering together OD-pairs that are temporally correlated. The OD-pairs in the failure signature are then spatially correlated by the localization algorithm to locate shared links along the paths. For example, in Figure V.1, nodes A through F form the edge nodes that are connected via intermediate nodes G and H. Nodes A to F are being monitored using the active measurement mechanism described earlier. If the set of paths A-G-H-D, F-G-H-C, B-G-H-E all fail in the same time interval (temporally correlated failures), spatial correlation leads to the link that is common to all these paths—the link from G to H.

In contrast to IP-layer faults, where traffic itself is often not impacted because routers can re-route traffic around the link failure, black holes result in many customer VPNs disconnected for extended periods of time, before operators can detect, localize and recover traffic. Operational experience shows that the total customer down-time is dominated primarily by the detection and localization step; once the problem is localized, it can often be quickly recovered (e.g., by rebooting the interface). In many cases, the actual repair can be completed off-line. The rest of this section formally describes the problem and explains the detection and localization steps in more detail.

V.A.1 Problem formulation

We consider a graph $G = \{V, E\}$ with vertex set V and edge set E that corresponds to the topology of the network. Denote A as the subset of the vertices, $A \subseteq V$, whose connectivity to each other we are interested in monitoring. In essence, these vertices represent the MPLS tunnel end points, between which black holes can occur. On this graph, between every origin u and destination v, $(u, v \in A)$, there exists a set of paths $P_u^v \subset P$, where P is the set of all paths between every OD-pair, that correspond to the IGP shortest paths between u and v. In real networks, P varies with time with topology changes.

In our failure model, we assume that a path $p \in P$ fails if any of the links $l_{i1}, l_{i2}, ..., l_{ik} \in E$ on that path fail, because the connectivity of a given path is directly dependent on only the links along that path. Conversely, if any edge fails, all the paths $p_i \in P$ that contain the edge fail. Given an input failure signature F consisting of a set of OD-pairs $(u, v) \in F$, the problem is to identify a hypothesis $H = \{l_{j1}, l_{j2}, ..., l_{jm}\} \in E$, consisting of all the candidate links that can explain the given failure signature F, and also matches the ground truth as much as possible.

Similar to the IP fault localization problem, we reduce the black-hole localization problem to the formulation in Chapter III. At first glance, this reduction seems straightforward; the symptoms comprise the OD-pairs that have lost connectivity and individual links along the path(s) between a given OD-pair correspond to the root causes. However, in contrast to the IP fault localization problem, there are a few additional issues such as dealing with multiple paths between OD-pairs, incomplete failure signatures and so on, that need to be taken into account for the reduction. We discuss these issues individually in the context of fault detection, construction of the risk model and fault localization algorithm, in the following subsections.

V.A.2 Fault detection

The goal of the fault detection system is to detect any black hole condition in the network that effects traffic between various OD-pairs. The monitoring system, therefore, injects end-to-end MPLS bi-directional probes periodically between all OD-pairs of interest. If probes do not get back to the origin within a certain timeout, the system determines that the probe has been lost between this OD-pair. The system forms the failure signature consisting of the OD-pairs between which probes are lost.

In practice, many equal-cost IGP paths can potentially exist between a given OD-pair, typically referred to as equal cost multi-path (ECMP) [37]. In such cases, the router splits traffic based on a hash of certain packet header fields such as source and destination IP addresses. Even if one path experiences a black hole, that would result in disconnecting some of the customer VPNs that traverse that particular MPLS path. While operators are, therefore, interested in monitoring all the paths, probes only monitor a single path between a given OD-pair. Depending on whether the packet exercised the failed path, the probe may or may not detect that a given OD-pair is black holed. Thus, the fault detection system inherently detects only *partial* failure signatures during black holes.

Of course, one could send multiple probes with varying header fields (typically source or destination IP addresses), so that different packets potentially hash to different routes. A sufficient number of such probes can exercise all paths that exist. However, this approach increases the probe bandwidth in the network besides requiring extra administrative effort in maintaining multiple IP addresses per router.

Another reason for partial failure signatures stems from the low frequency of probes. In order for a probe to observe a black hole, it needs to be transmitted within the duration of the black hole; of course, if the black hole remains persistent, eventually all probes will observe it. So, depending on the duration of the black hole and the timing of the probes, the set of lost probes may or may not contain all the OD-pairs really affected by the black hole.

Black holes cause every packet to be dropped at the failed interface, including the probe packets. However, a lost probe does not necessarily mean the presence of a black hole; probes can be dropped because of non-black-hole reasons as well, such as routing loops or congestion in the network, which cause transient packet losses in the network or in short, *noise*. Noise can, in many situations, be thought of as a result of link(s) that drop packets with some probability for a very short duration of time. Thus, the set of lost probes during a black hole can include both due to the black hole as well as noise. Since the fault detection system cannot differentiate between black holes and other transient failures, the failure signature typically consists of a mix of both signatures.

We now present a formal description of the fault detection mechanism. Define the *true link failure signature* $F_l(t)$ as the complete set of OD-pairs that would be impacted as a result of a failure of link l at time t. In other words, if any path $p \in P_u^v$ contains a link l at time t, then the OD-pair $(u, v) \in F_l(t)$. Note that a link failure signature's dependence on time is the result of flux in topology either due to operational maintenance activities or previous link failures.

Next, we define an *aggregate true failure signature* $\Phi(t)$ at any time instant

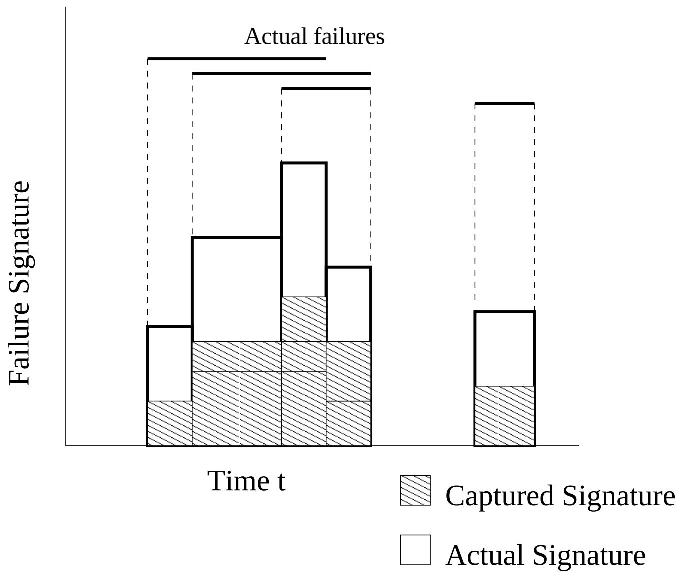


Figure V.2: The aggregate failure signature, $\Phi(t)$, changes as a result of overlapping failures. The shaded boxes indicate the portion of the failure that we detect using active measurement.

t as the union of all true link failure signatures of links that are in a failed condition (i.e., the link drops all packets) at time t. In Figure V.2, we show pictorially how the aggregate failure signature $\Phi(t)$ can increase as the number of overlapping failures increases. In the Figure, we can observe that $\Phi(t)$ is in fact a piece-wise constant function, with each piece indicating a given set of links that have failed in that time interval. Therefore, as a new link is added to the actual failure set, the number of tunnels that are dependent on the new link are added to $\Phi(t)$, thus increasing it's overall size. Similarly, as the failed link becomes functional again, the set of associated tunnels are removed resulting in a decrease in the size of $\Phi(t)$. When there are no more active failures, $\Phi(t)$ reduces to the null set.

Although, ideally, we would like to obtain the entire $\Phi(t)$ at every instant of time, it is often not practical since item the number of probes we can transmit through the network is constrained by CPU and bandwidth resources. Instead, the detection system captures only a subset of $\Phi(t)$ using active probes, both because the probes are at lower frequency as well as because of the multi-path effects. Therefore, the *measured failure signature* $\phi(t)$ is the part of the aggregate true failure signature at time t. In Figure V.2, the measured failure signature, as shown in the shaded boxes, is a subset of the aggregate true failure signature.

Note that the detection system only outputs a stream of probe losses during the failure. In order to form the measured failure signature, we need to devise a mechanism to group together probes that are lost due to the same root cause. However, given that multiple link failure signatures can overlap, it is not straightforward to separate them apart directly. One possible option is to use a similar clustering algorithm to that in IP fault localization (in Section IV.C.5), in which a continuous chain of events separated in time by less than a threshold are clustered together. While such a clustering algorithm worked well for IP faults, it holds less promise in this domain because of noisy probe losses that happen on a continuous basis in the network. Thus, if we apply the clustering algorithm above, there will always be some probe loss due to noise that will occur within a time less than the threshold, thus getting added to the cluster. In effect, every probe loss could potentially get clustered together, even those that are totally unrelated, thus defeating the purpose of clustering and complicating the localization step.

Therefore, we instead divide time into equal-size bins of, say, 15 minutes and consider the set of probe losses within the bin to form the measured failure signature that is subject to localization. Note that this approach still does not guarantee that individual link failure signatures are extracted. Indeed, the onus still remains on the localization algorithm to produce meaningful hypothesis with overlapping failure signatures. Clearly, either a given signature is completely within the time bin, or the signature straddles bin boundaries. In either case, at least half of the signature ends up becoming part of either of the two bins. As we shall show later in Section V.B, in many cases, a reasonable fraction (>10%) of the failure signature is enough for the localization algorithm to disambiguate the root causes. Thus, binning works fine for forming failure signatures. Next, we discuss how to construct the necessary risk model.

V.A.3 Risk model construction

As mentioned before, it is plausible to construct the risk model in a similar fashion to the IP fault localization problem. There are a few additional challenges, however. The first challenge is to determine the dependencies between the OD-pairs representing MPLS tunnels (symptoms) and the root causes of black holes (i.e., links). Because MPLS tunnels follow IGP shortest paths under normal conditions, we can use links along these shortest paths as root causes that can affect the particular tunnel. In order to obtain the shortest paths, we rely on an OSPF monitor [79] to passively record the link-state advertisements and compute shortest paths between every pair of nodes in the network. However, due to the dynamic nature of this topology, there could be many different topology snapshots during the particular failure interval. Unfortunately, it is not known *a priori* which topology contains the location of the fault. Therefore, the fault localization algorithm needs to consider multiple risk models using different topology snapshots.

The second challenge involves cases when multiple shortest paths exist between two end points. Unfortunately, since it is not possible to determine the exact path, we take a conservative stance and include the union of all links that belong to every possible shortest path between the OD-pair as possible root causes associated with the symptom.

V.A.4 Fault localization

Using the bipartite risk model and the detected failure signature, the fault-localization algorithm outputs an appropriate hypothesis that explains all the observed symptoms. While our localization engine is similar in spirit to SCORE (and more generally to the greedy approach explained in Chapter III), there are several key differences in our problem domain that prevent using either SCORE's greedy approximation to the set cover problem or even the Bayesian approaches in [45, 86]:

- **Lack of complete information.** In the problem domains considered previously, the failure signature consists of all impacted IP links. Due to our active measurement methodology, we are unlikely to collect the full set of impaired links, leading to a partial failure signature in many transient blackhole scenarios. Hence, algorithms that rely on the assumption that the lack of reported failure on a particular path implies that all the constituent links are good are not appropriate for our scenarios.

- **Noisy failure data.** A second major difference is the presence of noise that can bias localization towards large hypotheses with quite a few spurious candidates. We address this problem by subjecting the output of the localization algorithm to additional filtering described in Section V.A.5.

- **Scale.** The topology is much larger (in terms of the number of end-to-end paths) and dynamic in nature; calculating paths on-the-fly between all OD-pairs for computing the appropriate risk model for localization of every fault is impractically time consuming. This reality forces us to again deviate from algorithms that require considering all OD-pairs during diagnosis. We restrict ourselves only to the set of OD-pairs that indeed failed.

- **Redundant paths.** Finally, there can be multiple paths between an OD-pair due to equal cost multi-path—a scenario that does not exist in the earlier IP fault localization problem studied in Chapter IV. As we mentioned before, we address this problem by considering the union of all the paths between a given OD-pair at the instant of the failure. We will show that this approximation, while conservative, works well in practice.

Our localization algorithm called MAX-COVERAGE is derived from the same basic greedy approach defined in Algorithm 1 that we used for SCORE. The only place where it differs from the SCORE algorithm is in the two black-box routines left undefined in the original GREEDY in Algorithm 1—`calculateMetrics`()

Algorithm 4 `calculateMetrics(rootCauseVector R, failureSignature F)`

```
1: for (rootcause r ∈ R) do
2:    for (symptom s ∈ F) do
3:       if (checkAssociation(r, s)) then
4:          r.coverage++;
5:       end if
6:    end for
7: end for
```

Algorithm 5 `selectCandidates(rootCauseSet R)`

```
1: max = 0;
2: candidateSet = {}
3: for (rootcause r ∈ R) do
4:    if (r.coverage > max) then
5:       max = r.coverage;
6:       candidateSet = {r};
7:    else if (r.coverage == max) then
8:       candidateSet = candidateSet ⋃ {r};
9:    end if
10: end for
11: return candidateSet;
```

and `selectCandidates()`, which are defined for MAX-COVERAGE in Algorithms 4 and 5 respectively. MAX-COVERAGE iteratively picks the root cause (link) that explains the most number of symptoms in the failure signature, prunes this set of events from the failure signature and repeats the process until no more events remain in the failure signature. Therefore, for each root cause, the routine `calculateMetrics()` associates the number of symptoms in the failure signature explained with that particular root cause as the metric for ranking. Note that the `checkAssociation()` routine checks whether the a given symptom is associated with a given root cause according to the risk model. MAX-COVERAGE is biased in favor of link(s) that explain the most symptoms in the failure signature. The routine `selectCandidates`(), therefore, picks those candidate links have the highest value of the metric computed earlier. The rest of the steps proceed similar to the basic GREEDY in Algorithm 1.

V.A.5 Additional issues

From the network operator stand point, we need to address two additional issues. First, the core localization algorithm MAX-COVERAGE outputs a hypothesis given a topology and a failure signature. However, routing changes during the time interval of interest can result in multiple topologies that must be accounted for. In order to localize failures in such scenarios, we need to use the route before and after each routing change. If multiple route changes occur within a failure interval, applying different topologies results in different hypotheses for the same failure signature. Hence, we need a mechanism to combine all these hypotheses into one that the network operator can use for localization. Second, MAX-COVERAGE generates an explanation for every observation in the failure signature, even those due to noise, thus resulting in an unnecessarily large hypothesis making it cumbersome to the operator. We need a mechanism to reduce the size of the hypothesis further so that the system is more effective and usable to the network operator.

To address these issues, we first generate multiple hypotheses for a given failure signature using all the available topology snapshots obtained from the OSPF monitor in the failure interval, and apply the following two algorithms in sequence to output the final hypothesis:

- *A hypothesis selection algorithm* for selecting a hypothesis across different topology snapshots; and
- *A candidate selection algorithm* for selecting candidate links within the hy-

pothesis (based on the contribution of each failure in the hypothesis to the observation set) to deal with noise.

A hypothesis selection algorithm that has access to the ground truth (we call it ORACLE) can easily pick the best hypothesis that is closest to the truth. In practice, however, we do not have access to the ground truth, and hence ORACLE is not an algorithm we can implement in practice. Thus, we need an online approximation to ORACLE that can select the best possible hypothesis among seemingly plausible competing hypotheses. One such algorithm we found to be a good approximation is UNION (shown later in Section V.C.3), that computes the union of all the hypotheses with different topology snapshots.

For candidate selection, a natural choice is to use a threshold. We can use one of two standard types of thresholds—absolute and relative—to filter out candidates due to noise. We can use an absolute threshold to pick those candidate links that explain greater than a threshold number of events in the failure signature. On the other hand, using a relative threshold, we can pick candidates that explain greater than a particular fraction of all the events within the failure signature. In general, either of these thresholding mechanisms work fine. We, therefore, arbitrarily chose to use an absolute threshold to filter out candidates that explain observations less than the threshold. Putting all these components together, the overall system architecture is explained next.

V.A.6 System architecture

In Figure V.3, we show the complete data flow in the fault localization system. Each edge router issues probes to $(n-1)$ (n is the total number of routers) other destinations and report the probes that get lost to the monitoring server. The monitoring server invokes the localization algorithm with the failure signature obtained from the detection system and obtains hypothesis corresponding to each topology snapshot for that failure interval. The topology snapshots themselves are constructed from link-state advertisements obtained from the OSPF monitor. It

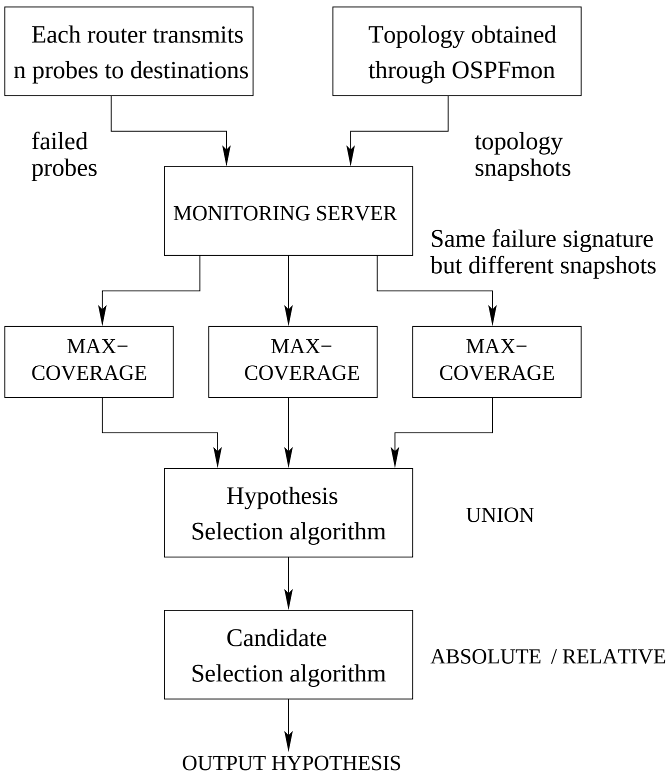

Each router transmits n probes to destinations
Topology obtained through OSPFmon
failed probes
topology snapshots
MONITORING SERVER
Same failure signature but different snapshots
MAX−COVERAGE
MAX−COVERAGE
MAX−COVERAGE
Hypothesis Selection algorithm
UNION
Candidate Selection algorithm
ABSOLUTE / RELATIVE
OUTPUT HYPOTHESIS

Figure V.3: System architecture

then uses the hypothesis selection algorithm followed by the candidate selection algorithm to output the final hypothesis that the operator uses to perform further diagnosis. In the next few sections, we evaluate this system using both simulation and offline analysis of real failure data, similar to the evaluation in IP fault localization.

V.A.7 Challenges with real-time tools

One more level of complexity is introduced when tools operate in real time with streaming symptoms arriving in real time. For offline data, clearly demarcating failure boundaries is relatively easier as compared to streaming data, because of the dilemma that the tool has to face—whether to wait for more symptoms or to proceed with what it has. We solve this using the following techniques.

- *Maintaining history*: We maintain a small amount of history consisting of symptoms that belong to the last cluster of events. As new events arrive, we determine whether to add these events to the last cluster or to start a new cluster.
- *Incremental hypothesis*: We generate incremental hypothesis with symptoms collected thus far. As new symptoms arrive, we check whether the new symptoms belong to the same cluster as the history. If they do, the hypothesis is refined with new evidence by re-running the algorithm with the larger cluster.
- *Sampling*: Sometimes, the number of symptoms could be too large to generate a quick diagnosis. For example, during black holes, we have observed as many as five thousand symptoms, for which calculating paths on-the-fly for all these failed probes is difficult. So, we reduce the size of the failure signature by randomly picking symptoms to feed the localization algorithm.

While we have built this real-time version of the tool, for our evaluation with real failure data, we operate the tool in an offline fashion.

V.B Simulation results

To evaluate the performance of the detection and localization system, the output of the localization algorithm needs to be compared against the ground truth. Often, however, such ground truth for many failure scenarios is hard to obtain. Therefore, we built a simulator that can inject artificial failures that mimic real-life failure scenarios, and compare the output of the algorithm with the injected failures using the metrics—accuracy and precision.

V.B.1 Metrics for comparison

Accuracy is the fraction of elements in ground truth G also contained in the hypothesis H, or $|G \cap H|/|G|$. If G is a proper subset of H, then the accuracy is 1. This metric alone can not capture the efficacy of the localization algorithm, however. For example, if we design an algorithm that always outputs U where U is the universal set of elements, then $G \subseteq U$ by definition, thus always leading to an accuracy of 1. Such an algorithm obviously is not very useful in practice.

Therefore, we define another metric called *precision* that quantifies the size of the hypothesis in relation to the ground truth. It is defined as the fraction of elements in the hypothesis that are also present in the ground truth or $|G \cap H|/|H|$. In effect, precision captures the amount of truth in the hypothesis. For example, a precision of 0.9 would imply that the 90% of the elements in the hypothesis match the ground truth.

High accuracy in a localization system means few false negatives, and high precision implies few false positives. Typically, most algorithms tend to trade one metric for the other depending on how conservative or aggressive the algorithm is. A conservative algorithm tends to include all the possibilities in order to achieve better accuracy while losing precision, while an aggressive algorithm includes only the significant ones thus gaining precision while somewhat sacrificing accuracy. Our goal is to ensure that both these metrics are within reasonable bounds.

V.B.2 Simulation methodology

In this subsection, we discuss the architecture of our simulator before we proceed to the simulation results. The essential ingredients of our simulator include:

1. *Topology.* In order to produce observations that are similar to that of the real-life fault observations, the simulator needs a topology model. Since topology is an important component in evaluating the efficacy of any heuristic or algorithm, we decided to use real topology data. We used a section of a real tier-one ISP backbone network that is MPLS switched.

2. *Failure model.* In our failure model, each link failure is characterized by three parameters—the start time of the failure, duration and finally, the nature of failure (soft or hard). Typically, black holes are hard failures, so, every packet that traverses the black-holing interface is dropped. On the other hand, soft failures are used to model any type of transient probe losses, and thus, packets are dropped with some configured probability.

3. *Detection model.* For detection, our simulator can generate a failure signature in one of two ways. First, the simulator can generate individual probes between all the OD-pairs, and for each link along one particular path, fail the probe if the link is in failed condition according to the failure model. It can also directly generate a random fraction of the true failure signature, and output it as the failure signature. While either of the two approaches work just as well, in our experiments, we mainly used the random fraction approach.

4. *Noise model.* Real networks are noisy. So, spurious packet losses in the network can get mixed with the actual failure signatures. The simulator is equipped with modeling noise events according to many different models. For example, the simulator generates *random noise* by failing probes that belong to a random set of OD-pairs. Noise can alternatively be modeled as a set of link failures with extremely short duration, so only a few OD-pairs are affected depending on when the probe traverses that particular link failure. We call this type as *structured noise.*

The simulator provides a realistic environment in which to evaluate detection models and diagnosis algorithms. We simulated mainly three different scenarios: without any noise, with random noise, and with structured noise. The scenarios without any noise, while unrealistic, determine an upper bound on the accuracy of the algorithm. Random noise simulates failure scenarios where the failure signature is mixed with spurious probe losses in the network. In our simulations, we added a random number of spurious observations with an average of 80 per failure. We arrived at the number 80 based on our experience with real data. Structured noise, on the other hand, models scenarios where failures of short duration overlap with the main failure(s) and appear as noise. As an arbitrary starting point, we generated structured noise by failing five links at random for five seconds; the simulated "real" faults last for 60 seconds.

For each of these experiments, the following steps were used to obtain a failure scenario and then identify the corresponding $\phi(t)$.

- For k overlapping failures, pick k links at random from the link set to form the true failure scenario. We vary k from one to five in our experiments. This forms the ground-truth set.

- Assign a random start time to each of these failures, such that all these failures are contained with in the 90 second bin. Of course, the duration of the outage has been fixed at 60 seconds for the main failures of interest.

- Compute the failure signature for every change point depending on the start and end times of the link failure. For example, if a link A fails in the interval (s_1, e_1), and B in the interval (s_2, e_2) and that $s_1 < s_2 < e_1$. Then, $\Phi(t)$ is a piece-wise constant function (similar to Figure V.2), so, between s_1 and s_2, $\Phi(t)$ consists of only the failure signature of A. On the other hand, $\Phi(t)$ consists of the union of failure signatures of A and B between s_2 and e_1, since both failures are active during this interval. In case we simulate structured noise, we add these short outages into the failure signature $\Phi(t)$.

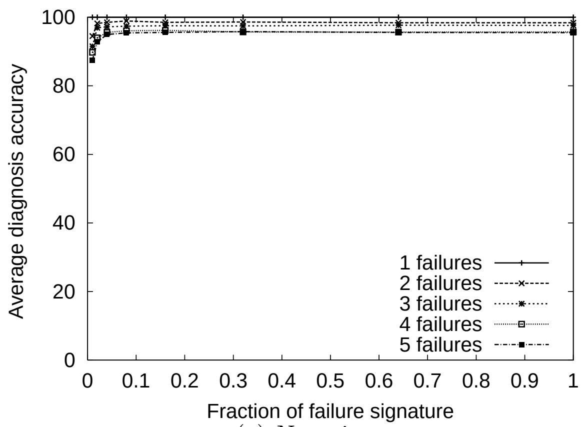
- We simulate ECMP by making the reverse mapping from a given link to its failure set, probabilistic, depending on the number of paths between a given origin and a destination that contain A. For example, if a link is present on half of the multiple paths between a given OD-pair, then the OD-pair belongs to the failure signature of A with probability 0.5.

- In each piece-wise constant interval, each OD-pair from $\Phi(t)$ is added to $\phi(t)$ with a probability $\alpha/60 \cdot d$, where d is the duration of that interval and α is the target fraction of the signature we would like to obtain.
- If we are simulating random noise events, we picked random OD-pairs and added them to the $\phi(t)$ to obtain our measured failure signature.
- The localization algorithm is run on this measured signature $\phi(t)$ to obtain a hypothesis.
- The hypothesis matched the ground truth if each of the links in the ground truth is also present in the hypothesis. The accuracy of the localization algorithm is the percentage of number of failure scenarios for which the ground truth matched the hypothesis. For the purposes of evaluating accuracy, we do not consider any noise events (both structured and random).

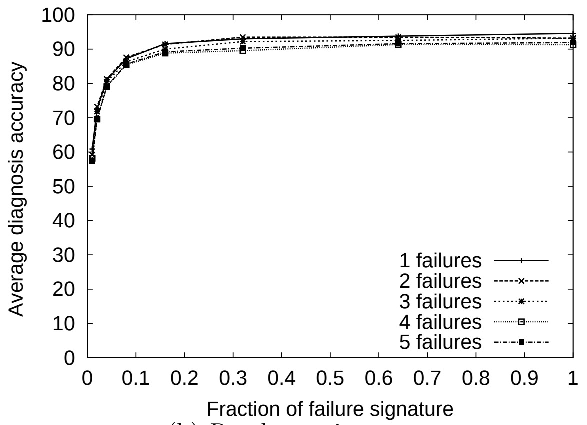
V.B.3 Accuracy of the localization algorithm

We measured the average accuracy as a function of the fraction of signature, α for varying number of simultaneous failures and for the three different types of failure scenarios. In simulations with no noise (shown in Figure V.4(a)), we can observe that the average accuracy is well above 90% even with five simultaneous failures and only 1% of the failure signature. Intuitively, this fact is because the groups of OD-pairs that form the failure signature for each link are large; hence, even a small fraction can create a sample of observations that can uniquely identify the injected failure.

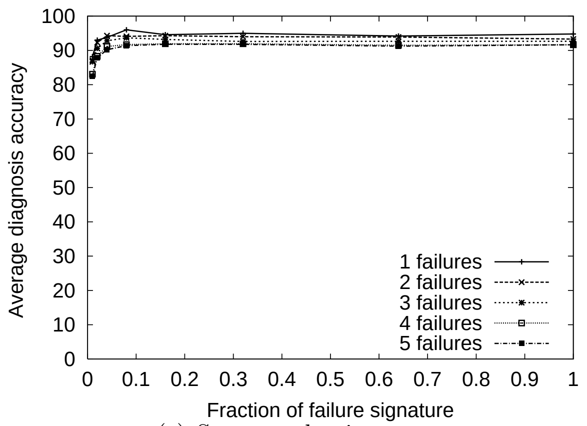
When random noise is introduced into the failure signature (shown in Figure V.4(b)), we can observe that the accuracy is reduced. In particular, lower fractions of the failure signatures are much more susceptible to noise than the higher ones. For example, at $\alpha = 0.01$, the average accuracy is only 60%, while it reaches 90% at $\alpha = 0.16$. This phenomenon is because at smaller fractions of the failure signature, there is a higher chance that the spurious observations can morph



(a) No noise



(b) Random noise events



(c) Structured noise events

Figure V.4: Accuracy of MAX-COVERAGE for different number of simultaneous failures under no noise, random and structured noise scenarios. The y-axis is the average accuracy measured over 500 random failure scenarios.

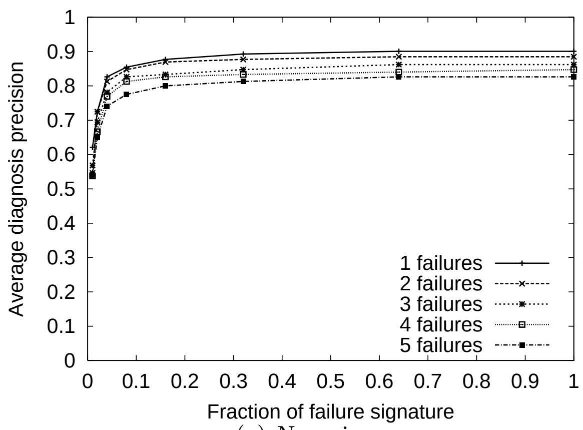
the failure signature of one shared risk into another. Since our algorithm tries to identify risk groups with highest coverage first, it is likely that the original failure signature and the noise will add up to a stronger candidate risk group different from the injected failure.

With structured noise (in Figure V.4(c)), we observe a similar, although less pronounced, phenomenon. The accuracy dips a little compared to the case when there is no noise but is higher than with random noise. The reason is as follows. Since noise is more structured in this case, the resultant failure signature is a composition of α fraction of the original failure signature and $\alpha \times \beta$ fraction of the five noise links, where β is the ratio of the failure durations of the noise and the original failure ($\beta = 5/60$ in our simulations). Since even a small α is enough to achieve high accuracy even for five simultaneous failures with no noise, we achieve high accuracy for the structured noise case.

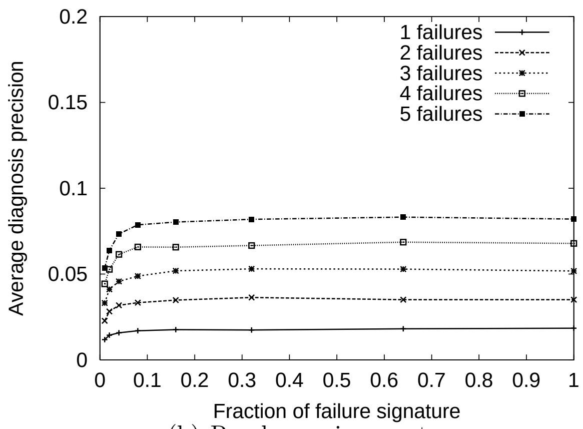
V.B.4 Precision of the localization algorithm

Along with the accuracy, we also evaluated the precision of the localization algorithm—the fraction of truth in the hypothesis—with varying signature fraction α. Without noise in Figure V.5(a), the algorithm enjoys extremely high precision, especially when $\alpha > 0.16$. Precision drops with lower values of α since the failure signature is not strong enough to distinguish between multiple contending risk groups. We also observe that the precision, similar to accuracy, is higher for scenarios with one failure than those with five. Lower accuracy implies that part of the ground truth is not present in the hypothesis, which in turn means that the hypothesis might contain additional candidates not part of the ground truth (i.e., lower precision) to cover all observations, thus leading to lower precision.

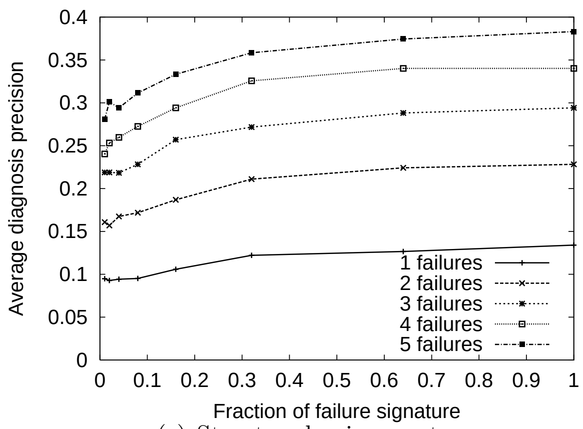
In the presence of noise, we only considered the injected faults as part of the ground truth and not the noise itself. Thus the localization precision is expected to be much lower as the algorithm tries to cover all observations including those caused by noise, which in turn leads to a larger hypothesis. We can observe



(a) No noise



(b) Random noise events



(c) Structured noise events

Figure V.5: Precision of MAX-COVERAGE for different number of simultaneous failures under no noise, random and structured noise scenarios. The y-axis is the average precision measured over 500 random link failure scenarios.

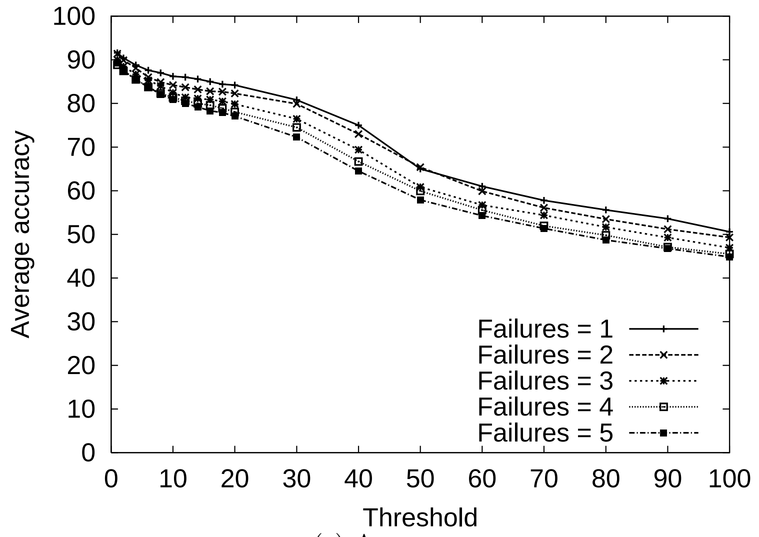
this trend for both the random (in Figure V.5(b)) and structured noise (in Figure V.5(c)) scenarios. For the structured noise, though, the precision is higher than that of random noise; fewer risk groups are required to cover the small fraction of structured noise introduced.

We also observe that the precision is higher for five failures than one in both noisy scenarios, while the opposite is true without noise. The reason for this phenomenon is straightforward: Since the amount of added noise remains constant across the varying number of simultaneous failures, the number of spurious observations and, therefore, the additional risk groups required to cover them remains similar in all cases. The amount of truth, however, increases linearly with the number of simultaneous failures injected, thereby increasing the overall precision.

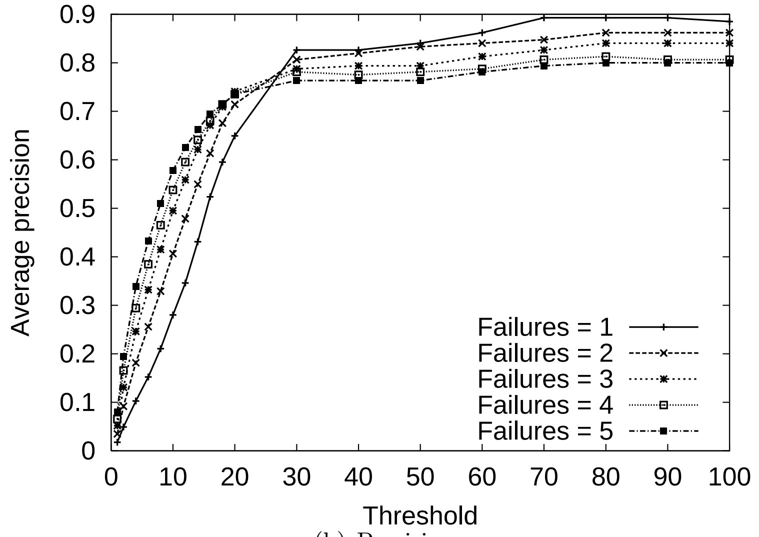
V.B.5 Candidate selection algorithm to improve precision

Fortunately, we can improve the precision without significantly decreasing the accuracy by applying the candidate selection algorithm described in Section **??**. In Figures Figure V.6(a) and Figure V.6(b), we plot the accuracy and precision obtained after applying the candidate selection algorithm for different absolute thresholds. For this experiment, we fixed the fraction of the failure signature to 0.16, which is still very low.

Eliminating candidate links from the hypothesis that were less than the threshold improves precision significantly until a threshold of about 25, after which the decrease in accuracy out-weighs the additional benefit obtained by increasing the threshold. The optimum threshold varies depending on the specifics of the topology and fault detection system, and should be derived empirically for a given deployment.



(a) Accuracy



(b) Precision

Figure V.6: Accuracy and precision of the candidate selection algorithm when we change the threshold. For these graphs, we fixed the fraction of the failure signature to 0.16 and varied the threshold.

Table V.1: Features of the MPLS fault monitoring system.

Feature	MPFM
Topology	Core and outside backbone
Number of routers	100s
Probe distribution	Periodic
Probe frequency	1 per minute
Layer of the probes	MPLS

V.C Experience with real failure data

In addition to the simulations, we also collect failure data from an MPLS fault monitoring (MPFM) system that monitors a section of a real MPLS-switched tier-one ISP backbone network. The MPFM system monitors MPLS tunnels that originate from a subset of edge routers in the backbone, traversing the backbone and finally terminating at other edge routers. Since the MPLS tunnels are established and maintained using the underlying IP topology (through OSPF), any IP-layer failure can impact the MPLS tunnels above the IP layer. The topology consists of a few hundred routers and the probes are transmitted at a periodic rate of one every minute, as shown in Table V.1.

The goal of our evaluation is to stress test our system; we consider every probe loss as part of the failure signature, including those due to noise. Note, however, that this does not bias the accuracy of our tool since we do not include the root causes of the noise (which we do not know anyway) in our ground truth. This however affects precision since the tool attempts to explain all symptoms including those due to noise. In the production version of the tool, we are mainly interested in characterizing large failures, thus noise can be reduced by considering only those OD-pairs with more than a threshold number of dropped probes.

V.C.1 Ground truth

In order to evaluate the efficacy of our fault localization system, we need access to ground truth for the failures. We performed our analysis in two parts—

automated and manual. Automated analysis allows us to study the efficacy of our system over a large number of failures. Unfortunately, for silent failures, there is no automated way to compare our hypothesis with ground truth, since the fault monitoring data does not contain any alarm to correlate with. For such failures, we relied on operator tickets that indicate the root cause of the failure obtained through manual diagnosis. Silent failures are relatively rare in practice, however. So, in order to test our system thoroughly, we relied on various types of non-silent failures for which we extracted ground truth from three data sources—OSPF link-state advertisements (LSAs), syslogs and SNMP data.

The set of *OSPF LSAs* obtained through the OSPF monitor indicates which IP link has been announced/withdrawn from the topology. Certain LSAs also contain OSPF weight changes that can trigger shortest-path computation, causing routing changes in the network. During many routing events in the network, the topology is unstable for a short period and probes can get dropped. For such routing incidents, we compare the hypothesis generated by our algorithm with LSAs corresponding to the routing events.

In the core backbone network, many IP links are in fact bundling of many member interfaces known as composite links [9]. The router typically load balances the packets among these multiple interfaces. Individual member failures within the composite links cause probes that are transmitted on that member to be dropped until the router re-balances the traffic onto other members. Since the composite link is active, such failures do not cause OSPF LSAs, but appear in router syslogs. We, therefore, used syslogs to obtain information regarding these failures. In conditions of high link utilization, such as during failures or during maintenance, links can experience heavy packet loss, and therefore, can cause end-to-end probes to get dropped along these links; such congestion events are found in *SNMP measurements*.

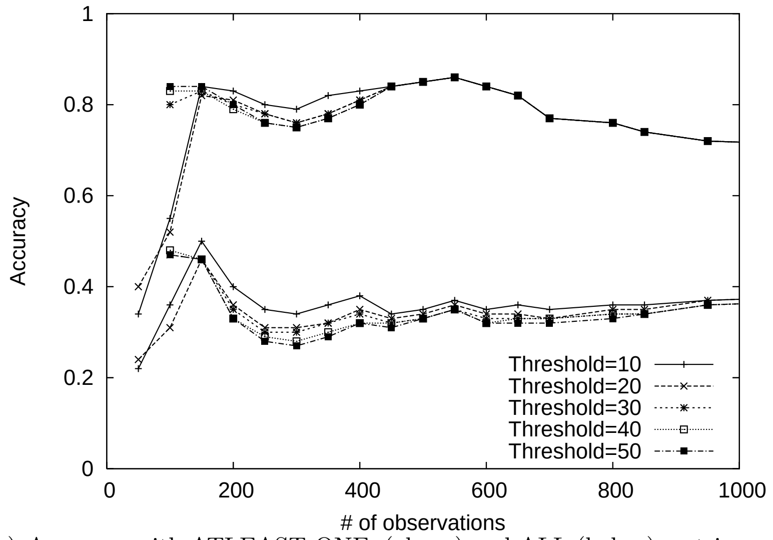
Note that the ground truth obtained through these data sets is only approximate, as there can be instances when a link failure is reported in the ground truth but the event does not impact traffic forwarding. In these cases, the failure signature will not contain any OD-pairs that are impacted by the spurious LSA or syslog message. Thus, the natural comparison with our hypothesis (namely, requiring that the ground truth be wholly contained in the hypothesis) is obviously unfair. As a relaxation from this strict accuracy metric (which we refer to as ALL), we define a more conservative accuracy metric called ATLEAST_ONE in which accuracy is defined to be 1 if at least one of the links in the ground truth is contained in the hypothesis and 0 otherwise. The real accuracy of our system lies between the ALL and ATLEAST_ONE metrics.

Now, we present the evaluation results using the methodology outlined in the previous section. Our results are in four phases: First, we evaluate the different candidate selection followed by the hypothesis selection algorithms. Third, we divide the failures in several categories and compare the performance of our localization algorithm for these different failure scenarios. Finally, we outline our experience in localizing real MPLS black-hole scenarios in the network.

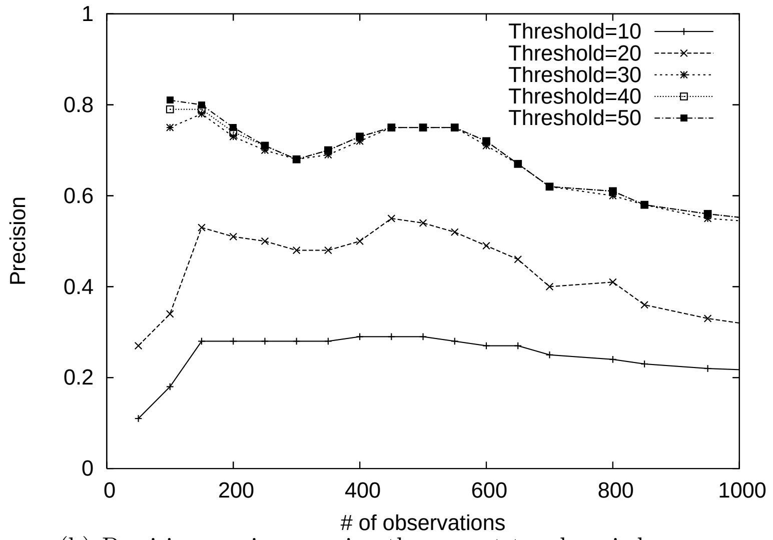
V.C.2 Candidate selection algorithm

In Figures Figure V.7(a) and Figure V.7(b), we plot accuracy using both the ALL and ATLEAST_ONE metrics and precision of localization. For this experiment, we picked the hypothesis with best accuracy among those with different topology snapshots, i.e., ORACLE. On the x-axis, we vary the cardinality of the failure signature (number of observations) from 50 all the way up to 1000 observations in steps of 50. On the y-axis, the average accuracy/precision corresponding to all failure intervals that have at least x observations is shown. In effect, these figures show the trend in the accuracy/precision as the failures impact more and more OD-pairs.

Several conclusions can be drawn. First, the number of failure intervals reduces exponentially from about 600 bins with more than 50 observations to about 20 bins with more than 1000 observations (not shown). This is expected,



(a) Accuracy with ATLEAST_ONE, (above) and ALL (below) metrics, assuming the correct topology is known.



(b) Precision, again assuming the correct topology is known.

Figure V.7: Accuracy and precision for the MAX_COVERAGE algorithm on real failures from a tier-one ISP, assuming the correct topology snapshot is known.

since the number of large failures is typically much smaller than the number of small failures. Overall, we obtained accuracy and precision of about 80% when considering failures with more than 150 observations. Second, the accuracy and precision of localization increase as the failure size increases initially from 50 to 150 observations. However, it decreases slightly after that but is inconclusive as the number of failure intervals is too small to have statistical significance. Larger failure signatures can indicate one of three things, assuming noise in the network remains the same across all failures. First, the fraction of the failure signature captured could be higher, i.e., the failure lasted for a larger duration. Second, the failure might have impacted many OD-pairs in the network, thus the failure occurred on a popular link that lies on many paths. Finally, there could have been many simultaneous failures, the likelihood of which is not insignificant due to router maintenance events. For the first two cases, it is not surprising that our fault localization algorithm performs well, as larger signature fraction means larger accuracy verified using simulations. For the final case, since we use the ATLEAST_ONE metric, there is a strong chance that at least one of the root causes is in our hypothesis. In fact, accuracy using ALL metric is about 40% less than the ATLEAST_ONE metric, both due to the approximate nature of our ground truth as well as the presence of many simultaneous failures in ground truth.

Third, a threshold of 30 that selects candidate links in the hypothesis that cover at least 30 observations seems to represent a good trade off between accuracy and precision. Below this threshold, the precision is significantly lower while accuracy is only slightly higher. Increasing the candidate selection threshold beyond 30 leads to a marginal decrease in the average accuracy, while precision does not improve any further.

V.C.3 Hypothesis selection algorithm

As mentioned earlier, for hypothesis selection algorithm, we use the UNION algorithm which outputs the union of all hypotheses corresponding to

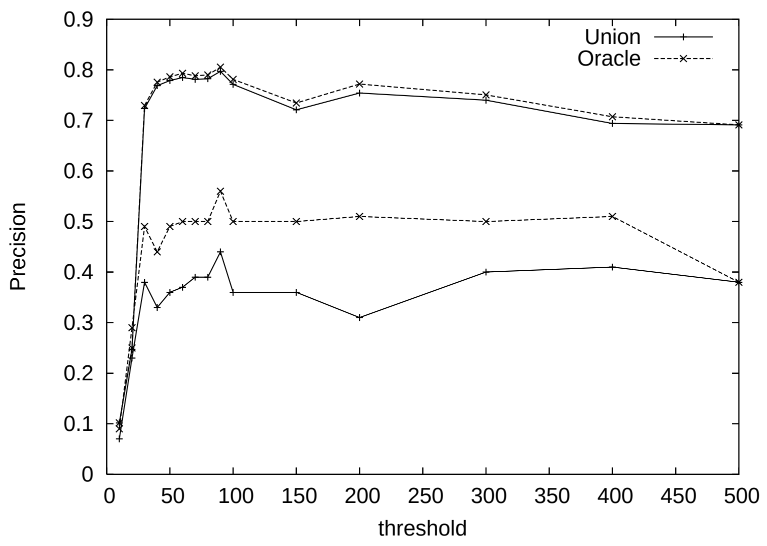


Figure V.8: Precision as a function of threshold on the x-axis, both assuming the correct topology and using the UNION heuristic. The upper two curves include all the failure intervals, while the bottom two curves only include those failures that had more than one topology snapshot in that interval.

various topology snapshots during the failure interval. Because UNION includes the links from all the hypotheses, it cannot cause a decrease in accuracy according to our definition. Therefore, precision is the only metric of interest in evaluating UNION, for which we compare against an oracle that can clairvoyantly pick the best out of all the hypotheses generated using different topology snapshots.

In Figure V.8, we plot the precision for the UNION hypothesis selection algorithm that combines multiple hypotheses obtained using different topology snapshots. The x-axis is the candidate selection threshold that we vary from 10 all the way up to 500. For each of these candidate selection thresholds, we identify all those failure intervals that had at least one candidate link remaining in the hypothesis after we apply the candidate selection thresholds and compute the average accuracy/precision for these failure intervals. The number of bins reduces with increasing candidate selection threshold (not shown in the Figure) due to the fact that we discard bins that do not have any candidates left in the hypothesis after we apply the threshold.

We plot two sets of curves in Figure V.8 corresponding to two different cases. The bottom two curves in the Figure corresponds to the case when we consider only those failure intervals that involved a topology change, while the top curves refer to the case when we consider all failure intervals. Note that if there is no topology change during a failure interval, UNION performs the same as the best possible hypothesis. Since a large majority of cases did not involve a topology change, the reduction in average precision due to the UNION algorithm is negligible. However, if we consider only those cases that involve a topology change, we can observe a dip in the precision due to the UNION algorithm (by about 15%). This means that, for particular scenarios involving multiple topology changes, an operator needs to look at hypotheses that are 15% larger than the best hypothesis clairvoyantly picked.

So far, in our analysis, we have considered all the failures to be of the same type. However, in practice, failures differ from each other depending on the nature of the root causes, especially for different non-black-hole failure scenarios we have considered. Therefore, we partition all the failures into classes and study them individually in the next subsection.

V.C.4 Analysis by failure type

We classify all the failures based on the root cause found in OSPF LSAs into the following types:

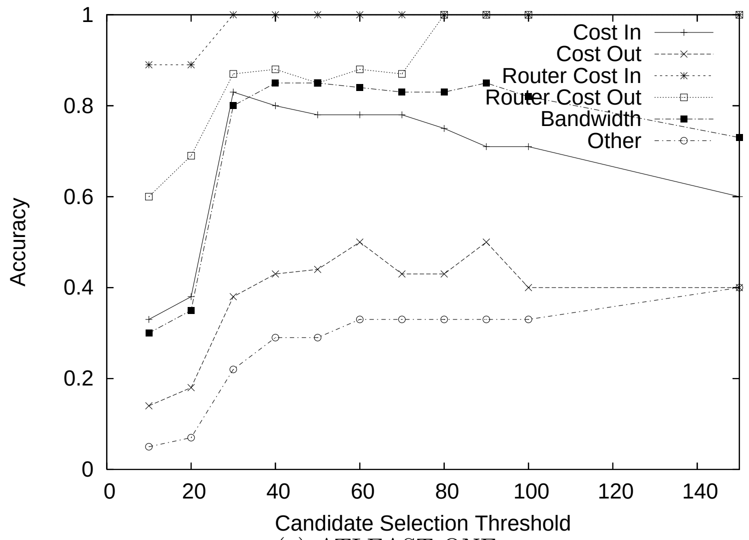
- *Router cost out*: Traffic is removed from all links associated with an entire router by changing the routing protocol weight up to an excessively high value.
- *Router cost in*: Traffic is moved back on to a router.
- *Link cost out*: Traffic is removed from a particular link and not the entire router.
- *Link cost in*: Traffic is moved back on to a given link.

- *Bandwidth events*: A part of the composite links in the network has failed. This failure typically can affect certain probes in the network. These messages are usually reported as bandwidth increased/decreased for the composite links in the syslogs, and hence we call them bandwidth events.
- *Others*: Any other failure that does not fit the above categories is included here.

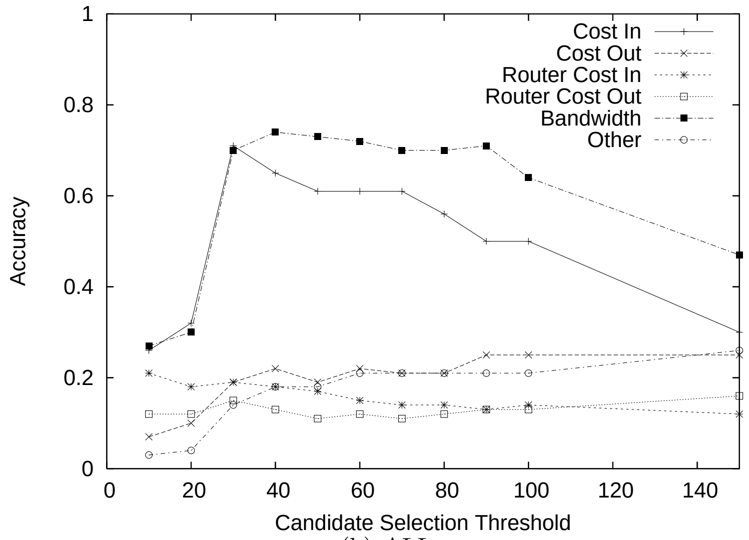
Since manual classification is too tedious, we applied simple heuristics to classify a failure event. For example, if all/most of the LSAs have one end point in common, then it is a router-related incident. If the LSA indicates a change of metric from higher (lower) cost to lower (higher) cost, then it is a cost-in (out) event. If there are only a few links (less than five) experiencing this change of OSPF weight, then we deem it an individual link cost in/out event. Partial composite member link failures are identified through the corresponding notification in the syslogs.

In Figure V.9, we plot the average accuracy on the y-axis while varying the candidate selection threshold on the x-axis for both ATLEAST_ONE and ALL metrics. Recall that increasing the candidate selection threshold automatically considers only failure intervals that have a large number of observations. From Figure V.9(a), we observe that localization was the most accurate for bandwidth-related events for the ATLEAST_ONE metric. This is because the number of simultaneous network events in the ground truth for bandwidth-related failures is small (unlike router cost in where all the links of that router are part of ground truth) and this leads to a more crisper and clearer signature to localize. Router cost out and cost in events ranked next in terms of accuracy according to the ATLEAST_ONE metric.

However, when we compared the ALL metric for different failure types (shown in Figure V.9(b)), the accuracy of link cost in and out events was better than that of the router events. This is due to the fact that during router cost



(a) ATLEAST_ONE



(b) ALL

Figure V.9: Accuracy (using both ATLEAST_ONE and ALL metrics) for different types of failures.

in/out events, a larger number of OD-pairs are impacted, most of which are directly passing through the router. So, the chances of finding at least one match between the ground truth and the hypothesis is far stronger (hence higher accuracy according to ATLEAST_ONE metric), while the fraction of matches between ground truth and the hypothesis is considerably lower for these events. Note that the lack of clear trend as we increase the threshold beyond 80 is due to the small number of failure intervals (around 20-25).

V.C.5 Real MPLS black holes

We describe three silent failures we analyzed using our system using data obtained from the MPFM system. We consulted operator logs to obtain the ground truth for these real black hole scenarios. While these black holes have already been fixed in the network, we applied our system on archived failure data to evaluate whether our system is indeed effective to localize real black hole scenarios.

In the first incident, misbehavior of a new device that was connected to the periphery of the network caused many routes to go through the device which were then subsequently black holed. This is a perfect example where we need to consider all the topology changes within a failure interval. In this case, our localization system output two candidate links as the hypothesis—the (properly functioning) link before and the (black hole) link after the re-routing of traffic. For this incident, the localization accuracy therefore is 100% while precision is only 50%.

In another failure scenario, the forwarding component of a line card failed to dequeue packets until the card was reset. Our localization system output a hypothesis that had five candidate links, out of which, when we applied our threshold of 30 eliminated the four false positives out of the hypothesis and contained only the actual failed link. This hypothesis therefore has 100% accuracy and precision.

Another known black-hole scenario happened due to a misconfiguration causing brief loss in connectivity to MPLS paths that traversed that link. Our localization algorithm output a hypothesis that contained four candidate links, two of which were eliminated after we applied our candidate selection algorithm. Out of the remaining two, one was the actual black hole while the other was a false positive. However, the false positive could not be easily distinguished from the actual black hole since both these links appeared on all the paths corresponding to the impacted OD-pairs.

V.D Related work

Though there is a tremendous amount of literature in the area of network fault management and monitoring, there has been little discussion of silent failures or black holes of the types considered here. Systems and general techniques for network data correlation are widely used, and are under continuous refinement as new statistical methodologies for fault and anomaly detection are developed [70, 17, 58, 68, 23, 36].

Inference problems generally are of wide interest in the operational and networking research communities. While we know of no other work that targets the silent failure detection problem we consider here, there is considerable research in use of partial or incomplete data to reconstruct unknown network internal and external topology, traffic and performance. We take a simple, greedy approach to inference, which we find works well. More complex approaches might also be of interest. In particular, there is the rich area of network tomography, an approach to (massively under-constrained) linear inference, which has been applied to inferring topology and link performance from end-to-end measurements, as well as to inferring OD-traffic demands from link traffic measurements [15, 35, 63, 67, 93, 94, 101, 102].

Our detection system uses standard mechanisms in route and topology monitoring and packet probing to identify reliability metrics such as packet loss, delay, etc., on a per-path basis. Such measurement is routinely performed by many

ISP backbone operators using a variety of different tools such as ZING [61] and BADABING [83].

V.E Summary

In this chapter, we developed and evaluated a simple yet effective methodology for the localization of black holes or silent failures in the network. One of our key contributions is the successful application of risk modeling to localize failures even in the presence of noisy data. Using real failure data obtained from a tier-one network's MPLS fault monitoring system, we demonstrated that our system can effectively aid network operators in troubleshooting failures.

The reason for applying risk-modeling methodology to localize black holes is the lack of direct mechanisms to isolate the root cause. More specifically, active probing approaches were necessary to detect black holes because routers currently do not self-detect such problems automatically. Hence, ISPs are forced to use such indirect approaches.

So far, we have focused on applying the risk-modeling methodology to fault-localization problems commonly found in backbone networks. In the next two chapters, we identify two key problems in these individual problem domains and propose clean-slate architectural approaches to solve them.

First, in the IP fault localization domain, we observed that one of the fundamental problems is maintaining accurate associations across IP and optical domains. While SCORE algorithm deals with such errors using the error threshold, still, a systematic architectural approach is necessary to ensure such associations are accurately maintained without having to wait for a fault to occur to realize associations were incorrect. Thus, in the next chapter, we discuss clean-slate approaches for accurate cross-layer visibility.

Second, we propose a general measurement architecture called m-Plane in Chapter VII that both scales well with network size as well as, perhaps more importantly, facilitates direct measurement of fault localization. m-Plane architecture is based on novel router primitives that allow individual router- and link-level measurements composed together to form end-to-end path metrics. We also show how m-Plane can be incrementally deployed in the network, and the associated benefits as we increase the number of upgraded routers.

V.F Acknowledgments

This chapter is based on the paper titled "Detection and Localization of Network Black holes", appeared in the *Proceedings of the Infocom* , held at Anchorage in May 2007, which is joint work with Jennifer Yates (at AT&T Labs – Research), Albert Greenberg (at Microsoft Research), and Alex C. Snoeren. The dissertation author was the primary investigator and author of this paper [55].

Chapter VI

Cross-layer visibility as a Service

Layering in IP networks fundamentally hides the complexity of lower (upper) layers (e.g., the underlying optical network topology) and exposes very simple interfaces to upper (lower) layers (e.g., a logical link), thus allowing parallel and independent evolution of the layers while preserving the interface between them. For example, an IP router does not need to know exactly all the optical components that constitute its IP link to an adjacent router. Thus, routing protocols such as OSPF or IS-IS can only focus on the link-level abstraction without any knowledge of what optical components the link comprises of or where these components are located within the network. While this logical separation helps contain the complexity beneath simple interfaces, we argue that strict layering results in poor *cross-layer visibility*, negatively impacting many management functions including fault localization that rely on accurate cross-layer associations.

In this chapter, we discuss two approaches for obtaining accurate cross-layer associations: In the first approach, we consider "fattening" the interfaces so that every layer automatically determines as well as disseminates dependencies throughout the network as shown in Figure VI.1(a). In the second, we do not perturb the layer boundaries we have today, instead provide cross-layer visibility to various network management functions by joining different databases as shown in Figure VI.1(b). We discuss the pros and cons of these two approaches and

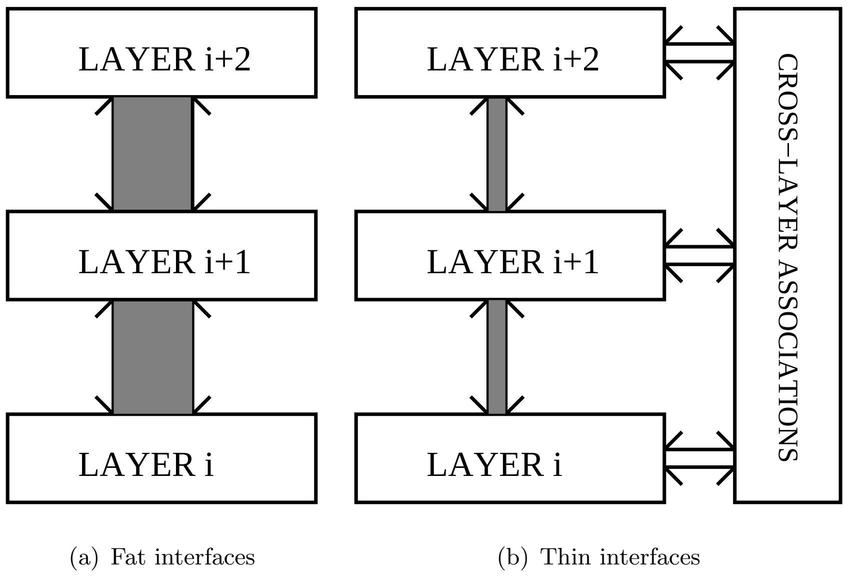


Figure VI.1: Two ways to provide cross-layer visibility

argue that an effective approach to cross-layer visibility should have room for many independent mechanisms that allow discovering associations with increasing levels of complexity as well as accuracy. Further, we argue that the architecture should provide cross-layer visibility as a *service* to all the network management functions that rely on accurate associations across layers.

The rest of this chapter is organized as follows. First, we outline some of the important network management tasks that rely on accurate cross-layer associations in Section VI.A. In Section VI.B, we discuss why it is hard to provide accurate cross-layer associations. We then argue that fattening the interfaces to provide accurate cross-layer visibility is not desirable in Section VI.C. In Section VI.D, we present an architecture that provides an evolutionary path that gradually increases the level of accuracy in these associations while containing complexity within layers.

VI.A Importance of cross-layer visibility

In today's IP backbone networks, each IP link consists of a connected set of optical components organized in different topologies (e.g., ring, mesh, etc.).

A single link consists of many different optical components and many different links can share a particular component, thus creating a many-to-one, one-to-many mapping. Cross-layer visibility refers to the associations between higher-layer abstractions and lower-layer components and vice versa. For example, in IP networks it refers to the association between an IP point-to-point link to the set of optical components that comprise the link. Accurate associations are critical to the functioning of various operational tasks—some of which have been described below.

Backbone planning. Backbone planning involves engineering the network to withstand a wide range of potential failure scenarios including possible attack scenarios and planned traffic growth, as well as to support additional services and features in the network. An accurate audit of the network that transcends all layers, therefore, is a key ingredient in backbone planning. Similarly, IP paths are typically selected to avoid any single points of failures or SRLGs [90]. Accurate IP-to-optical associations in databases are required to choose physically diverse paths to carry traffic to withstand failures in the lower layers. Erroneous IP-to-optical associations in databases can result in engineering the network to incorrectly choose non-diverse paths to carry traffic; a single failure in turn can partition the network.

Customer fault tolerance. Customers (e.g., e-commerce businesses) are primarily interested in obtaining uninterrupted network connectivity either from one single service provider or through different service providers via multi-homing. One common question they often face is about the level of diversity in their connectivity to the backbone. Even when they connect to different points-of-presence (PoPs) within the same provider, or to two different carriers (e.g., Sprint and AT&T) there could be shared risks lurking (e.g., fibers passing through the same tunnel) that could be of concern to the customer. Whether to disclose such information about physical connectivity completely or in part is often a policy decision; nevertheless, accurate cross-layer mappings are important in order to answer these questions.

Alarm suppression and diagnosis. As discussed in Chapter IV, alarms are generated from various network elements (e.g., optical equipment, routers etc.), sometimes at different layers to indicate IP link failures. For example, a fiber cut can cause the router to raise alerts indicating that the interface is down at SONET, PPP, IP, and MPLS layers in addition to the loss of signal (LoS) alarms raised by certain optical components. If multiple links are affected due to shared risks, all of the links, and potentially their associated optical components, raise alarms at all impacted layers overwhelming the network operator. Accurate associations are required to group these alarms together into a single event. Further, the accuracy of diagnosis (either manually or using our fault-localization system described in Chapter IV) is limited by the consistency of the IP-to-optical database. Accurate associations are also critical in proactive root-cause analysis of other performance related problems such as chronic intermittent flapping of interfaces, link degradation, etc., that potentially may have not (yet) triggered alarms.

Maintenance. Network operators often gracefully remove the traffic on a link (by increasing the OSPF weight of a link or some such mechanism) before performing maintenance (e.g., repairing a faulty component, provisioning a new link, software upgrades, etc). Mis-associations across layers can cause operations to induce unwanted faults into the network. For example, if the IP-to-optical associations were wrong, operators intending to perform maintenance on a link between Los Angeles and San Francisco might instead inadvertently impact traffic flowing between San Diego and Los Angeles.

VI.B Why is it hard?

It might appear to the reader that accurately maintaining such associations should be a straightforward task. After all, the network operators provision the network in a centralized manner; therefore, they can log these associations in databases. However, as we have mentioned before in Chapter IV, a live operational network incurs significant churn as links are provisioned, old equipment is replaced with new equipment, faulty components are repaired, interfaces are re-homed and so on. Database errors can result from this inherent churn—for example, if operations fails to update the relevant databases as an IP link is moved from a failed line card (slot) to a different, operational card (slot).

Additionally, this task is complicated by the presence of restoration at individual layers. For example, a failure within a SONET ring is recovered by rapidly protection switching to re-route the traffic the other way around the ring. In more "intelligent" optical networks, optical-layer restoration causes the path to re-route from the primary to an alternate path. These dynamic path changes at lower layers are typically achieved without impacting the upper layer connectivity; IP links are, by design, oblivious to restoration at lower layers. Of course, one can argue that restoration in lower layers reduces the need or in some cases obviates the need for cross-layer visibility. While this is partially true, cross-layer visibility is still important because:

- IP layer might experience subtle changes in other performance metrics such as end-to-end delay;
- operations personnel need to ensure that restoration itself does not have any problems;
- it is cheaper with the current technology to provide IP-level restoration than optical; thus optical layer protection is often not used—particularly on high speed links [57].

This flux in topology can make it harder to diagnose failures or other performance issues without the presence of accurate cross-layer associations. One can, therefore, conceive that the network ought to be engineered to provide such information, perhaps by widening the interface between layers (e.g., exposing changes in optical topology to IP links), in the context of network management. While this conceptually clean design exposes such associations as a part of the network, we argue that this is neither *practical* nor *desirable* in the next section.

VI.C Fattening layers is not a good idea

A fat interface between layers allows information to flow from one layer to another layer as a part of the architecture itself. For example, if the network layer (IP/MPLS) were made aware of the underlying components in optical topology, this could allow the network layer to make better choices in recovering from failure situations. Indeed, in the context of fast restoration from failures in the MPLS domain, Interior Gateway Protocol (IGP) extensions in [47, 82] incorporate shared risk link groups (SRLGs) in their link state advertisements (LSAs). These SRLGs themselves could be auto-discovered through other means (such as through optical topology information obtained through link-management protocols such as LMP [56]). This availability of SRLGs allows the computation of backup paths that are physically diverse from the primary paths. While such an approach has the clear advantage that cross-layer associations can be directly and accurately obtained from the network, we argue that this approach does not scale well. Some of the reasons are listed below.

- *Complexity.* Exposing lower-layer topology to upper layers adds complexity into the network (increased processing due to new types of messages) and limits scalability (too many devices results in higher messaging overhead). In doing so, the routers are unnecessarily burdened with flooding messages from not only its own layer but also from layers below. This additional burden, while potentially helpful for the purposes of better management of the network, stunts the evolution of the network. Every optical device needs to export an interface that provides visibility into its layer to the layers above. Suddenly, lower-layer devices need to be aware of the upper layers making them more complex than they need to be.

- *Interoperability.* Interoperability does not scale well with number of different types of devices; the larger the number of devices that need to be interoperable, the more difficult it becomes to achieve consensus on one protocol. Besides, it necessitates long design and testing cycles across large number of devices and manufacturers.

- *Security.* In some cases, this additional visibility into lower layers is not desirable due to security reasons. For example, consider the case of physical-layer virtual private networks (VPNs), where a customer directly obtains a circuit from a provider. The service provider only manages the raw optical circuit and has no visibility into the network layer at all. Forcing wider interfaces would expose more information to the service provider that might not be acceptable to the customer. In other cases, allowing individual network elements to be queried can also make the infrastructure more vulnerable than it has to be.

- *Incompleteness.* Even if one were to imagine complexity, interoperability and security were not of concern, fundamentally, it is difficult to achieve complete cross-layer visibility. For example, it is difficult to automatically identify whether two physical pieces of fiber are traversing the same conduit, or if two conduits traverse the same tunnel, etc. Obtaining the exact location of each of the fibers and other geographical information such as proximity to faults, volcanic regions etc., is an extremely tedious task. Also, the definition of what constitutes diversity (e.g., how geographically far apart should physically diverse fibers be) is a matter of policy and can often be hard to define [90].

In practice, therefore, many ISPs today provide cross-layer visibility to various network management functions by maintaining complex databases and analyzing large amounts of topology, configuration, and measurement data collected from network elements at each layer. Still, this ad hoc approach of collecting and analyzing data in home-grown databases is not a sufficient solution, either. Instead, we argue that the management layer should provide cross-layer visibility as a service, with well-defined interfaces for populating the external databases and querying the information in the next section.

VI.D Cross-layer visibility as a service

Rather than dictating what the network elements store and export—the approach taken by the Simple Network Management Protocol (SNMP) [16]—we focus on what information is *imported* into the management database. This subtle distinction is extremely important, as it allows many different solutions for providing the information. Although the network elements themselves could generate the data (as in SNMP), the information could also come from separate measurement devices or even human operators. This approach accommodates the inherent diversity across the layers and the natural evolution of techniques for collecting the data. We also present a possible evolution path for three layers—determining the IP forwarding path, mapping an IP link to optical components, and identifying fibers running through the same geographic location. These examples could easily be extended to include other protocol layers, such as paths through an overlay network or a sequence of tunnels or MPLS label-switched paths.

Greater uniformity in the data representation would make it easier to evolve a network, integrate two networks after an acquisition, and employ third-party network-management tools. More broadly, we argue that the management system should have interfaces for different stake holders—such as network designers, network managers, and customers—to query the data, with explicit policies governing the kinds of information each party can access. For example, a customer could ask if two IP paths (or two access links) are physically diverse but might not be told that the fibers run through the same tunnel. In contrast, a network manager troubleshooting a reachability problem could perform a complete *traceroute* of an IP path across all of the layers. A network designer could conduct a "what-if" analysis of the effects of planned maintenance on the link loads. The system can also keep a log of past queries, to learn more about the cause and impact of failures by analyzing patterns in the queries. Maintaining explicit cross-layer visibility information presents a number of interesting research and operations issues which we discuss in Section VI.E.

VI.D.1 Architecture

As some ISPs already perhaps employ today, we advocate that each AS have a possibly distributed management database that stores the topology at each layer and how a link at one layer maps into a set of components at the layer below. For example, the database would store the IP topology (i.e., the routers and the links between them) as well as the forwarding paths between each pair of routers. The database would also store the optical topology and which sequence of optical components, such as fibers and amplifiers that form the link between two adjacent IP routers. Similarly, the database would keep track of which fibers run through the same conduit, as well as the geographic path the conduit traverses from one termination point to another. The database should have unique names for devices at each layer, as well as indices necessary to map between layers.

In addition, cross-layer visibility should be provided as a service to a variety of clients. Today, traceroute is the primary way a customer determines the path its traffic takes through the network. Yet, traceroute is problematic for several reasons: (i) ISPs often disable or rate-limit ICMP to avoid overloading their routers, or to hide their topology information, (ii) the probes do not see the network elements at lower layers (e.g., inside an MPLS label-switched path, or the optical components between two routers), and (iii) analyzing changes in the path requires frequent probes to capture both the old and new paths. Instead, our management system could provide a "cross-layer traceroute" service, without customers probing the network directly. Similarly, the management system could support queries for network designers to identify shared risks and model the effects of failures on the flow of traffic through the network.

Providing cross-layer visibility as an off-line service has several advantages:

- *Lower overhead on the routers:* Queries are answered by the management system, rather than the routers themselves. The system can also cache the results of recent or common queries to reduce the overhead of satisfying future queries.
- *Answering historical questions:* By maintaining a log of network changes over time, the service can answer queries that require historical data. For example, a customer could inquire about a performance problem that started ten minutes ago, and the service could report whether a failure forced the customer's traffic onto a path with a longer round-trip time.
- *Application of security policies:* The management system can apply explicit policies to control what kind of information is revealed, and to whom. For example, a customer may be allowed to ask if two paths have a shared risk, but not learn exactly what component is shared and where it is located. In addition, by forcing all queries through the service, the AS can protect its routers from probe traffic while still providing good network visibility to customers.
- *Flexible policies for defining shared risks:* The notion of a shared risk is extremely subjective [90], and the service can accommodate different notions by allowing queries at different granularities and incorporate extra information. For example, a network designer may want to know if two fibers lie near the San Andreas fault in San Francisco. Or, one customer might be interested in link-disjoint paths and another in PoP-disjoint paths through the network.
- *Cooperation between ASes:* ASes could cooperate to provide greater visibility into shared resources. For example, an ISP that leases fiber from another

provider could automatically learn the geographic path it follows (abstracted as deemed fit by the providers), or a multi-homed customer could determine its vulnerability to failures affecting both of its providers. Or, a governmental agency could conduct a realistic study of the effects of a serious catastrophe (such as a terrorist attack) on the Internet infrastructure.

With standard representations of the topology and paths at each layer, and the dependencies between layers, ASes can provide these kinds of valuable services.

VI.D.2 Independent evolution of each layer

By defining the data *imported* by the management system, rather than *exported* by the network elements, our architecture supports many ways of learning the intra-layer topology and paths, and the cross-layer mappings:

IP topology and forwarding paths: The IP-level topology for an AS consists of routers and links, and a forwarding path consists of one or more sequences of IP links. The topology and paths can be learned in various ways, with different degrees of accuracy and timeliness:

- *Static view:* The topology can be recorded by the operators as equipment is installed, or reverse-engineered from the router configuration state. The IP forwarding paths can be computed by modeling which paths the routers, as configured, would select. However, these static views do not capture which routers and links are unavailable at a given time.
- *Periodic snapshot:* A monitoring system can poll the routers for their status and forwarding tables, or run traceroute probes to map the topology. The forwarding paths can be computed on the measured topology, identified from the forwarding tables, or extracted directly from the traceroute results.
- *Continuous view:* A monitor could collect routing-protocol messages, field alarms when equipment goes up/down, or analyze syslog output generated

by the routers to provide an up-to-date view of the topology and paths. If the AS supports explicit routing (e.g., using MPLS label-switched paths), the management plane would know the forwarding paths because it would be responsible for configuring them.

With our architecture, an AS can easily evolve its network design and monitoring infrastructure while maintaining the same representation of the topology and paths in the external database and management applications.

Optical components and paths: The optical topology consists of a diverse array of devices, including fibers, amplifiers, cross connects, and add-drop multiplexers. The sequence of optical components underlying an IP link could be learned in various ways, depending on the sophistication of the optical components:

- *Completely manual:* The operators can keep track of optical components and their relationship to IP links as the equipment is installed. To reduce the likelihood of inaccuracies in the database, the AS can apply basic consistency checks, such as verifying that two ends of an IP link map to the same circuit identifier. As a second line of defense against errors, the AS can monitor the effects of optical failures on the IP layer to identify and apply correlation algorithms to identify incorrect mapping information [45, 54].

- *Partially automated:* Manually constructing the list of optical devices underlying a link is not sufficient if any of the underlying components adapt automatically to failures. For example, an intelligent optical cross-connect may reroute the traffic through an intermediate cross-connect when a component along the direct path has failed. Similarly, a SONET ring may adapt by redirecting traffic around the ring in the opposite direction. Capturing these changes requires logging of alarms or periodic probing of the adaptive components and correlation across layers. Although automatic restoration protects the IP layer from optical failures, knowing the new mapping is important for troubleshooting performance problems (e.g., a sudden increase in round-trip times) and identify new shared risks. As an added benefit, these automatic routing changes at the optical level also provide opportunities to identify mistakes in the human-entered databases, while reducing the effects of the failure from the IP layer. Still, some parts of the database may remain human-generated, such as the identity of the ingress and egress cross connects, or the list of optical amplifiers between any two cross-connects.

- *Completely automated:* Discovering the optical components becomes much easier if the network elements have a common control plane, such as Generalized MPLS (GMPLS) [62]. For example, GMPLS includes LMP [56] that performs neighbor discovery between adjacent network elements so they can dynamically establish a light path from one router to another. LMP provides the names and attributes of the optical components, obviating the need for human-generated databases to map between the IP and optical levels.

In our architecture, an AS can gradually deploy more intelligent optical devices and new auto-discovery protocols, while maintaining the same representation of the path through the optical layer between two routers.

Fiber and fiber spans: A fiber map captures the topology of the underlying transport network. A fiber consists of multiple *spans*, a segment of fiber traversing a single conduit; a fiber span, in turn, consists of multiple fibers traversing the same conduit. This information could be learned in various ways:

- *Completely manual:* As with other optical components, the operators can keep track of the location of fiber and the mapping to/from spans as the fibers are installed, or leased from other providers. The failure of fiber spans (e.g., due to a physical cuts), as they occur, provide an opportunity to identify incorrect mappings. Measurements of propagation delay across a link (and comparison with the supposed fiber path) is another way to detect serious inconsistencies.

- *Intelligent conduits:* Since fibers are passive devices, they do not automatically advertise their operational status (e.g., loss of signal), presence in a particular conduit, or the physical paths they traverse. Creating new techniques for auditing the management database, or even automatically generating the data, is an exciting direction for future research. We envision several possible approaches, including:

  - *Active devices at conduit end-points:* Optical amplifiers along the optical path could report their identity and geographic location [78]. In addition, the individual fibers could have RFID tags where they enter and leave and conduit.

  - *Active devices along the conduit:* For even higher accuracy, the conduits could have active devices, such as audio or wireless transmitters, placed at fixed intervals. These devices could be coupled with GPS receivers (to allow the devices to broadcast their geographic locations), or a separate measurement system could analyze the signal strength to aid in locating the devices. Closer spacing of these devices would provide more fine-grain data, at the expense of higher cost.

  - *Multi-layer packet monitoring:* To verify the mapping of fibers to IP links, we could envision a new generation of packet monitors that combine IP packet capture, reading of audio or RFID tags, and reporting of geographic positioning information. For example, a packet monitor could be used to tap a fiber and analyze the IP packet stream, perhaps on a per-wavelength basis. By capturing the routing protocol messages (e.g., OSPF HELLO messages or link-state advertisements), the monitor can determine the IP addresses of the routers on either end of the associated IP link. Over a period of time, the packet monitor could be installed at various points in the network to collect accurate mappings of IP links to/from fibers (and fiber spans) to check and update the information in the database.

In our architecture, the management database would store the mappings of fibers to spans, as well as the geographic path of the spans (at some known level of accuracy), however they are determined.

VI.E Summary

Over the years, networks have naturally evolved into layers, facilitating parallel and independent evolution within the confines of these layers. Network management, on the other hand, requires accurate vertical cross-layer view of the network for various operational tasks such as backbone planning, fault diagnosis and maintenance. This chapter addresses the challenges of providing cross-layer visibility to network-management applications, and advocates against expanding the interfaces between layers for auto-discovery of the cross-layer associations. Instead, we propose an architecture where such associations can be learned or maintained automatically, not by widening the layers, but by defining the data that should be imported into a management database. The architecture provides cross-layer visibility as a service to other applications and users that depend on this information. In the next chapter, we propose new router primitives for a clean-slate measurement architecture that allows scalable composition of path metrics and direct fault localization.

VI.F Acknowledgments

This chapter is based on the paper titled “Cross-layer visibility as a service,” that appeared in the *Proceedings of the Fourth ACM Workshop on Hot Topics in Networks*, held at College Park, MD, in November 2005, co-authored with Jennifer Rexford (at Princeton University), Jennifer Yates (at AT&T Labs – Research), Albert Greenberg (at Microsoft Research), and Alex C. Snoeren. The dissertation author was the primary investigator and author of this paper [51].

Chapter VII

Scalable Measurement Architecture

Our application of risk-modeling methodology to black hole localization is motivated by the fact that the network elements do not detect such failure scenarios and react to them automatically. While vendors constantly develop work-arounds for specific protocols, experience suggests that there are always new protocols or new enhancements to old protocols that can potentially cause forwarding problems such as black holes. Of course, similar to our black-hole localization system, we can instrument active probes at the desired protocol layer and combine the detection mechanism with risk-models to localize the root cause of problems.

Such an approach, however, is not scalable due to the following reasons. First, active probes between every pair of end points scales as $O(n^2)$, leading to a significant overhead for large values of n (e.g., per-VPN monitoring). Second, many ISPs report that configuring and managing measurement servers at end points itself is challenging. Third, indirect approaches such as ours can lead to inaccuracies due to the inherent inefficiencies in monitoring using active probes and/or risk-model creation, as we have observed in the black-hole detection problem. Of course, we can devise domain-specific heuristics that can improve the accuracy for the common-case failure scenarios, but it can be difficult to always devise such heuristics for the worst-case scenarios.

Motivated by these observations, in this chapter, we develop *protocol-agnostic router primitives* that allow direct isolation of the location of the failure with significantly lesser probe overhead in comparison to active probing. These primitives are based on specialized hardware assistance in the router to monitor various forwarding paths within the router as well as links to other external routers. Every router in the network reports these individual router- and link-level measurements to a centralized monitoring station. Depending on the granularity of these measurements, the monitoring station can then compose end-to-end path properties beginning with basic metrics such as connectivity, loss, delay to more complicated metrics such as available bandwidth, jitter, etc.

Our composition-based measurement architecture, m-Plane, enables service providers to monitor their network both efficiently as well as accurately. Efficiency of m-Plane stems from $O(m)$ scaling of measurement cost, where m is the number of links in the network, assuming that the cost of monitoring any given forwarding path within a router is negligible. Accuracy in localizing the root cause, on the other hand, is achieved by the direct monitoring of individual segments. In addition, m-Plane also eliminates the need for additional measurement servers in the network, since the routers themselves handle measurement and monitoring functionality.

In theory, if every router in the Internet were equipped with these primitives, one can compose true end-to-end path properties from individual measurements. However, inter-AS cooperation is often challenging to accomplish as ASes might not be willing to share the exact location of, say, a congested link along an end-to-end path. Therefore, more realistically, we believe that m-Plane is better suited to monitoring paths from one border router to another within a given domain, which service providers already perform today using active probes. While we are fully aware of the need to monitor many metrics of interest (e.g., delay, loss), we focus mainly on scalable connectivity monitoring in this chapter, which

Table VII.1: Number of probes issued and the actual number of measurements required for different ISP topologies.

AS Name	# of backbone routers	# of probes between customer routers	# of minimum measurements
ASN-TELSTRA	139	3,199,185	22,632
Sprint	573	22,872,466	264,362
E-Bone	117	15,753	2,769
NTTC-GIN-AS	787	7,536,903	120,019
TISCALI-BACKBONE	170	28,920	4,119
Level3	454	679,195	234,371
AT&T	530	39,600,550	129,719

itself is non-trivial as we shall observe.

Note that our router primitives are not meant to completely replace the need for end-to-end probing which will probably always be done at some limited frequency for customers to assure themselves of end-to-end performance especially across multiple networks. However, the router primitives we propose can greatly facilitate very quick (say milliseconds) and direct isolation of network health problems within a single administrative domain. The rest of the chapter is organized as follows: First, we characterize the amount of probing performed today using active measurements in Section VII.A. Second, we present the m-Plane architecture discussing both clean-slate as well as incremental deployment scenarios, and other implementation issues in Section VII.B. We quantify the benefit obtained by deploying m-Planein Section VII.C.

VII.A Scaling active measurement

Customer VPNs typically originate as well as terminate at customer-edge (CE) routers. Typically, a tier-one ISP has thousands to hundreds of thousands of customers. Given the large number of CEs and $O(n^2)$ scaling properties of active probing, it is no surprise that ISPs today restrict probe end points to the provider edges (PEs), thus keeping n to smaller values. However, there is a growing need for VPN-specific performance guarantees [69] using service-level agreements (SLAs), in addition to monitoring liveness to detect black holes and other forwarding problems. This fact is especially true for customers who migrate from proprietary Frame Relay networks (with a Committed Information Rate (CIR)) to VPN service. To better understand the challenges involved in such measurements, we attempt to precisely quantify the magnitude of the scaling problem using Rocketfuel [84] topologies for a set of real ISPs.

Unfortunately, the Rocketfuel topologies for major ISPs consist mainly of backbone routers, with a small number of customer routers that the authors could map in [84]. Further, the maps are old and incomplete; the set of customer edges that are represented in the topologies are only a small fraction of today's reality. In particular, the largest Rocketfuel topologies have less than 10,000 customer interfaces. However, most tier-one backbone providers support on the order of 100,000 customer interfaces—an order of magnitude larger than reported in the paper. Hence, our analysis is likely to under-predict the true cost of full customer-to-customer measurement in today's ISPs.

Table VII.1 compares the total number of end-to-end probes required between all customer routers with the minimum number of actual measurements required, in terms of the number of links and total number of per-router ingress-egress pairs in the topology graph. Intuitively, if we can compute end-to-end metrics by composing link and router metrics, then the overhead scales with the number of links and interface pairs. We can see that there is an order-of-magnitude difference between these two values, suggesting that there is considerable scope for improvement, plausibly using new mechanisms. In the particular cases of Sprint and AT&T, there is almost a two order-of-magnitude difference between the total number of paths to be measured and the inherent number of measurements required in terms of number of links and total number of router interconnections. Note that we have counted each active path probe only once, when in reality a probe traverses several routers' links; accounting for path lengths would increase the differential even more. Note also that if we conduct measurements at the routers, we may be able to completely dispense with the external measurement endpoints. For example, Table VII.1 suggests that a complete approach to customer VPN SLAs for Sprint would require around 5,000 measurement end points.

One key limitation with our analysis is that we assume a complete mesh between every pair of CEs. Customer sites are often grouped into VPNs, however, and are far more interested in performance metrics within their VPN than outside of it. Additionally, within a customer VPN, the topology may not be a full mesh but instead may be a hub and spoke model [71]. For a hub and spoke model, the number of probes is $O(m \times n)$ (as opposed to $O(n^2)$), when m spokes communicate with n hubs. Despite these issues, we believe Table VII.1 provides a first cut analysis of the opportunity for improving on end-to-end probes. In particular, if we conservatively consider active probes between every pair of backbone routers—as opposed to the customer interfaces—we still find a 35-fold improvement (not shown in the table) for most topologies. Hence, using the number of backbone routers provides a lower bound, the results in Table VII.1 provide an upper bound, and the operational reality lies in between.

VII.B m-Plane Architecture

The previous section shows a large difference between the overhead of active probing (as it is currently done) and the inherent complexity (in terms of the forwarding paths in routers and number of links). Clearly, if we wish to go closer to the lower bound, we need a *compositional approach*, where routers measure metrics on links and nodes, and a centralized monitoring station obtains these link and node metrics from routers and composes them to obtain end-to-end properties. Note that while we limit the scope of this chapter to connectivity, whenever possible we use the terms "metrics" to emphasize the potential to generalize to other metrics

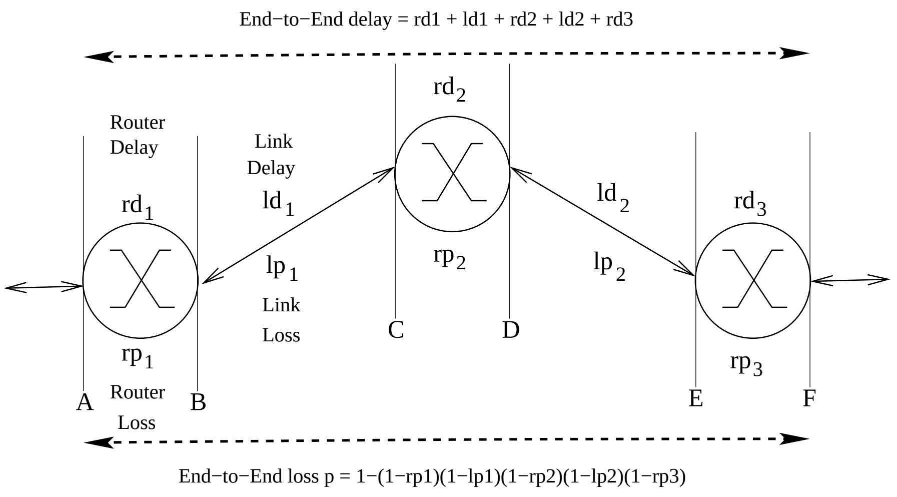

End−to−End delay = rd1 + ld1 + rd2 + ld2 + rd3
Router Delay
rd1
Link Delay
ld1
rd2
ld2
rd3
lp1
Link Loss
rp2
lp2
rp1
Router Loss
rp3
A
B
C
D
E
F
End−to−End loss p = 1−(1−rp1)(1−lp1)(1−rp2)(1−lp2)(1−rp3)

Figure VII.1: Path metrics can be composed from individual router-level and link-level metrics.

such as delay and loss.

Our proposed architecture of m-Plane for different measurement metrics is based on two critical observations. First, end-to-end connectivity can be composed from individual router- and link-level connectivity information. More generally, many other metrics can also be composed of individual router- and link-level metrics. For example, in Figure VII.1, we can observe that overall path connectivity from A to F is ensured if individual segments that comprises of the path, i.e., AB, BC, CD, DE and EF are all connected. Similarly, the average delay of path AF can be composed by adding the delays of the individual segments AB through EF. The average loss of path AF can be computed as $1 - \Pi(1 - p_i)$ where p_i is the loss probability of a segment along the path.

Second, the typical set of measurements of interest include *both* end-to-end path properties as well as individual hop properties along the path. For example, tomographic approaches including the black hole detection and localization mechanisms used in the previous chapter, measure properties of multiple end-to-end paths to infer individual hop properties. Such an approach is born out of necessity since there is no inherent support from individual routers and

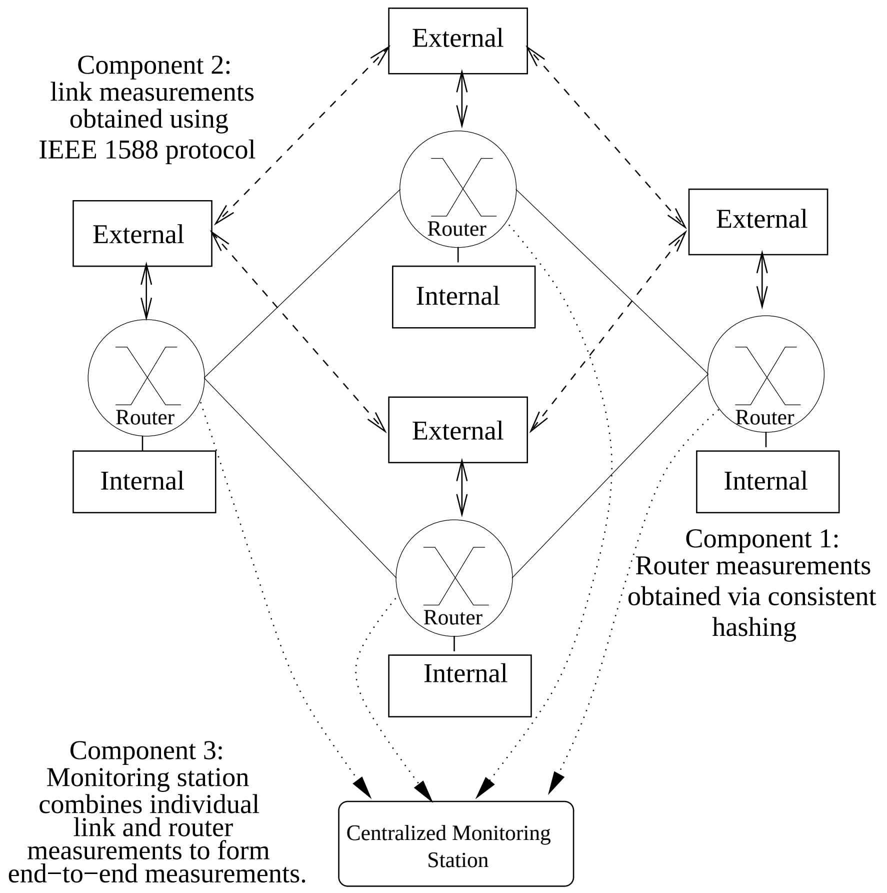


Figure VII.2: m-Plane architecture. Each router is equipped with link-level and router-level measurement modules.

links to accurately characterize measurement properties. On the other hand, since the tomography problem is intrinsically under-constrained, the errors are model-dependent. Our approach follows the inverse approach: we directly measure individual router and link properties to compute both hop-by-hop and end-to-end path properties. Compared to tomographic approaches, composition appears more deterministic and thus less sensitive to traffic models.

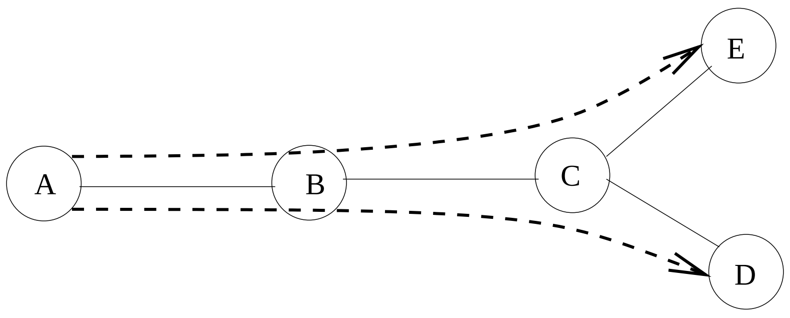
In m-Plane, the responsibility of monitoring a given path is shared between all the individual routers that comprise the path. Each router monitors the forwarding path within the router and the link to the next router along the path. These individual measurements are then forwarded to a centralized monitoring station where the properties of all the required paths are composed from these individual measurements. Of course, the monitoring station needs access to the path between two routers. Such information can be obtained from a passive IGP monitor (such as OSPF monitor [79]). Using this information, the monitoring station identifies the exact path between a given pair of routers, and composes the path properties from individual segment properties.

Thus, the m-Plane architecture shown in Figure VII.2 consists of three components:

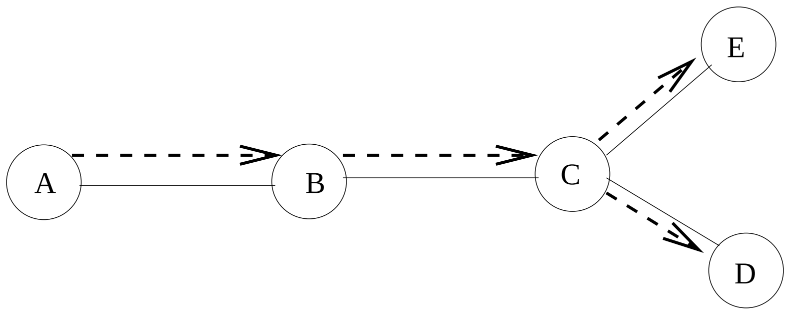
1. **Node Measurement:** First, each router is equipped with an internal measurement module. The internal measurement module is responsible for measurement within the router. It measures metrics of interest (such as connectivity, delay, loss, etc.) for all the internal forwarding paths within a router. For example, a basic set of such forwarding paths consists of every ingress-egress pair in the router. Further, these measurements are obtained by observing data traffic without injecting any special probes into the forwarding path as we shall explain later in Section VII.B.1.

2. **Link Measurement:** Second, each router is equipped with an external measurement module that is responsible for measuring link-level properties. The external measurement module measures link-level metrics from the egress of a router to the ingress of the adjacent router. Minimally, measurements must be done for the set of routers adjacent to a given router in a clean-slate deployment (in Section VII.B.2). However, to support incremental deployment, measurements may also be needed over virtual links to other measurement-friendly routers, which we will explain in Section VII.B.3.

3. **Measurement Composition:** Third, a centralized monitor identifies forwarding paths in the network with the help of a topology monitor (e.g., an OSPF monitor [79]) and composes end-to-end path properties from the internal and external measurements forwarded by each router.



(a) End-to-end active probes



(b) Local link-level active probes

Figure VII.3: Two probes share the same hops

In effect, the internal and external measurement modules within the router provide segment-level measurement properties, which are then combined by the monitoring station to provide end-to-end measurements. We describe these primitives in more detail next.

VII.B.1 Measurement primitives

Measurement recording and reporting are inherently de-coupled in the m-Plane architecture—unlike current active measurement approaches that use probes as both the recording and reporting mechanisms. Today's end-to-end active probes are fundamentally wasteful because multiple probes often probe the same segments. For example, in Figure VII.3(a), we show two different probes, A to D and A to E, that traverse the same path until C, after which they take different paths. The segment from A to C is common to both these probes. A lot of bandwidth, therefore, is unnecessarily wasted due to the tremendous amount of overlap between end-to-end active probes in the network. The m-Plane architecture eliminates this fundamental redundancy by breaking down end-to-end paths into individual router- and link-level segments, that are monitored by routers separately as shown in Figure VII.3(b).

As a result, the total number of probes in the network scales as the square of the number of forwarding paths within a router, i.e., $O(d^2)$, d being the degree, and number of links $O(m)$. The resulting complexity is $O(nd^2+m)$, is much smaller than the $O(n^2)$ complexity ensuing from end-to-end active probing. Additionally, within a router, we can record measurements passively without injecting additional traffic by sampling packets at the ingress and egress of a forwarding path within a router. Thus, with a little additional hardware complexity within the routers, the factor $O(nd^2)$ disappears leaving only $O(m)$ active probes in the network. Of course, the data plane is still used to periodically transmit information to the monitoring station, but the bandwidth required is small and can be scheduled to avoid impacting normal data traffic.

Node measurements: Most of the performance problems in the network occur at routers, where both software (e.g., routing) and hardware (queuing, forwarding and switching) functions can cause non-deterministic delays, losses and connectivity problems. If the metrics are always flow-dependent (e.g., TCP 5-tuple), then our measurement solution would need to measure various metrics on a per-flow basis. Fortunately, we note that in many real routers, forwarding metrics depend on the forwarding class more than a particular flow. For example, all flows traveling between the same input and output ports of a router in a given quality-of-service class are often treated identically in terms of queuing and switch scheduling. Thus, we group such flows into what we call a *measurement equivalence class* (MEC).

Note that we classify only those flows that are treated similarly by a given router as an MEC; we do not assume that two flows that belong to the same MEC in a given router necessarily belong to the same MEC at the next router. However, moving from per-flow measurements to per-MEC measurements is a great improvement in scalability. For example, a router with 16 ports and five QoS classes has $16 \times 15 \times 5 = 1200$ MECs, while most backbone routers may routinely deal with millions of concurrent TCP flows. Even if the flow definition were limited to prefix-to-prefix, or VPN-to-VPN, our results indicate that the number of such flows is much larger than the number of MECs. Once we divide traffic into classes, we can measure most properties of a class by observing a few samples of traffic that belongs to a given class.

Determining where to take measurements inside of a router depends on both the desired granularity and the types of measurements needed. For example, if only aggregate router-level metric is required, then measurement points need to be located only at the router ingress and egress. At the other end of the spectrum, a router vendor interested in internal debugging may instantiate different measurement hooks at almost all major locations within the router. An intermediate stance that allows determining queuing delay (say, for debugging SLA issues) would be to place measurement points at the input of all queues.

To actually conduct measurements, we use consistent hashing (similar to that of trajectory sampling by Duffield *et al.* [26], though only *local* within a router) to sample the same set of packets in a distributed way based on the hash of the contents. At the ingress of a router, a *label* for each packet is generated by hashing invariant content within the packet such as source and destination IP address, ports, identification field and data payloads in hardware (see [26] for reference implementation complexity.) Packets are consistently sampled based on the value of the label and information associated with these labels (e.g., timestamp for measuring delay) is stored at the monitoring point.

All the measurement points within the router use the same hash function; all monitors therefore, record information that belongs to a small subset of packets independently. The original trajectory sampling only collects packet labels at distributed points in the network. In our case, we need to store additional

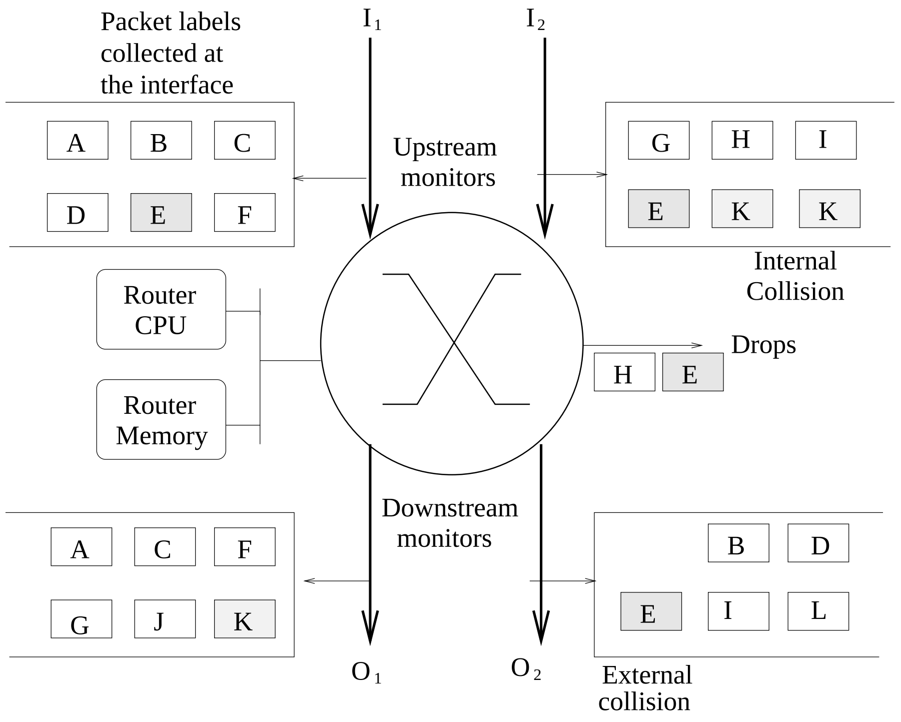


Figure VII.4: Consistent hashing example within router with two input and output ports. Notice the internal (label K) and external (label E)collisions that can occur within the router.

information such as timestamps along with each sampled packet label; we refer to the inclusion of additional state such as a timestamp along with packet labels as *generalized trajectory sampling*.

In Figure VII.4, we show a router with two input ports (I_1 and I_2) and two output ports (O_1 and O_2), thus leading to four MECs. At each of the four interfaces, we perform consistent hashing so that a stream of labels are timestamped at each of the interfaces. For example, at I_1, packet labels $A - F$ are sampled, out of which $\{A, C, F\}$ are forwarded to O_1 and $\{B, D, E\}$ to O_2. Since both I_1 and O_1 use the same hash function, therefore the same labels $\{A, C, F\}$ are also collected at O_1. In addition to these, some packets from I_2 are also forwarded to O_1, resulting in additional labels $\{G, J, K\}$. For measuring delay, each interface needs to associate a hardware timestamp generated by synchronized clocks additionally with each of these packet labels.

The measurement monitors collect labels in a measurement cycle, the duration of which is configured based on system constraints such as memory and bandwidth. A large measurement cycle leads to large local memory requirements at each interface while a smaller measurement cycle may lead to higher bandwidth usage. At the end of each such cycle, a new set of samples are collected while the old ones are flushed to the router memory, accessible to the router CPU as shown in Figure VII.4. The router CPU deduces the required metrics based on the information recorded by the sampled packets within the router. For example, to monitor only connectivity, just the presence of label at both monitoring locations is enough. For example, the liveness of the forwarding path traversed by the packet label C can be deduced from the presence of the label at the upstream monitor, I_1 and the downstream monitor, O_1. Similarly, delay can be obtained by subtracting the timestamps recorded at the two monitors, and loss by computing the fraction of labels that have reached the upstream monitor.

One problem associated with consistent hashing is the potential for hash collisions. Collisions can be internal (within the same interface) or external (across two interfaces). External collisions, in particular, cause ambiguity in associating labels across downstream and upstream monitors (e.g., E in Figure VII.4). We mitigate the impact of collisions in two ways. First, as suggested in [26], we add a few payload bytes during hashing to reduce the chance of an internal collision to less than 10^{-3}. Additionally, we throw away duplicate labels discovered while correlating the label sets from multiple monitors. Assuming good hash functions, throwing away duplicate labels will not lead to measurement bias.

Link measurements: Most link-level properties remain invariant regardless of the traffic distribution. For example, link connectivity information, propagation delay remain independent of whether a packet is a delay-sensitive VoIP packet or a best-effort TCP packet. Packet size affects the transmission time, but it can be scaled linearly. Yet, despite the inherent stability of link measurements, links must be constantly monitored for two reasons. First, lower-layer optical devices can intelligently re-route traffic (e.g., a SONET ring can mask failures), affecting propagation delay and perhaps loss properties without affecting connectivity at the IP layer. Second, loss rates need to be monitored periodically for bit errors resulting from optical fiber and/or component degradations. However, such information need not be classified on a per-flow basis. We use active probes issued at the egress interface of one router to the ingress interface at the other end to measure link-level metrics.

Monitoring station: Each router in the network transmits the router- and link-level measurements to a monitoring station in a measurement-state packet (MSP) to enable composition of path properties. The monitoring station can easily compute liveness of any given end-to-end path by observing whether all the individual segments that comprise that path are alive. Even if one segment is not functional, the end-to-end path is not functional. Because the measurements are individual segment-based, the monitoring station directly knows the location of the fault unlike the indirect mechanisms in Chapter V.

In order to generalize for loss and delay, each router can characterize the distribution of loss and delay experienced by packets within a given MEC using the labels collected at the ingress and egress of the router. Unlike path liveness which is a binary metric, delay and loss are statistical properties for which both mean as well as variance are of interest. Therefore, each router CPU calculates the average delay and loss for each MEC and reports them in the MSP, from which the monitoring station can estimate the mean and variance for the entire path.

VII.B.2 A clean-slate deployment

In this section, we show how to scalably obtain measurements between these various edge routers within an AS in a clean-slate architecture, assuming that all the routers (including the edge routers) in the network can be upgraded to include support for measurements. The paths to be monitored begin at the

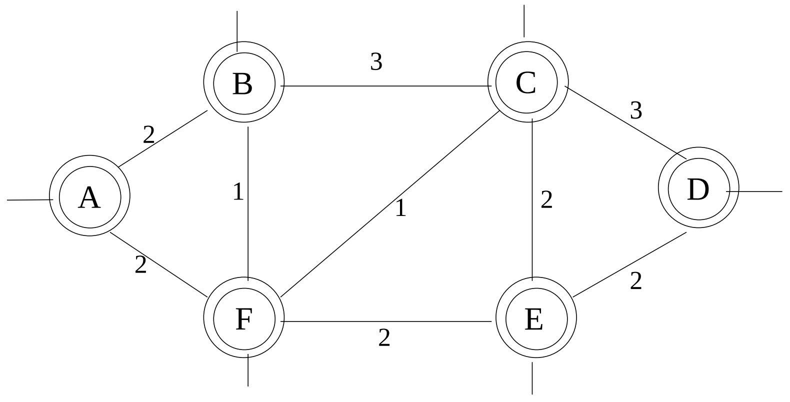
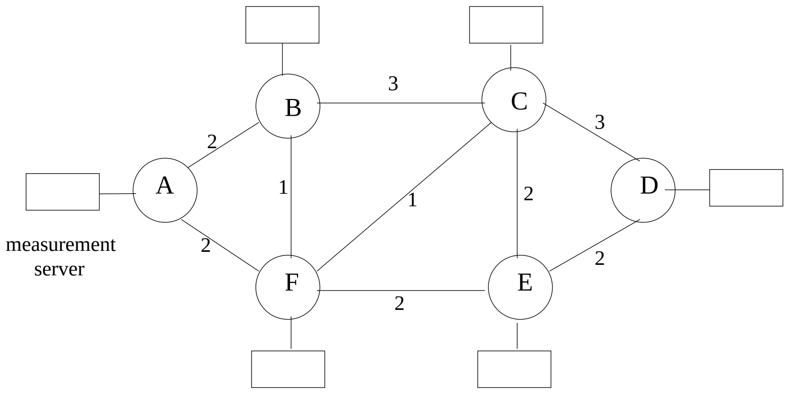


Figure VII.5: Toy topology with six measurement capable routers or m-routers in a clean slate design. The numbers are the associated link costs.

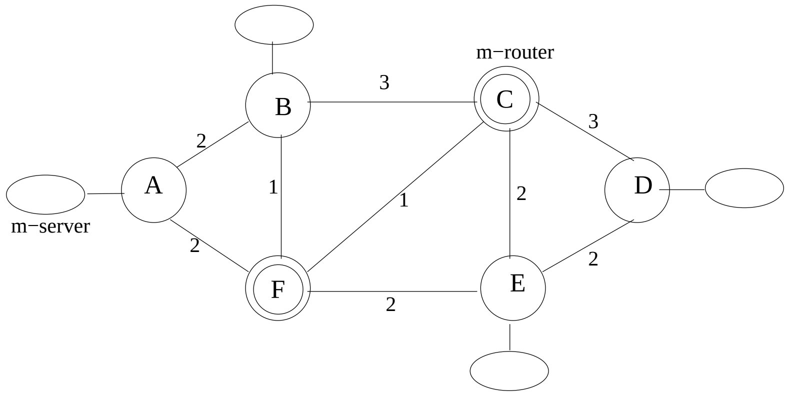
ingress interface to the edge-router to the egress interface of another edge-router. For example, in Figure VII.5, we show a sample topology with six measurement-capable routers, known as *m-routers*. Following the shortest-path routing, the path between A and C goes through F, i.e., $A.F.C$. This path $A.F.C$ can be broken down into the following segments: 1) A's ingress to A's egress (router-level), 2) A's egress to F's ingress (link-level), 3) F's ingress to egress (router-level), 4) F's egress to C's ingress (link-level), 5) C's ingress to C's egress (router-level). Each router needs to individually provide support for monitoring the path segments that either are internal to the router, or originate/terminate at the router.

If all the routers in the network provide such link- and router-level measurements, then it is straightforward to estimate accurate end-to-end properties by combining these measurements for all the routers along the path, as has been shown before. We envision the presence of a centralized monitoring station (or a farm of stations) where each router transmits measurements collected by the router (both at the router and link level) as shown in Figure VII.2. The monitoring station combines these individual measurements reported by each router into the end-to-end properties for all the paths.

Since the monitoring station uses the control plane to construct paths, calculated metrics can differ from actual metrics during periods when the control



(a) Current topology



(b) Partially upgraded topology

Figure VII.6: Partial deployment of the m-Plane architecture. Measurement servers attached to upgraded routers are removed. Old measurement servers are upgraded to m-servers, that also listen to the OSPF LSAs and maintain their own shortest path tree.

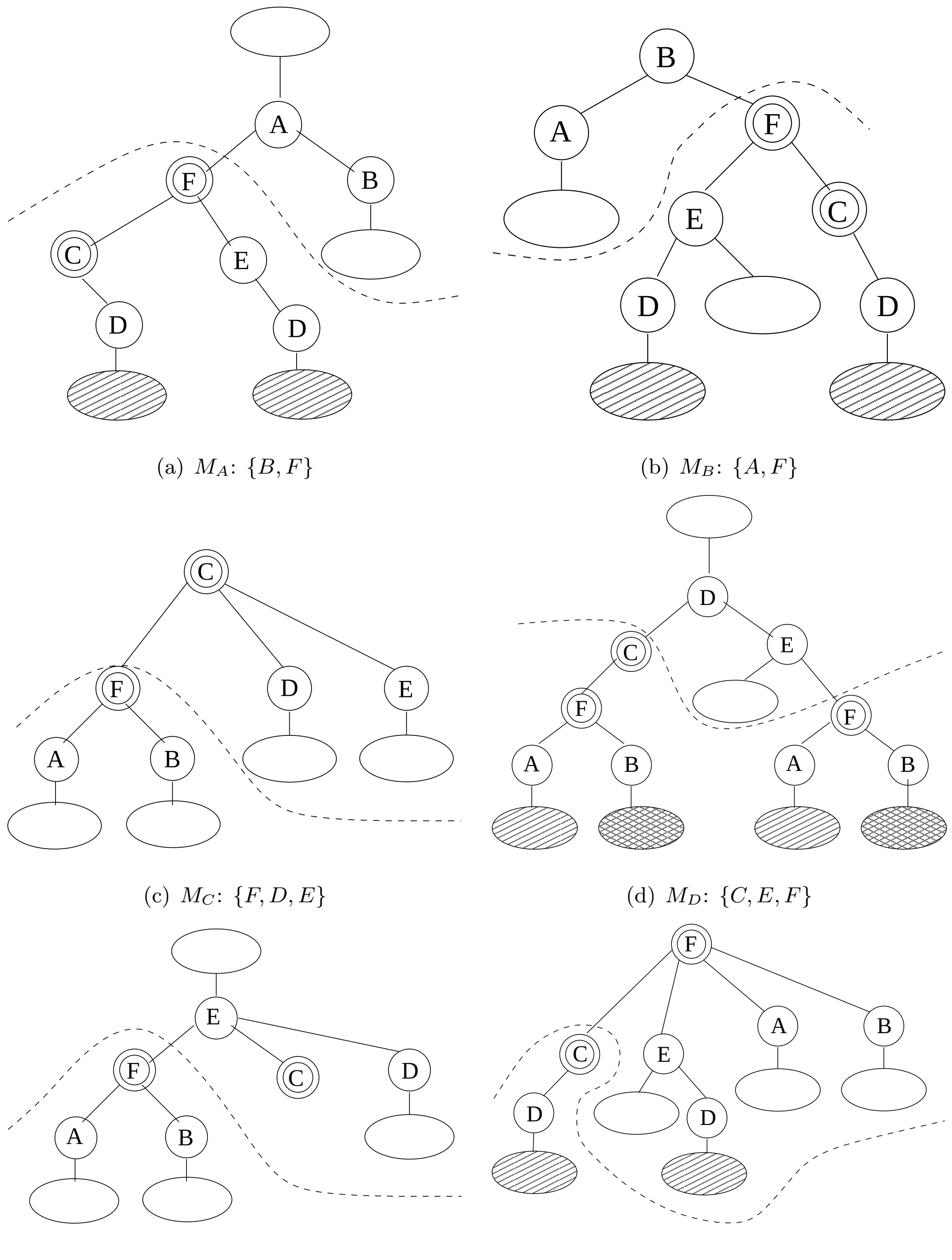
and data planes differ (e.g., during reconvergence events). We assume that for applications of interest and most metrics (e.g., delay, loss) the period for which this discrepancy occurs is small enough to be ignored. However, measuring reconvergence events will require additional mechanisms.

VII.B.3 Incremental deployment

Incremental deployment of the m-Plane architecture is challenging since end-to-end metrics seemingly cannot be composed unless there is cooperation from all the routers along the path. Currently, without support from routers, measurements are performed through measurement servers, which we call *m-servers.* In a partial deployment of m-Plane architecture, we assume that a subset of routers is upgraded to m-routers. In Figure VII.6, we show a toy topology with six routers connected via undirected edges and associated edge costs. Attached to each of the routers is an m-server (shown in Figure VII.6(a)) that issues data-plane probes to other such m-servers to measure path properties of interest. Let us suppose that we chose two out of the six routers to be upgraded to m-routers. This incremental deployment of m-Plane proceeds in three steps discussed below.

Step 1: In the first step shown in Figure VII.6(b), the set of m-servers connected directly to the m-routers are removed since their functionality is subsumed by the m-routers. Further, the set of measurement servers directly connected to non-upgraded routers are transformed into m-servers that also listen to the topology updates (OSPF LSAs) in the network. Thus, the m-servers are capable of reconstructing the forwarding paths in the network similar to the m-routers.

Step 2: In the second step, each m-server or m-router identifies a set of nodes for which it monitors path-properties to. We call this set the *m-set.* It does so by first computing a self-sourced shortest-path spanning tree using Dijkstra's algorithm. The shortest path trees computed at each of the six nodes is shown in Figure VII.7. The m-router does not need to explicitly perform this computation and can leverage the existing shortest-path tree already computed by the OSPF process on the router. It then determines the m-set by making a cut in the tree whenever an m-router or an m-server is encountered. If an m-router is encountered, the rest of the paths to various destinations in the subtree of this m-router are monitored by that m-router (and hence not required by this router). An m-server is encountered if no such m-router exists along the path (and hence it has to monitor this path itself). In Figure VII.7, we show such m-sets for all the routers for the toy topology in Figure VII.6. Note that in Figure VII.7(f), the m-set consists of D_2 that corresponds to the second shortest path to D through E. The



(a) M_A: $\{B, F\}$ (b) M_B: $\{A, F\}$

(c) M_C: $\{F, D, E\}$ (d) M_D: $\{C, E, F\}$

(e) M_E: $\{C, D, F\}$ (f) $M_F : \{A, B, E, D_2\}$

Figure VII.7: In this figure, we show the various shortest-path trees constructed locally by the m-servers and the m-routers to determine which set of segments to monitor. X_i refers to the ith-shortest path to X, in the case when a router X can be reached via multiple shortest paths.

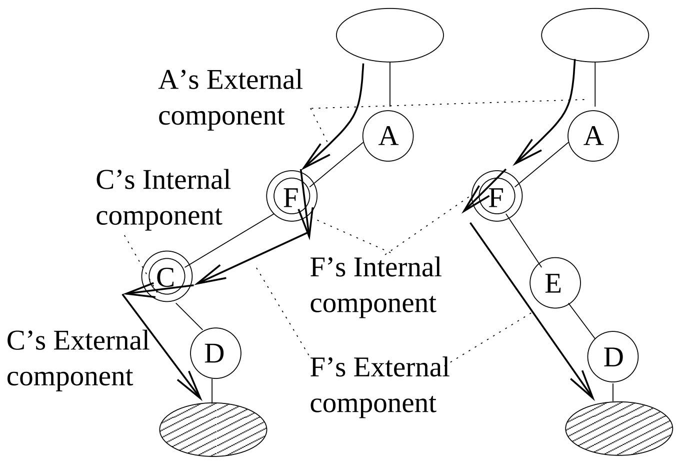


Figure VII.8: Two shortest paths to destination D composed at the monitoring station. The m-server attached to A's external component provides measurement metrics until m-router B. B's and E's internal and external components are used to reconstruct the properties of the rest of the paths to D.

router F does not need to monitor the first shortest path to D through C.

Step 3: Finally, a link between two m-routers (or virtual link consisting of paths through non-upgraded routers) or between an m-router to m-server or between two m-servers are monitored using regular active probing. The m-routers report the internal measurements and the measurements to the nodes in the m-set periodically to a monitoring station using measurement state packets (MSPs). The m-servers only report the measurements to the nodes in the m-set within the MSP.

Next, the monitoring station uses the topology information to build n shortest-path trees with each node as source, and uses the MSPs to completely characterize all the $O(n^2)$ paths in the network. For example, in Figure VII.8, we show how the monitoring station obtains path properties from A's m-server to D's m-server. Recall from Figure VII.7, that A's m-set consists of B and F. Therefore, A's MSP consists of the path properties till router B. B is an m-router; hence B's MSP consists of both the internal and external components. D is reached via two paths, one through E and one through C. The properties of the path through E require further factoring in E's MSP (internal and external components). On the

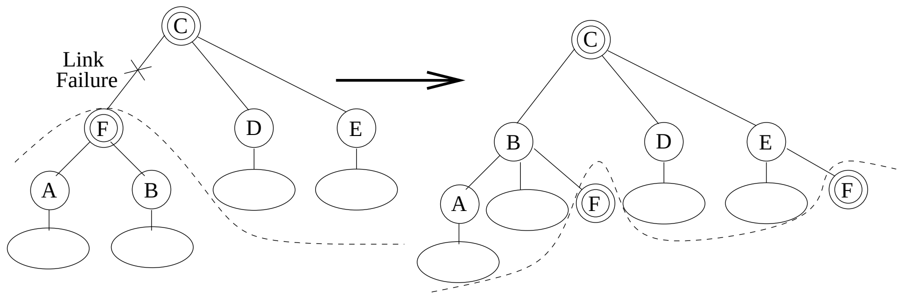


Figure VII.9: Topology changes and recomputation of m-Set. The link $\{C, F\}$ in the topology breaks, triggering recomputation of the m-set. The old m-set consists of $\{F, D, E\}$, while the new m-set consists of $\{A, B, D, E, F_1, F_2\}$, where F_1 and F_2 correspond to the two equal-cost paths to destination F.

other hand, the path through C is directly monitored by the m-router B. These steps can be repeated when other routers are upgraded to m-routers. Eventually, when all the routers are transformed into m-routers, the architecture is exactly the clean-slate architecture we discussed in Section VII.B.2.

So far we have described how to incrementally deploy m-routers but have assumed that all m-servers are upgraded at once. To allow incremental deployment of m-servers, m-routers must continue to work with existing measurement servers which continue to send probes. In such cases, the first m-router can intercept the probe and reflect it back, as if were the final destination. The monitoring station, takes this fact into account by composing the metrics of the extra segment (from the m-router to the destination) with the measurements reported by the measurement server.

VII.B.4 Topology changes

Real networks exhibit a lot of churn; hence, the architecture should accommodate such topology changes. Similar to what happens today, the m-routers (and the m-servers) recompute the new shortest paths when they receive OSPF LSAs, and update their m-set by identifying the cut in the shortest-path tree again.

During such periods, every m-router must continue to measure properties of the links or virtual links to the routers in both old and new m-sets for a configurable amount of time (usually dictated by the routing reconvergence period) to ensure paths are composable at the monitoring station. Afterwards, the m-routers phase out the routers in the old m-set and restrict measurements to only those in the new m-set. Since the MSPs contain the exact m-set along with their measurements, the monitoring station can compute properties for both the old paths as well as the new paths.

For example, in Figure VII.9, we can observe that when the link $\{C, F\}$ is down, a recomputation of the shortest-path tree at the m-router C is triggered. The old m-set consists of just $\{F, D, E\}$, whereas the new m-set consists of $\{A,B,F_1,D,E,F_2\}$, where F_1 and F_2 correspond to the two equal-cost paths to the destination F. Once the link $\{C, F\}$ is up, m-router C reverts back to the old m-set and uses it to report the MSPs. Note that this is unlike the clean-slate architecture, where such m-sets are not explicitly required. Each m-router just maintains all the adjacencies and does not pay heed to the shortest paths. Next, we describe the packet formats required for implementing m-Plane.

VII.B.5 Packet formats

The m-Plane architecture requires two other types of messages: one to advertise the location of m-routers, and one to transmit measurement data to the central monitoring station (known as a measurement state packet, or MSP).

Advertising the presence of m-routers. Each m-router needs to identify the presence of other m-routers in the network in order to construct its m-set. One way to do this is to configure each of the m-routers with information about the other m-routers in the network. Every time a new upgraded router is added into the network, however, all the existing m-routers would need to be reconfigured. Instead, we leverage the existing OSPF protocol to allow m-routers to advertise their presence to other m-routers in the network. (Similar modifications could be made to IS-IS or any other link-state intra-domain routing protocol.) We propose to use one of the reserved bits in the *Options* field [65] of the OSPF control-plane messages for this task; the field exists precisely to advertise special capabilities of routers in the network.

Currently, two out of eight bits reserved for the options field are used: the T-bit (to indicate type-of-service capability) and the E-bit (to indicate external routing capability). We use one of the six unused bits (which we call the M-Bit) to advertise the presence of an m-router. A router that transmits OSPF control-plane messages with this bit set is capable of performing and reporting router- and link-level measurements. Legacy routers in the network do not pay attention to this bit. All OSPF control plane messages contain the options field; while OSPF HELLO messages are not transported beyond the next-hop, the link-state advertisements are flooded throughout the network. Thus, every m-router in the network learns of the presence of other m-routers in the network, while other routers operate as usual.

MSP packet format. MSPs contain two components—internal and external—that correspond to the link- and router-level measurements. The external components of the MSP consist of the various links that are monitored by the router. There are many ways in which a link can be represented. A common practice in many ISP networks is to represent the two ends of a link with IP addresses. Typically, the IP address for the two ends of a link are part of a /30 prefix. Another way a link can be represented is by using the SNMP interface numbers on the routers. Regardless of the specific scheme used to represent the interfaces, the MSPs contain information about the links represented by two 32-bit identifiers (for the source and destination) and their associated measurement properties.

While the MSP format need not be standardized since none of the m-routers or m-servers directly communicate with each other, in the interest of concreteness we describe a sample layout of the fields. The exact packet format is subject to change, but the fields identified in the Figure VII.10 are required. Each

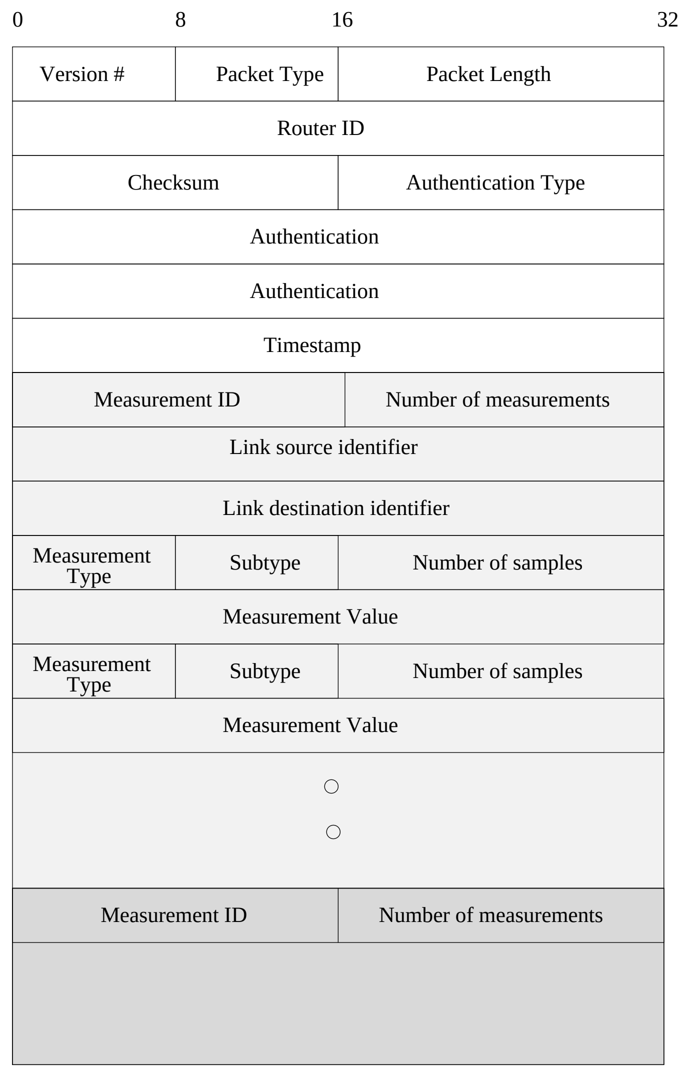


Figure VII.10: Packet format of the measurement state packet (MSP).

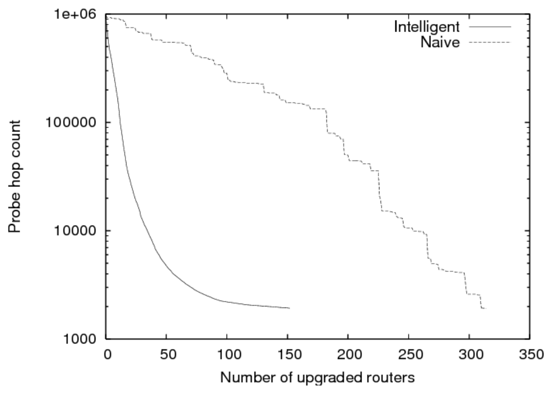
MSP consists of multiple records, with each record consisting of a record header consisting of the measurement identifier and number of <attribute,value> pairs in the record. The next two 32-bit fields are reserved for source and destination identifiers of the measurement, followed by the <attribute, value> pairs. The attribute field is specified as a measurement type (delay, loss, etc.), measurement subtype (mean, max, average, std. deviation, etc.), followed by the number of samples used to arrive at these statistics. The next 32-bit field is for the value of the measurement. Several of these <attribute, value> pairs are defined in one measurement record.

VII.C Overhead reduction

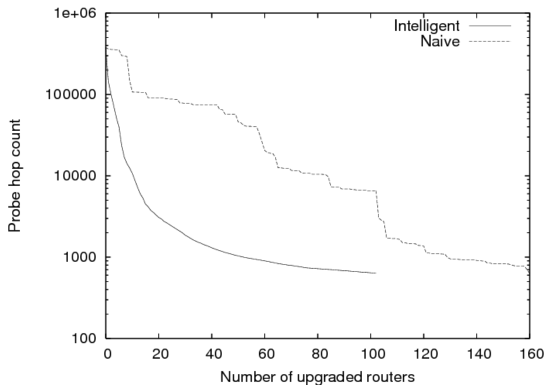
We now attempt to quantify the benefits achieved by incrementally deploying our architecture in real networks. Lacking access to actual tier-one ISP topologies, we conducted our evaluation using the Rocketfuel topologies annotated with inferred link weights [84]. Despite the known deficiencies of this data, they suffice to demonstrate general trends. We compare the benefits of upgrading in a naive (random) fashion to an intelligent upgrade strategy.

We use a simple metric called *probe hop count* to quantitatively describe the benefit achieved by upgrading existing routers to m-routers. Probe hop count is defined as the sum of all the hops taken by every active probe that traverses the network. When active probes are issued from every measurement server to another, this translates to the sum of hop-lengths of all the $O(n^2)$ shortest-paths (including the multiple paths between a given pair of routers) in the network. On the other hand, in the m-Plane architecture, the probe hop count reduces to the total number of links in the network, since each m-router transmits messages only to its adjacent routers.

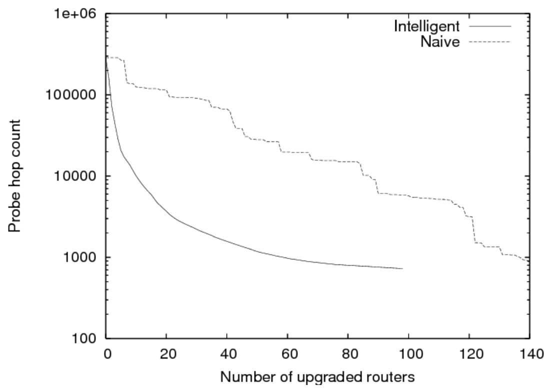
In order to identify candidate routers to upgrade, we guide the search in the direction of reducing the probe hop count metric as much as possible as shown in Algorithm 6. In particular, we select the routers that reduce the probe hop count the most. Our algorithm first calculates the shortest paths between all pairs of end points, including the duplicates (which may be used by equal-cost multi-path routing algorithms). It then computes the number of shortest paths that traverse each router by incrementing the counts for all intermediate routers on each path (excluding the source and destination of a path). Then, it selects the router with the maximum count as the router to upgrade. To select additional routers, the algorithm breaks all paths traversing the selected router into two



(a) Sprint (314 routers)



(b) Tiscali (160 routers)



(c) AboveNet (140 routers)

Figure VII.11: Incremental benefit for various ISP topologies. Notice that the y-axis is log scale.

Algorithm 6 `IdentifyRoutersToUpgrade(V, E, numUpdate)`

```
1: S = ComputeShortestPaths(V, E);
2: U = {};
3: numiter = 0;
4: while (numiter < numUpdate) do
5:     for path p ∈ S do
6:         for router r ∈ p − {src, dst} do
7:             count[r] + +;
8:         end for
9:     end for
10:    maxRouter = findMax(count)
11:    U = U + {maxRouter}
12:    for path p ∈ S do
13:        if maxRouter ∈ p then
14:            S = S − {p}
15:            p_1 = split p from src till maxRouter
16:            p_2 = split p from maxRouter till dst
17:            $S = S + \{p_1, p_2\}$
18:        end if
19:    end for
20:    numiter + +;
21: end while
```

segments—source to the router and router to destination—and adds them to the set of shortest paths. If any of these segments already exist then they are not added to avoid double counting. Note that only the paths that pass through a router contribute to the count of that router; paths that originate or terminate at a given router do not contribute to its count. This step is important to ensure that the search process always identifies intermediate routers, as opposed to access routers, which do little to break up source-destination paths.

Figure VII.11 shows the results of both upgrade strategies on three representative Rocketfuel AS topologies (results were similar on all of the topologies we considered). The curve for all the topologies is convex in shape; upgrading the first few routers results in maximum benefit, while the marginal benefit reduces drastically after a while. On average, upgrading about 15% of the routers in an intelligent fashion results in a two order-of-magnitude reduction in the probe hop count. For example, the Sprint topology in Figure VII.11(a) requires approximately one million end-to-end active probes to measure each path without any upgraded routers. Upgrading 45 routers out of 315 results in a probe hop count of only 10,000—a two order-of-magnitude reduction in measurement overhead.

VII.D Related work

Many techniques have been suggested in the literature for measurement in backbone networks. These techniques can be broadly classified into three categories.

Active measurement. Active measurement involves injecting synthetic data-traffic into the network to measure path metrics of interest. There exist many different tools publicly available for measuring specific properties, such as end-to-end delay and loss [61, 77, 83], available bandwidth [40, 91], per-hop capacity and so on (see [4] for references to many other available tools). While these tools are based on sound statistical foundations, the active measurement approaches appear inherently intrusive. Their presence is necessitated by the lack of inherent measurement support from the routers.

Inference techniques. Another class of mechanisms combines router-level coarse link measurements to infer path properties. A classic example is traffic-matrix estimation [94, 101], where traffic demands between every pair of edge routers are estimated using individual link-level SNMP loads [64]. Other tomographic approaches (e.g., [104]) measure end-to-end path properties via active probes and use topology to infer individual hop-properties. The main limitation of these approaches appears to be the assumptions that go into the inference model.

New router primitives. Finally, the third class of measurements is based on router primitives. For example, Machiraju *et al.,* in [59] argues for a measurement friendly network architecture where individual routers provide separate priority levels for active probes. Duffield *et al.* suggest the use of router support for sampling packet trajectories [26]. Many high-speed router primitives have also been suggested in the literature for measurement [24, 29]. Finally, Cisco and other router vendors (e.g., Juniper) provide basic measurement primitive called NetFlow [66] that is used extensively by network operators for billing, accounting, traffic matrix estimation, anomaly detection and other such applications.

Many of these solutions however are problem-specific and, to the best of our knowledge, it appears that there have not been many attempts to design a scalable router primitives to estimate end-to-end path properties in the literature.

VII.E Summary

In this chapter, we have initiated research into designing router primitives to make measurement a first-class entity. We have proposed initial models quantifying the inherent inefficiency in current measurement approaches using active probes. We have shown the efficiency gains in a clean slate architecture can range from a factor of 35 to a factor of 100. Motivated by this observation, we proposed a clean slate router architecture with new mechanisms for scalable monitoring of nodes and links. While the basic primitives are simple (as they must be to be implemented in hardware) we found that the incremental deployment protocol (which can fortunately be implemented in software) was much more challenging to design. As a first-order metric, in this chapter, we focused on efficient scalable mechanisms for connectivity monitoring. Future work requires a more complete treatment of many other metrics of interest, such as delay, loss and jitter using the same basic ideas.

VII.F Acknowledgments

This work is joint with George Varghese and Alex C. Snoeren and is not yet published.

Chapter VIII

Conclusions

Our work has been motivated by a simple observation: Current failure monitoring mechanisms in the network detect that a failure happened, but do not indicate either the root cause or the location of the failure, thus requiring additional mechanisms for fault localization. To reduce the overall repair or outage duration, fault localization needs to be fast. Since manual processes are often slow, we argue that fault localization should be automated, and we presented mechanisms based on risk modeling to automate fault localization in common failure modes.

However, not all failure instances require a lot of effort to localize the root cause, even among silent failures. Sometimes, just a simple visual inspection of the observed symptoms is enough to localize the failure. For example, when all the failed MPLS tunnels share one end point due to a failure near the edge, it does not take much effort (a few seconds) to pin-point the location. On the other hand, if the actual fault lies in the core of the network, visual inspection alone is not sufficient to localize unless we join the failure signature with the risk model. In such cases, it can take minutes to even hours to localize the failure with today's manual approaches; an automated localization system such as ours can reduce the duration significantly.

Today's backbone networks are rife with such instances, where visual inspection alone is not enough to isolate the failed component. In this dissertation, we focused on two such instances—IP link fault and black hole localization—for which we have designed, implemented and deployed systems in a tier-one backbone network. Both our systems apply risk modeling to encode dependencies between a given set of root causes and the set of symptoms dependent on them. This simple dependency model is surprisingly powerful in representing a wide variety of fault-localization problems, such as the two problems considered in this dissertation. We begin this chapter with a summary of our experience followed by open challenges and future work.

VIII.A Experience using risk modeling

Given that the risk model is at the heart of our localization methodology, it is important to devise the right risk model for localization. Primarily, the risk model originates from operational domain knowledge through a careful analysis and understanding of the type of failures that one experiences. For example, for MPLS fault localization, we observed from operational experience that the primary root cause for most failures is a topology change. Therefore, our risk model consisted only of IP links. Had we observed that optical failures were causing black holes, we would have modeled optical layer equipment such as amplifiers, fibers etc., in the risk model. Moreover, the risk model has to match the failure-detection system. For example, in the black-hole localization problem, there is no need to model customer facing links in the topology due to the fact that the measurement system using active probes never traverses any of those links.

Constructing the right risk model is not easy, even if the category of risks to be modeled is known. For example, in an OSPF network, as we have seen before, multiple paths can exist between a given source and a destination if the paths share the same cost (ECMP [37]). The router at the first fork in the paths splits traffic equally among the paths based on a deterministic but unknown hash-function applied on the source-destination IP address of the packet. The correct risk model should incorporate the dependencies along the exact path taken by a probe, which unfortunately is frequently not known precisely.

Similarly, in composite links [9], many optical circuits are bundled together into one logical IP link and the router splits traffic according to a hash-function. Failures involving only a member circuit can result in only some probes that traverse the member circuit (out of all the member circuits) to fail, while others succeed. In both of these cases, the risk model needs to be constructed based on the instantaneous path traversed by the probe, which is difficult. Therefore, we were forced to consider a risk model that represented the union of all the paths, which is not entirely accurate.

Also, it is often not enough to just model the risks once; determining the right risks to model is a continuous process in many cases. For example, in the IP fault localization scenario, we observed that modeling an OSPF area as a software risk shared by many IP links was not enough. One particular failure scenario involved only 70% of the OSPF area, which we determined using SCORE's error threshold. Upon further investigation, we found that we had to introduce a new risk model, an OSPF area with MPLS enabled, in order to correctly capture the particular failure scenario. Of course, the error threshold was helpful in determining that none of the risk groups represented an exact fit with the failure signature. In general, therefore, the risk model should be continuously updated based on the various failures we observe.

Risk models are almost always dynamic; the rate at which a given risk model exhibits churn varies depending on the problem. Because of this churn, there could be differences between the risk model and reality that affect localization. For example, if humans are managing the topology information from which the risk model is constructed, there is a strong likelihood that the risk model is out-of-sync with reality due to human errors. Our IP fault localization system uses error thresholds to deal with these. Of course, exactly identifying the errors automatically is a challenge. In contrast, automatic generation of the risk model by querying the network, in order to keep the risk model consistent with reality, can be burdensome to the network elements.

Ultimately, the most important aspect of risk modeling is the access to dependency data. In both of our examples, we were fortunate to have access to the associations between root causes and symptoms. For example, in the IP fault localization scenario, our risk models were based on SRLGs already maintained by ISPs for planning diversity in the network. Similarly, we have access to the associations—although not directly—between MPLS tunnels and IP links using a OSPF monitor. Without access to these dependencies, the risk-modeling approach is not possible. While we discussed automatically determining accurate dependencies in the context of IP links and optical components, there are several other instances in literature where such uncovering such associations proves to be challenging [10].

After identifying the right risk model for a given problem, the next issue is determining the right fault-localization algorithm to use. We believe it is a perplexing fact that simple greedy-based approximation algorithms output hypothesis close enough to ground truth. Indeed, we demonstrated that both SCORE as well as MAX-COVERAGE algorithms work well in practice for the particular problems they attempt to solve. We conjecture that this phenomenon could be because most real-life failure scenarios, especially in the problem domains we considered, tend to be simple in nature, and, hence, lend themselves to efficient localization even with the simplest of heuristics.

The additional advantage of any other more complicated inference technique, such as Bayesian analysis or more powerful statistical techniques, appears to be marginal as there is little room for improvement that is not worth the additional complexity, at least in the problem domains considered in this dissertation. On the other hand, if the scenarios are sufficiently complex or are not modeled properly in the risk model to begin with, we believe that it is not easy for any localization algorithm, including complicated inference techniques, to be accurate.

Thus, in our experience, especially in situations where risk models can be large, simplicity can translate to computational feasibility and effectiveness to the users of the tool.

VIII.B Future work and open research problems

One of the key operational realities that we pointed out in Chapter IV is the lack of accurate SRLG databases. One of the useful by-products of our system is the ability to identify database errors, which can help network operators clean up their databases. For example, in the context of IP fault localization, the hypothesis corresponding to a particular failure signature may contain two SRLGs, and by reducing the threshold a little might result in one SRLG. Are there really two failures in the network or is the database incorrect ? The tool currently has no automated way of differentiating the two and leaves it to the operator to manually determine this. It is an open problem to devise mechanisms to distinguish between the two, perhaps by correlating with some additional information.

One of the most promising research directions, we believe, is in the direct localization of failures. As we have shown in Chapter VII, direct localization of performance problems to the granularity of a link or router is possible with simple router-level primitives. More generally, if every component, either software or hardware, closely monitors the inputs and outputs and correlates them for each type of traffic, then we can reduce the root-cause set significantly. While we have shown that our primitives work in the case of connectivity monitoring, and to some extent delay and loss, extending this to other metrics such as available bandwidth and jitter is an open problem. The primitives we have designed primarily monitor data traffic characteristics. Extending these primitives to monitor protocol properties such as reconvergence time is an exciting and challenging problem. With new applications such as video and voice, it is also an interesting problem to devise mechanisms in the network to monitor and localize application-level performance.

Fault management is an important piece of the overall manageability puzzle of any large backbone network. We have begun by observing that fault localization is the most time-consuming aspects of fault management today. Consequently, this dissertation focused on providing fast and accurate fault localization mechanisms for many common failure modes observed in practice. While this dissertation is only a modest step towards building large-scale robust autonomous systems that are capable of both self-diagnosis as well as self-repair, we believe it can be viewed as a significant step in fault localization.

Glossary

AS	Autonomous system
BGP	Border gateway protocol
CE	Customer edge
DoS	Denial of service
DWDM	Dense wavelength division multiplexing
ECMP	Equal cost multi-path
GMPLS	Generalized multi-protocol label switching
IGP	Interior gateway protocol
IPFL	IP fault localization
IS-IS	Intermediate system - intermediate system
ISP	Internet service provider
LDP	Label distribution protocol
LMP	Link management protocol
LoS	Loss of signal
LSA	Link-state advertisement
LSP	Label-switched path

MEC	Measurement equivalence class
MPFM	MPLS fault monitoring
MPLS	Multi-protocol label switching
MSP	Measurement state packet
OD	Origin-destination
OSPF	Open shortest path first
PE	Provider edge
PoP	Point of presence
PPP	Point-to-point protocol
QoS	Quality of service
RSVP	Reservation protocol
SLA	Service level agreement
SNMP	Simple network management protocol
SONET	Synchronous optical network
SRLG	Shared risk link group
VPN	Virtual private network

Bibliography

[1] Avici TSR router. http://www.avici.com/documentation/datasheets/Avici_TSR.pdf.

[2] British Telecom. http://www.bt.com.

[3] Cisco ONS 15600 reference manual. http://www.cisco.com/en/US/products/hw/optical/ps4533/products_technical_reference_book09186a008069b5a3.html.

[4] NLANR Network Performance and Measurement Tools. http://dast.nlanr.net/NPMT/.

[5] Occam's razor. http://en.wikipedia.org/wiki/Occam's_Razor.

[6] Sample Configuration for BGP with Two Different Service Providers (Multihoming). http://www.cisco.com/warp/public/459/27.html.

[7] University of Oregon Route Views Project. http://www.routeviews.org.

[8] L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas. LDP Specification. RFC 3036, IETF, Jan. 2001.

[9] AVICI Systems Inc. http://www.avici.com.

[10] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *ACM SIGCOMM*, Aug. 2007.

[11] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. RFC 2475, IETF, Dec. 1998.

[12] S. Brugbosi, G. Bruno, et al. An expert system for real-time fault diagnosis of the Italian telecommunications network. In *3rd Symposium on Integrated Network Management*, pages 617–628, 1993.

[13] R. Bush and D. Meyer. Some Internet architectural guidelines and philosophy. RFC 3439, IETF, Dec. 2002.

[14] R. Callon. Use of OSI IS-IS for routing in TCP/IP and dual environments. RFC 1195, IETF, Dec. 1990.

[15] J. Cao, D. Davis, S. V. Wiel, and B. Yu. Time-varying network tomography. *Journal of American Statistcal Association*, 95(452):1063–1075, 2000.

[16] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A simple network management protocol (SNMP). RFC 1157, IETF, May 1990.

[17] C. S. Chao, D. L. Yang, and A. C. Liu. An automated fault diagnosis system using hierarchical reasoning and alarm correlation. In *Journal of Network and Systems Management*, volume 9, pages 183–202, 2001.

[18] S. Chaudhuri, G. Hjalmtysson, and J. Yates. Control of lightpaths in an optical network. Jan. 2000. `http://www.research.att.com/areas/opticalnetworking/IPoverWDMpublications.html`.

[19] M. Chen, A. Zheng, J. Lloyd, M. I. Jordan, and E. Brewer. A statistical learning approach to failure diagnosis. In *International Conference on Autonomic Computing*, New York, NY, May 2004.

[20] D. Scott. Making smart investments to reduce unplanned downtime. Tactical Guidelines, TG-07-4033, Gartner Group Research Note, Mar. 1999.

[21] B. Davie and Y. Rekhter. *MPLS: technology and applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

[22] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *Symposium proceedings on Communications architectures & protocols*, pages 1–12. ACM Press, 1989.

[23] R. H. Deng, A. A. Lazar, and W.Wang. A probabilistic approach to fault diagnosis in linear lightwave networks. In *Integrated Network Management III*, pages 697–708, Apr. 1993.

[24] A. Dobra, M. Garofalakis, J. E. Gehrke, and R. Rastogi. Processing complex aggregate queries over data streams. In *ACM SIGMOD*, 2002.

[25] J. C. Doyle, J. Carlson, S. H. Low, F. Paganini, G. Vinnicombe, W. Willinger, J. Hickey, P. Parrilo, and L. Vandenberghe. Robustness and the Internet: Theoretical foundations. In *Robust design: a repertoire from biology, ecology, and engineering*. Oxford University Press, 2003.

[26] N. G. Duffield and M. Grossglauser. Trajectory sampling for direct traffic observation. In *Proceedings of the ACM SIGCOMM*, pages 271–282, Aug. 2000.

[27] A. Elwalid, C. Jin, S. H. Low, and I. Widjaja. MATE: MPLS adaptive traffic engineering. In *IEEE Infocom*, 2001.

[28] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *SIGCOMM Internet Measurement Workshop*, Nov. 2001.

[29] C. Estan and G. Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. In *ACM Trans. Comput. Syst.*, Aug. 2003.

[30] L. Fang, A. Atlas, F. Chiussi, K. Kompella, and G. Swallow. LDP failure detection and recovery. *IEEE Communications*, 42(10):117–123, Oct. 2004.

[31] M. Fecko and M. Steinder. Combinatorial designs in multiple faults localization for battlefield networks. In *IEEE Military Commun. Conf (MILCOM)*, 2001.

[32] G. Forman, M. Jain, M. Mansouri-Samani, J. Martinka, and A. C. Snoeren. Automated whole-system diagnosis of distributed services using model-based reasoning. In *9th IFIP/IEEE Workshop on Distributed Systems: Operations and Management*, Oct. 1998.

[33] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine*, pages 118–124, October 2002.

[34] B. Gruschke. Integrated event management: Event correlation using dependency graphs. In *9th IFIP/IEEE Workshop on Distributed Systems: Operations and Management*, Oct. 1998.

[35] A. Gunnar, M. Johansson, and T. Telkamp. Traffic matrix estimation on a large ip backbone: A comparison on real data. In *ACM Internet Measurement Conference*, October 2004.

[36] P. Hong and P. Sen. Incorporating non-deterministic reasoning in managing heterogeneous network. In *Integrated Network Management II*, pages 481–492, Apr. 1991.

[37] C. Hopps. Analysis of an equal-cost multi-path algorithm. RFC 2992, IETF, Nov. 2000.

[38] K. Houck, S. Calo, and A. Finkel. Towards a practical alarm correlation system. In *4th IEEE/IFIP Symposium on Int. Net. Mgmnt.*, 1995.

[39] HP Technologies, Open View. `http://www.openview.hp.com`.

[40] M. Jain and C. Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput. *IEEE/ACM Transactions in Networking*, 11(4):537–549.

[41] G. Jakobson and M. D. Weissman. Alarm correlation. *IEEE Network*, 7(6):52–59, Nov. 1993.

[42] C. Jin, H. Wang, and K. G. Shin. Hop-count filtering: An effective defense against spoofed DDoS traffic. In *ACM Conference on Computer and Communications Security (CCS)*, Oct. 2003.

[43] I. P. Kaminow and T. L. Koch. Optical Fiber Telecommunications IIIA, editors, 1997.

[44] S. Kandula, D. Katabi, B. Davie, and A. Charny. TeXCP: Responsive Yet Stable Traffic Engineering. Aug. 2005.

[45] S. Kandula, D. Katabi, and J. P. Vasseur. Shrink: A tool for failure diagnosis in IP networks. In *Proc. ACM SIGCOMM MineNet Workshop*, Aug. 2005.

[46] R. M. Karp. Reducibility among combinatorial problems. *R. E. Miller and J. W. Thatcher (editors): Complexity of Computer Computations*, pages 85–103.

[47] D. Katz, K. Kompella, and D. Yeung. Traffic engineering extensions to OSPF version 2. RFC 3630, Sept. 2003.

[48] Katzela and Schwartz. Schemes for fault identification in communication networks. *IEEE/ACM Transactions on Networking*, 3, 1995.

[49] S. Kliger, S. Yemini, Y. Yemini, D. Ohlse, and S. Stolfo. A coding approach to event correlation. In *Fourth International Symposium on Integrated Network Management*, 1995.

[50] M. Kodialam, T. Lakshman, and S. Sengupta. Efficient and robust routing of highly variable traffic. In *ACM HotNets*, Nov. 2004.

[51] R. Kompella, A. Greenberg, J. Rexford, A. C. Snoeren, and J. Yates. Cross-layer visibility as a service. In *ACM HotNets*, Nov. 2005.

[52] R. Kompella, S. Singh, and G. Varghese. On scalable attack detection in the network. *IEEE/ACM Transacations on Networking*, 15(1), Feb. 2007.

[53] R. Kompella and G. Varghese. Reduced state fair queuing in core and edge routers. In *NOSSDAV*, June 2004.

[54] R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren. IP fault localization via risk modeling. In *Proc. Networked Systems Design and Implementation*, May 2005.

[55] R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren. Detection and localization of network black holes. In *IEEE Infocom*, May 2007.

[56] J. Lang. Link management protocol (LMP). In *Internet Draft, draft-ietf-ccamp-lmp-10.txt*, Oct. 2003.

[57] G. Li, D. Wang, R. Doverspike, C. Kalmanek, and J. Yates. Economic analysis of IP/Optical network architectures. In *Proc. Optical Fiber Communication Conference*, Mar. 2004.

[58] G. Liu, A. K. Mok, and E. J. Yang. Composite events for network event correlation. In *Integrated Network Management VI*, Boston, MA, May 1999.

[59] S. Machiraju and D. Veitch. A measurement-friendly network (mfn) architecture. In *Proceedings of ACM SIGCOMM Workshop on Internet Network Management (INM)*, Pisa, Italy, September 2006.

[60] R. Mahajan, S. Bellovin, S. Floyd, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. *ACM SIGCOMM CCR*, 32(3), July 2002.

[61] J. Mahdavi, V. Paxson, A. Adams, and M. Mathis. Creating a scalable architecture for Internet measurement. In *INET'98*, July 1998.

[62] E. Mannie. Generalized multi-protocol label switching (GMPLS) architecture. RFC 3945, Oct. 2004.

[63] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: Existing techniques and new directions. In *ACM SIGCOMM*, Pittsburg, USA, August 2002.

[64] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: Existing techniques and new directions. In *Proceedings of the ACM SIGCOMM*, 2002.

[65] J. Moy. RFC 2328: OSPF Version 2, Apr. 1998.

[66] Cisco NetFlow. `http://www.cisco.com /warp /public /732 /Tech /netflow`.

[67] A. Nucci, R. Cruz, N. Taft, and C. Diot. Design of IGP link weights for estimation of traffic matrices. In *IEEE Infocom*, Hong Kong, March 2004.

[68] Y. A. Nygate. Event correlation using rule and object based techniques. In *Integrated Network Management*, pages 278–289.

[69] L. Phifer. SLAs meet VPNs. `http://www.isp-planet.com/business/slas_for_vpns1.html`.

[70] P.Wu, R. Bhatnagar, L. Epshtein, M. Bhandaru, and Z. Shi. Alarm correlation engine (ACE). In *Network Operation and Management Symposium*, pages 733–742, 1998.

[71] S. Raghunath, K. Ramakrishnan, S. Kalyanaraman, and C. Chase. Measurement based characterization and provisioning of IP VPNs. In *Internet Measurement Conference*, 2004.

[72] R. Ramaswami and K. Sivarajan. *Optical Networks : A Practical Perspective.* Academic Press/Morgan Kaufmann, Feb. 1998.

[73] Y. Rekhter and T. Li. RFC 1771: A Border Gateway Protocol 4 (BGP-4), Mar. 1995.

[74] R. L. Rivest and C. E. Leiserson. *Introduction to Algorithms.* McGraw-Hill, Inc., New York, NY, USA, 1990.

[75] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. RFC 3031, IETF, Jan. 2001.

[76] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, Nov. 1984.

[77] S. Savage. Sting: a TCP-based network measurement tool. In *USITS'99*, pages 71–79, 1999.

[78] P. Sebos, J. Yates, D. Rubenstein, and A. Greenberg. Effectiveness of shared risk link group auto-discovery in optical networks. In *Optical Fiber Comm. Conf.*, Mar. 2002.

[79] A. Shaikh and A. Greenberg. OSPF monitoring: Architecture, design and deployment experience. In *NSDI*, 2004.

[80] M. Shreedhar and G. Varghese. Efficient fair queueing using deficit round robin. In *ACM SIGCOMM*, pages 231–242, 1995.

[81] SMARTS Inc. `http://www.smarts.com`.

[82] H. Smit and T. Li. Intermediate system to intermediate system (IS-IS) extensions for traffic engineering (TE). RFC 3784, IETF, June 2004.

[83] J. Sommers, P. Barford, N. Duffield, and A. Ron. Improving accuracy in end-to-end packet loss measurement. In *ACM SIGCOMM*, 2005.

[84] N. Spring, R. Mahajan, and D. Wetherall. Measuring isp topologies with rocketfuel. In *ACM SIGCOMM*, 2002.

[85] M. Steinder and A. Sethi. End-to-end service failure diagnosis using belief networks. In *Network Operation and Management Symposium*, Florence, Italy, Apr. 2002.

[86] M. Steinder and A. Sethi. Increasing Robustness of Fault localization through Analysis of Lost, Spurious and Positive Symptoms. In *IEEE Infocom*, 2002.

[87] M. Steinder and A. S. Sethi. A survey of fault localization techniques in computer networks. *Science of Computer Programming*, 53:165–194, 2004.

[88] W. R. Stevens. *TCP/IP Illustrated, Volume 1: The Protocols.* Addison Wesley, Reading, Massachusetts, 1994.

[89] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high speed networks. In *ACM SIGCOMM*, Sept. 1998.

[90] J. Strand, A. Chiu, and R. Tkach. Issues for routing in the optical layer. In *IEEE Communications Magazine*, Feb. 2001.

[91] J. Strauss, D. Katabi, and F. Kaashoek. A Measurement Study of Available Bandwidth Estimation Tools. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference '03*, Miami, Florida, October 2003.

[92] A. S. Tanenbaum. *Computer networks.* Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[93] C. Tebaldi and M. West. Bayesian inference on network traffic using link count data. *J. American Statistical Assoc.*, 93(442):557–576, 1998.

[94] Y. Vardi. Network tomography: estimating source-destination traffic intensities from link data. *J. American Statistical Assoc.*, 91:365–377, 1996.

[95] J.-P. Vasseur, M. Pickavet, and P. Demesteer. *Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS.* Morgan Kaufmann, 2004.

[96] H. Wietgrefe, K. Tochs, et al. Using neural networks for alarm correlation in cellular phone networks. In *Proc. International Workshop on Applications of Neural Networks in Telecommunciations*, 1997.

[97] A. Yaar, A. Perrig, and D. Song. Siff: A stateless internet flow filter to mitigate ddos flooding attacks. In *IEEE Symposium on Security and Privacy*, 2004.

[98] A. Yaar, A. Perrig, and D. X. Song. Pi: A path identification mechanism to defend against DDoS attack. In *IEEE Symposium on Security and Privacy*, 2003.

[99] S. A. Yemini, S. Kliger, E. Mozes, Y. Yemini, and D. Ohsie. High speed and robust event correlation. In *IEEE Communications*, volume 34, pages 82–90, 1996.

[100] L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reservation protocol (rsvp) – version 1 functional specification. RFC 2205, IETF, Sept. 1997.

[101] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale ip traffic matrices from link loads. In *ACM SIGMETRICS 2003*, June 2003.

[102] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information-theoretic approach to traffic matrix estimation. In *ACM SIGCOMM*, August 2003.

[103] R. Zhang-Shen and N. McKeown. Designing a predictable Internet backbone network. In *ACM HotNets*, Nov. 2004.

[104] Y. Zhao, Y. Chen, and D. Bindel. Towards unbiased end-to-end network diagnosis. In *ACM SIGCOMM*, 2006.